Signetics

Microprocessor Data Manual 1986

Signetics

Microprocessor Products

Data Manual 1986

Microprocessor Products

Signetics reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Signetics assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement. Applications that are described herein for any of these products are for illustrative purposes only. Signetics makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Signetics registers eligible circuits under
the Semiconductor Chip Protection Act.

Signetics

Contents

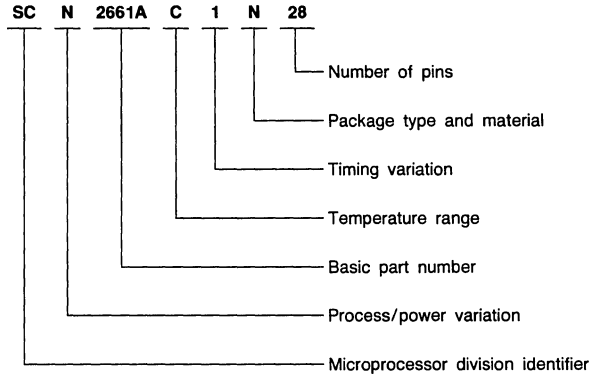
Microprocessor Products

Contents

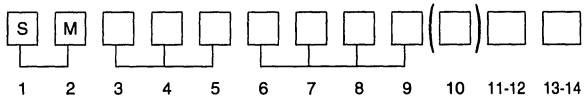
Ordering Information

Microprocessor Products

Microprocessor Part Numbers:



Microsystems Part Number Template:



Pos.	Description
1 – 2	SM to signify Signetics Microsystems
3 – 5	Indicates Product Family — VME — Modular Board Products DEV — Development Support SFT — Software Support MAN — Manuals KIT — Development Kits
6 – 9(10)	Individual Product Identifier — number to be defined by Product Marketing Manager 0xxx — Accessories 1xxx — Other Modules 2xxx — CPU Modules 3xxx — Memory Modules 4xxx — Mass Storage Modules 5xxx — Communication Modules 6xxx — Industrial I/O Modules 7xxx — Video Modules 8xxx — Target Software 9xxx — Evaluation Kits
11 – 12	IC Package Type Identifier — we will use SD as our standard package identifier
13 – 14	Revision Identifier

Product Status

DEFINITIONS		
Data Sheet Identification	**Product Status**	**Definition**
Objective Specification	**Formative or In Design**	This data sheet contains the design target or goal specifications for product development. Specifications may change in any manner without notice.
Preliminary Specification	**Preproduction Product**	This data sheet contains preliminary data and supplementary data will be published at a later date. Signetics reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
Product Specification	**Full Production**	This data sheet contains Final Specifications. Signetics reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

Signetics

Microprocessor Products

Section 1
Quality and Reliability

1

January 1986

Quality and Reliability

Microprocessor Products

INTRODUCTION

The Microprocessor Division was formed in 1979 when it became apparent that microprocessor circuits and their peripheral devices would become increasingly important system components to the OEM customer base. Because of performance to cost ratios, reduced power requirements and inherent reliability, these components provide the solution to designers' complex system requirements.

NMOS became the dominant technology because of the simplicity of its design and processing and its large historical reliability data base. To date, NMOS has serviced 95% of the division's device requirements. Bipolar devices are used to a limited extent as interface drivers between the NMOS system components and CRT displays.

The Microprocessor Division serves a broad base of customers with its product lines of eight and sixteen bit microprocessors, microcontrollers, and data communication and CRT peripheral circuits. The division's customers range from large mainframe computer manufacturers to small systems users, and the division can generally satisfy all the microprocessor requirements of any customer.

Our goal is to establish ourselves as the preferred alternate source in both the 68000 microprocessor and the 80XX microcontroller product areas and at the same time become the prime supplier of the interface and peripheral circuits required to integrate the parts within a system. Our design philosophy is to give superior performance while minimizing the overall number of parts required for the overall system.

The terms quality and reliability are often misinterpreted. In general, quality refers to the condition of a device when received; reliability covers extent of useful life. Quality is readily measurable; reliability is predictable and verifiable based on historical evidence.

MICROPROCESSOR DIVISION RELIABILITY

No amount of stress testing can improve a product's reliability. Stress tests are used to measure and define an end of life which can be expected from a family of products. Reliability as well as quality must be built in through proper design, processing, assembly, testing and handling. For Signetics' Microprocessor Division parts, accelerated life test data show an extremely reliable product line and serve as one input for continuous product reliability improvement.

DESIGN

Product quality and reliability begin in design. Strategic questions directly affecting reliability must be answered. How much static protection is required on input leads? Do formal design rules exist? Can they ever be violated? What method is used to anticipate future processes or "shrinks"?

Within Signetics' Microprocessor Division, rigid guidelines are in effect to ensure compliance with our design rules. Design rules, once established, are inviolate.

QUALIFICATION TESTING

Signetics' Microprocessor Division verifies device reliability through a series of qualification tests and a continuous reliability monitor program, Sure III (Systematic Uniform Reliability Evaluation).

All new fab processes at Signetics are qualified by stress testing parts from a variety of production lots. This accelerated stress testing is shown in the table.

SURE III

Continuous reliability monitoring is performed via our SURE III program. Devices are randomly selected from production lots and subjected to the same environmental stresses noted in the table. The program is administered by the Corporate Reliability Engineering Group, which publishes a summary of results on a quarterly basis.

The SURE III program covers two functions: Monitoring short term and long term reliability performance.

LONG-TERM AUDIT

One hundred devices from each generic family are subjected to each of the following stresses every other four weeks:

- High Temperature Operating Life — T_J = 150°C, 1000 hours — (Static Biased or Dynamic Operation, as appropriate);
- Temperature-Humidity Biased Life — 85°C, 85% RH, 1000 hours, static biased;
- Temperature Cycling (Air-Air) — −65°C to +150°C, 1000 cycles.

SHORT-TERM MONITOR

Every week 20-piece samples from each generic family are run to 96 hours of pressure pot (15 psig, 121°C, 100% saturated steam), 300 cycles of thermal shock (−65°C to +150°C) and 168 hours of high temperature operating life (T_J = 150°C, static or dynamic operation).

In addition, each Signetics assembly plant performs SURE product monitor stresses weekly on each generic family and molded package, by pin count and frame type. Fifty pieces are subjected to 300 cycles of thermal shock (Cond. C) and 100 devices are subjected to pressure pot stress at 20 psig for 72 hours (168 hour equivalent at 15 psig).

Quality and Reliability

Accelerated Life Stress Tests

TEST		TEST CONDITION	NUMBER OF DEVICES	DURATION
DHTL	Dynamic high temperature life	T_A@125°C (operating)	52	1000 hrs
SHTL	Static high temperature life	T_A@125°C (operating)	52	1000 hrs
THBS	Biased temperature humidity life	85°C/85% RH (operating)	52	1000 hrs
PPOT	Pressure pot (autoclave)	121°C/15 psig (storage)	52	96 hrs
TMSK	Thermal shock	−55°C/125°C liq. to liq. (storage)	77	300 cycles
TMCL	Temperature cycle (air-to-air)	−65°C/150°C (storage)	77	300 cycles

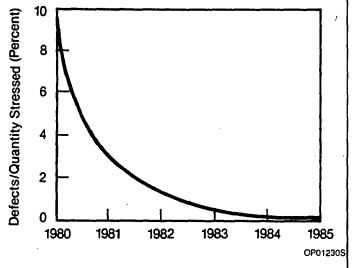


Figure 1. Microprocessor Reliability Index

Figure 1 shows the relative improvement in Sure III performance for the Microprocessor Division for the past five years. This curve combines all the stress tests and plots the overall percentage defective by year. Clearly, the reliability of our product line is improving, and more importantly, it is consistent and predictable.

SIGNETICS' QUALITY IMPROVEMENT PROGRAM

Signetics began its Quality Improvement Program in 1980, and developed it around the concepts outlined in the book, "Quality is Free," by Phil Crosby.

This program, which is actively supported by top management, defines Quality as "Conformance to the Specification." With this definition in mind, our performance standard is "Zero Defects." Tracking charts measuring quality improvement targets are used and displayed throughout the Division.

Microprocessor Division personnel are all actively involved in this program. Our people have taken formal training in Quality College and have pledged to "Do It Right the First Time, On Time." Administrative personnel promise to "Make Certain" of their own work.

The program improves quality through education, commitment and feedback.

Internally, there are many signs of the program's success. The SURE III Reliability Assurance Monitor shows improved results in each of the past three years. In-line quality improvements are impressive. All new products are placed on QRA Hold until completion of environmental stress testing; all significant process changes go through qualification prior to release to production; test programs are controlled and released only after extensive engineering correlation; wafers with less than the required minimum number of good die are scrapped to avoid jeopardizing product quality.

These are examples of how the Quality Improvement Program functions internally. However, the real measure of the program is customer belief in and acceptance of our improvement.

Over the past few years, customers have verified a continuous improvement in the quality and reliability of Microprocessor Division parts. Figure 2 shows the improvement in visual mechanical and electrical outgoing quality levels over the past several years.

In addition to the marked improvement in our AOQ (Average Outgoing Quality), we have become a qualified source for the 80XX and the 68000 families, as well as our own proprietary peripheral circuits, for more than 200 customers. We have developed Ship-to-Stock programs for several key accounts, eliminating the need for costly incoming inspection by our customers.

The improved reliability of our parts has allowed several customers to drop their requirements for burn-in for high reliability programs. This, of course, has lowered their costs significantly and has allowed them to improve their over-all system reliability.

Reliability testing is an important monitor of our manufacturing process. Signetics' microprocessor parts not only meet specifications when shipped but continue to operate satisfactorily throughout their lifetime.

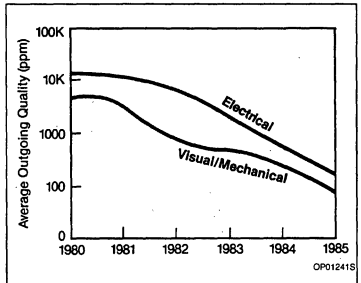


Figure 2. Outgoing Quality Level Microprocessor Division

Signetics

Section 2
Microprocessor Products

INDEX

SCN2641
Asynchronous Communications Interface

Product Specification

Microprocessor Products

DESCRIPTION

The Signetics SCN2641 is a universal asynchronous data communications controller chip that interfaces directly to most 8-bit microprocessors and may be used in a polled or interrupt-driven system environment. The SCN2641 accepts programmed instructions from the microprocessor while supporting asynchronous serial data communications in full- or half-duplex mode.

The SCN2641 serializes parallel data characters received from the microprocessor for transmission. Simultaneously, it can receive serial data and convert it into parallel data characters for input to the microcomputer.

The SCN2641 contains a baud rate generator which can be programmed to either accept an external clock or to generate internal transmit or receive clocks. Sixteen different baud rates can be selected under program control when operating in the internal clock mode.

The SCN2641 is constructed using Signetics n-channel silicon gate depletion load technology and is packaged in a 24-pin DIP.

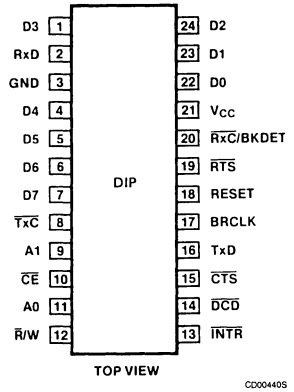
FEATURES

- **5- to 8-bit characters plus parity**
- **1, 1½ or 2 stop bits transmitted**
- **Odd, even or no parity**
- **Parity, overrun and framing error detection**
- **Line break detection and generation**
- **False start bit detection**
- **Automatic serial echo mode (echoplex)**
- **Local or remote maintenance loopback mode**
- **Baud rate:**
  - **DC to 1M bps (1X clock)**
  - **DC to 62.5K bps (16X clock)**
  - **DC to 15.625K bps (64X clock)**
- **Internal or external baud rate clock**
- **16 internal rates**
- **Double-buffered transmitter and receiver**
- **Single +5V power supply**
- **400 mil package width**

APPLICATIONS

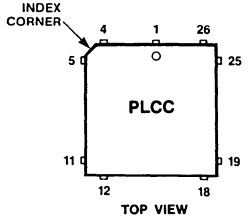
- **Intelligent terminals**
- **Network processors**
- **Front-end processors**
- **Remote data concentrators**
- **Serial peripherals**

PIN CONFIGURATION



TOP VIEW

CD00440S



TOP VIEW

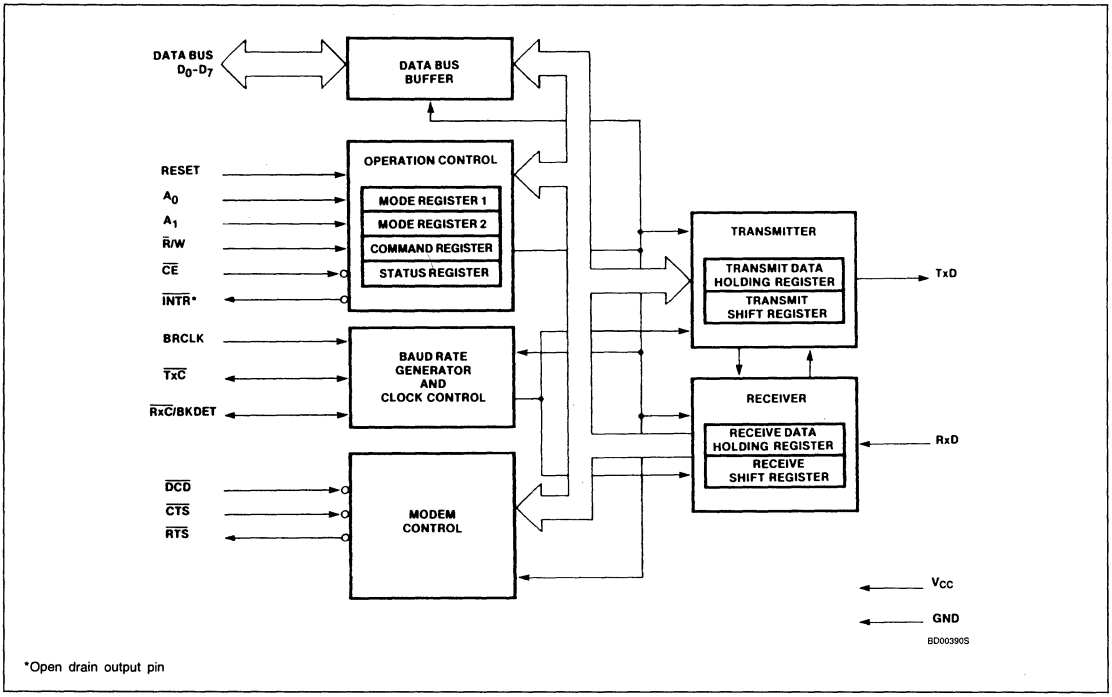
CD0028PS

Pin	Function	Pin	Function
1	D2	15	INTR
2	D3	16	DCD
3	RxD	17	CTS
4	GND	18	NC
5	D4	19	TxD
6	D5	20	BRCLK
7	D6	21	RESET
8	NC	22	NC
9	D7	23	RTS
10	TxC	24	NC
11	A1	25	RxC/BKDET
12	CE	26	VCC
13	A0	27	D0
14	R/W	28	D1

ORDERING CODE

PACKAGES	V_{CC} = 5V± 5%, T_A = 0°C to 70°C
Plastic DIP	SCN2641CC1N24
Plastic LCC	SCN2641CC1A28

BLOCK DIAGRAM



*Open drain output pin

BLOCK DIAGRAM

The SCN2641 consists of five major sections. These are the transmitter, receiver, timing, operation control and modem control. These sections communicate with each other via an internal data bus and an internal control bus. The internal data bus interfaces to the microprocessor data bus via a data bus buffer.

Operation Control

This functional block stores configuration and operation commands from the CPU and generates appropriate signals to various internal sections to control the overall device operation. It contains read and write circuits to permit communications with the microprocessor via the data bus and contains mode registers 1 and 2, the command register, and the status register. Details of register addressing are presented in the SCN2641 programming section of this data sheet.

Timing

The SCN2641 contains a baud rate generator (BRG) which is programmable to accept external transmit or receive clocks or to divide an external clock to perform data communications. The unit can generate 16 commonly used baud rates, any one of which can be selected for full-duplex operation. See table 1.

Receiver

The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for certain errors and sends an "assembled" character to the CPU.

Transmitter

The transmitter accepts parallel data from the CPU, appends start and stop bits, and, optionally, a parity bit, and outputs a composite serial stream of data on the TxD output pin.

Modem Control

The modem control section provides interfacing for two input signals and one output signal used for "handshaking" and status indication between the CPU and a modem.

INTERFACE SIGNALS

The SCN2641 interface signals can be grouped into two types: the CPU-related signals (shown in table 2), which interface the SCN2641 to the microprocessor system and the device-related signals (shown in table 3), which are used to interface to the communications device or system.

OPERATION

The functional operation of the SCN2641 is programmed by a set of control words supplied by the CPU. These control words speci-

Table 1. BAUD RATE GENERATOR CHARACTERISTICS (BRCLK = 3.6864MHz)

MR23 – 20	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	50	0.8kHz	—	4608
0001	75	1.2	—	3072
0010	110	1.7596	-0.022	2095
0011	134.5	2.152	—	1713
0100	150	2.4	—	1536
0101	300	4.8	—	768
0110	600	9.6	—	384
0111	1200	19.2	—	192
1000	1800	28.8	—	128
1001	2000	32.055	0.174	115
1010	2400	38.4	—	96
1011	3600	57.6	—	64
1100	4800	76.8	—	48
1101	7200	115.2	—	32
1110	9600	153.6	—	24
1111	19200	307.2	—	12

fy items such as baud rate, number of bits per character, etc. The programming procedure is described in the SCN2641 programming section of this data sheet.

After programming, the SCN2641 is ready to perform the desired communications functions. The receiver performs serial to parallel conversion of data received from a modem or equivalent device. The transmitter converts parallel data received from the CPU to a serial bit stream. These actions are accomplished within the framework specified by the control words.

Receiver

The SCN2641 is conditioned to receive data when the $\overline{DCD}$ input is low and the RxEN bit in the command register is true. The receiver looks for a high-to-low transition of the start bit on the RxD input line. If a transition is detected, the state of the RxD line is sampled again after a delay of one-half of a bit time. If RxD is now high, the search for a valid start bit is begun again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input line at one bit time intervals until the proper number of data bits, the parity bit, and one stop bit have been assembled. The data is then transferred to the receive data holding register, the RxRDY bit in the status register is set, and the $\overline{INTR}$ output is asserted. If the character length is less than 8 bits, the high-order unused bits in the holding register are set to zero. The parity error, framing error, and overrun error status bits are strobed into the status register on the positive-going edge of $\overline{RxC}$ corresponding to the received character boundary. If the stop bit is present, the receiver will immediately begin its search for the next start bit. If the stop bit is absent (framing error), the receiver will interpret a space as a start bit if it persists into the next bit time interval. If a break condition is detected (RxD is low for the entire character as well as the stop bit), only one character consisting of all zeros (with the FE status bit set) will be transferred to the holding register. The RxD input must return to a high condition before a search for the next start bit begins.

Pin 20 can be programmed to be a break detect output by appropriate setting of MR27 – MR24. If so, a detected break will cause that pin to go high. When RxD returns to mark for one RxC time, pin 20 will go low. Refer to the break detection timing diagram.

Transmitter

The SCN2641 is conditioned to transmit data when the $\overline{CTS}$ input is low and the TxEN command register bit is set. The SCN2641 indicates to the CPU that it can accept a character for transmission by setting the TxRDY status bit and asserting the $\overline{INTR}$ output. When the CPU writes a character into the transmit data holding register, these conditions are negated. Data is transferred from the holding register to the transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again. Thus, one full character time of buffering is provided.

The transmitter automatically sends a start bit followed by the programmed number of data bits, the least significant bit being sent first. It then appends an optional odd or even parity bit and the programmed number of stop bits. If, following transmission of the data bits, a new character is not available in the transmit holding register, the TxD output remains in the marking (high) condition and the TxEMT/DSCHG status bit and the $\overline{INTR}$ output are asserted. Transmission resumes when the CPU loads a new character into the holding register. The transmitter can be forced to output a continuous low (BREAK) condition by setting the send break command bit (CR3) high.

Table 2. CPU-RELATED SIGNALS

PIN NAME	PIN NO. DIP	PIN NO. PLCC	INPUT/ OUTPUT	FUNCTION
V_{CC}	21	26	I	+5V supply input
GND	3	4	I	Ground
RESET	18	21	I	A high on this input performs a master reset on the SCN2641. This signal asynchronously terminates any device activity and clears the mode, command and status registers. The device assumes the idle state and remains there until initialized with appropriate control words.
$A_1 - A_0$	9, 11	11, 13	I	Address lines used to select internal SCN2641 registers.
$\overline{R}/W$	12	14	I	Read command when low, write command when high.
$\overline{CE}$	10	12	I	Chip enable command. When low, indicates that control and data lines to the SCN2641 are valid and that the operation specified by the $\overline{R}/W$, A_1 and A_0 inputs should be performed. When high, places the $D_0 - D_7$ lines in the three-state condition.
$D_7 - D_0$	7-4,1, 24-22	9,7-5, 2,1, 28,27	I/O	8-bit, three-state data bus used to transfer commands, data and status between the SCN2641 and the CPU. D_0 is the least significant bit; D_7 the most significant bit.
$\overline{INTR}$	13	15	O	Interrupt request output (open drain). This output is asserted (low) under the following conditions. 1. When the transmitter holding register (THR) is ready to accept a data character from the CPU. This corresponds to assertion of status bit SR0. If this is the only condition asserting the output, the output will be negated (high) when the THR is loaded by the CPU, or if the transmitter is disabled via command register bit CR0. 2. When the receiver holding register (RHR) has a character ready to be read by the CPU. This corresponds to assertion of status bit SR1. If this is the only condition asserting the output, the output will be negated (high) when the RHR is read by the CPU, or if the receiver is disabled via command register bit CR2. 3. When the transmitter has completed serialization of the last character loaded by the CPU. This corresponds to assertion of status bit SR2. If this is the only condition asserting the output, the output will be negated (high) when the THR is loaded by the CPU. 4. When a change of state has occurred at the $\overline{DCD}$ input while either the receiver or the transmitter are enabled. This corresponds to assertion of status bit SR2. If this is the only condition asserting the output, the output will be negated (high) when the status register is read by the CPU.

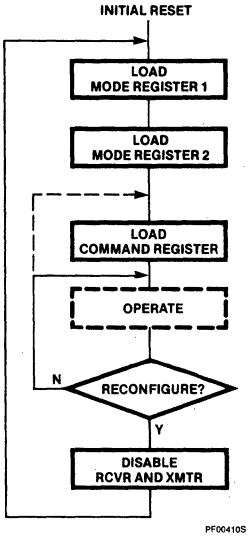
PROGRAMMING

Prior to initiating data communications, the SCN2641 operational mode must be programmed by performing write operations to the mode and command registers. The SCN2641 can be reconfigured at any time during program execution. A flowchart of the initialization process appears in figure 1.

The internal registers of the SCN2641 are accessed by applying specific signals to the $\overline{CE}$, $\overline{R}/W$, A_1 and A_0 inputs. The conditions necessary to address each register are shown in table 4.

Reading or loading the mode registers is done as follows: the first write (or read) operation addresses mode register 1 and a subsequent operation addresses mode register 2. If more than the required number of accesses are made, the internal sequencer recycles to point at the first register. The pointer is reset to mode register 1 by a RESET input or by performing a read command register operation, but is unaffected by any other read or write operation.

The SCN2641 register formats are summarized in tables 5, 6, 7 and 8. Mode registers 1 and 2 define the general operational characteristics of the SCN2641, while the command register controls the operation within this basic framework. The SCN2641 indicates its status in the status register. These registers are cleared when a RESET input is applied.



NOTE:
Mode Register 1 must be written before 2 can be written. Mode register 2 need not be programmed if external clocks are used.

Figure 1. SCN2641 Initialization Flowchart

Mode Register 1 (MR1)

Table 5 illustrates mode register 1. Bits MR11 and MR10 select the baud rate multiplier. 1X, 16X and 64X multipliers are programmable if the external clock input option is selected by MR24 or MR25.

MR13 and MR12 select a character length of 5, 6, 7 or 8 bits. The character length does not include the parity bit, if programmed, and does not include the start and stop bits.

MR14 controls parity generation. If enabled, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR15 selects odd or even parity when parity is enabled by MR14.

MR17 and MR16 select character framing of 1, 1.5 or 2 stop bits. (If 1X baud rate is

programmed, 1.5 stop bits default to 1 stop bit on transmit.)

The bits in the mode register affecting character assembly and disassembly (MR12 – MR16) can be changed dynamically (during active receive/transmit operation). The character mode register affects both the transmitter and receiver; therefore, character changes should be made when RxEN and TxEN = 0 or when TxEN = 1 and the transmitter is marking in half-duplex mode (RxEN = 0).

To effect assembly/disassembly of the next received/transmitted character, MR12 – MR15 must be changed within n-bit times of the assertion of RxRDY/TxRDY. (n = smaller of the new and old character lengths.)

Mode Register 2 (MR2)

Table 6 illustrates mode register 2. MR23, MR22, MR21 and MR20 control the frequency of the internal baud rate generator (BRG). Sixteen rates are selectable as per table 1. MR23 – MR20 are don't cares if external clocks are selected (MR25 – MR24 = 0). The individual rates are given in table 1.

MR24 – MR27 select the receive and transmit clock source (either the BRG or an external input) and the function at pins 8 and 20. Refer to table 6.

Command Register (CR)

Table 7 illustrates the command register. Bits CR0 (TxEN) and CR2 (RxEN) enable or disable the transmitter and receiver respectively. If the transmitter is disabled, it will complete the transmission of the character in the transmit shift register (if any) prior to terminating operation. The TxD output will then remain in the marking state (high), while the TxRDY and TxEMT status bits go low. Disabling the receiver causes the RxRDY status bit to go low. If the receiver is disabled, it will terminate operation immediately. Any character being assembled will be negated. A 0-to-1 transition of CR2 will initiate start bit search on the second $\overline{RxC}$ rising edge following the transition.

Bit CR5 (RTS) controls the RTS output. Data at the output is the logical complement of the register data.

Setting CR3 will force and hold the TxD output low (spacing condition) at the end of the current transmitted character. Normal operation resumes when CR3 is cleared. The user should wait at least one bit time after terminating the break before loading the THR with the next character to be transmitted.

Setting CR4 causes the error flags in the status register (SR3, SR4 and SR5) to be cleared. This is a one-time command. There is no internal latch for this bit.

Table 3. DEVICE-RELATED SIGNALS

PIN NAME	DIP	PLCC	INPUT/ OUTPUT	FUNCTION
BRCLK	17	20	I	Clock input to the internal baud rate generator (see table 1). Not required if external receiver and transmitter clocks are used.
$\overline{RxC}$/ BKDET	20	25	I/O	Receiver clock. If external receiver clock is programmed, this input controls the rate at which the character is to be received. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. Data are sampled on the rising edge of the clock. If internal receiver clock is programmed, this pin can be a 1X/16X clock or a break detect output pin.
RxD	2	3	I	Serial data input to the receiver. "Mark" is high, "space" is low.
$\overline{TxC}$	8	10	I/O	Transmitter clock. If external transmitter clock is programmed, this input controls the rate at which the character is transmitted. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. The transmitted data changes on the falling edge of the clock. If internal transmitter clock is programmed, this pin can be a 1X/16X clock output.
TxD	16	19	O	Serial data output from the transmitter. "Mark" is high, "space" is low. Held in mark condition when the transmitter is disabled.
$\overline{DCD}$	14	16	I	Data carrier detect input. Must be low in order for the receiver to operate. Its complement appears as status register bit SR6. Causes a low output on $\overline{INTR}$ when its state changes if CR2 or CR0 = 1. If DCD goes high while receiving, the RxC is internally inhibited. Operation of the receiver resumes on the second $\overline{RxC}$ rising edge following assertion of DCD.
$\overline{CTS}$	15	17	I	Clear to send input. Must be low in order for the transmitter to operate. If it goes high during transmission, the character in the transmit shift register will be transmitted before termination.
$\overline{RTS}$	19	23	O	General-purpose output which is the complement of command register bit CR5. Normally used to indicate request to send. See Command Register (CR5) for details.

Table 4. REGISTER ADDRESSING

$\overline{CE}$	A_1	A_0	$\overline{R}/W$	FUNCTION
1	X	X	X	Three-state data bus
0	0	0	0	Read receive holding register
0	0	0	1	Write transmit holding register
0	0	1	0	Read status register
0	0	1	1	Invalid
0	1	0	0	Read mode registers ½
0	1	0	1	Write mode registers ½
0	1	1	0	Read command register
0	1	1	1	Write command register

NOTE:
See AC characteristics section for timing requirements.

When CR5 (RTS) is set, the RTS pin is forced low. A 1-to-0 transition of CR5 will cause $\overline{RTS}$ to go high (inactive) one TxC time after the last serial bit has been transmitted. If a 1-to-0 transition of CR5 occurs while data is being transmitted, $\overline{RTS}$ will remain low (active) until both the THR and the transmit shift register are empty and then go high one TxC time later.

The SCN2641 can operate in one of four submodes. The operational submode is determined by CR7 and CR6. CR7 – CR6 = 00 is the normal mode, with the transmitter and receiver operating independently in accordance with the mode and status register instructions.

CR7 – CR6 = 01 places the SCN2641 in the automatic echo mode. Clocked, regenerated received data are automatically directed to the TxD line while normal receiver operation continues. The receiver must be enabled (CR2 = 1), but the transmitter need not be enabled. CPU-to-receiver communications continue normally, but the CPU-to-transmitter link is disabled. Only the first character of a break condition is echoed. The TxD output will go high until the next valid start is detected. The following conditions are true while in automatic echo mode:

1. Data assembled by the receiver are automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. The $\overline{INTR}$ pin will reflect only the data set change condition.
4. The TxEN command (CR0) is ignored.

Two diagnostic submodes can also be configured. In local loopback mode (CR7 – CR6 = 10), the following loops are connected internally:

1. The transmitter output is connected to the receiver input.
2. $\overline{RTS}$ is connected to $\overline{CTS}$.
3. The receiver is clocked by the transmitter clock.
4. The $\overline{RTS}$ and TxD outputs are held high.
5. The $\overline{CTS}$, $\overline{DCD}$ and RxD inputs are ignored.

Additional requirements to operate in the local loopback mode are that CR0 (TxEN), CR1 and CR5 (RTS) must be set to 1. CR2 (RxEN) is ignored by the SCN2641.

The second diagnostic mode is the remote loopback mode (CR7 – CR6 = 11). In this mode:

1. Data assembled by the receiver are automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. No data is sent to the local CPU, but the error status conditions (PE, OE, FE) are set.
4. The $\overline{INTR}$ output is held high.
5. CR0 (TxEN) is ignored.
6. All other signals operate normally.

Status Register

The data contained in the status register (as shown in table 8) indicate receiver and transmitter conditions and modem/data set status.

SR0 is the transmitter ready (TxRDY) status bit. It is valid only when the transmitter is enabled. If equal to 0, it indicates that the transmit holding register has been loaded by the CPU and the data has not been transferred to the transmit shift register. If set equal to 1, it indicates that the holding register is ready to accept data from the CPU. This bit is initially set when the transmitter is enabled by CR0, unless a character has previously been loaded into the holding register. It is not set when the automatic echo or remote loopback modes are programmed. When this bit is set, the $\overline{INTR}$ output pin is low, except in the automatic echo and remote loopback modes.

SR1, the receiver ready (RxRDY) status bit, indicates the condition of the receive data holding register. If set, it indicates that a character has been loaded into the holding register from the receive shift register and is ready to be read by the CPU. If equal to 0, there is no new character in the holding register. This bit is cleared when the CPU reads the receive data holding register or when the receiver is disabled by CR2. When set, the $\overline{INTR}$ output is low.

Table 5. MODE REGISTER 1 (MR1)

MR17	MR16	MR15	MR14	MR13	MR12	MR11	MR10
Stop Bit Length		Parity Type	Parity Control	Character Length		Mode and Baud Rate Factor	
00 = Invalid 01 = 1 stop bit 10 = 1½ stop bits 11 = 2 stop bits		0 = Odd 1 = Even	0 = Disabled 1 = Enabled	00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits		00 = Invalid 01 = 1X rate 10 = 16X rate 11 = 64X rate	

NOTE:
Baud rate factor applies only if external clock is selected. Factor is 16X if internal clock is selected.

Table 6. MODE REGISTER 2 (MR2)

MR27 – MR24										MR23 – MR20
	TxC	RxC	Pin 8	Pin 20		TxC	RxC	Pin 8	Pin 20	Baud Rate Selection
0000	E	E	TxC	RxC	1000	E	E	NF	RxC/TxC	
0001	E	I	TxC	1X	1001	E	I	TxC	BKDET	
0010	I	E	1X	RxC	1010	I	E	NF	RxC	
0011	I	I	1X	1X	1011	I	I	1X	BKDET	See baud rates in table 1
0100	E	E	TxC	RxC	1100	E	E	NF	RxC/TxC	
0101	E	I	TxC	16X	1101	E	I	TxC	BKDET	
0110	I	E	16X	RxC	1110	I	E	NF	RxC	
0111	I	I	16X	16X	1111	I	I	16X	BKDET	

NOTES:
E = External clock
I = Internal clock (BRG)
NF = No function; output not valid
1X and 16X are clock outputs

Table 7. COMMAND REGISTER (CR)

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Operating Mode		Request To Send	Reset Error	Force Break	Receive Control (RxEN)		Transmit Control (TxEN)
00 = Normal operation 01 = Automatic echo mode 10 = Local loopback 11 =Remote loopback		0 = Force $\overline{RTS}$ output high after TxSR serialization 1 = Force $\overline{RTS}$ output low	0 = Normal 1 = Reset error flags in status register (FE, O	0 = Normal 1 = Force break	0 = Disable 1 = Enable	Not used. Must be programmed to '1'	0 = Disable 1 = Enable

Table 8. STATUS REGISTER (SR)

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
	Data Carrier Detect	Framing Error	Overrun Error	Parity Error	TxEMT/DSCHG	RxRDY	TxRDY
Not used	0 = $\overline{DCD}$ input is high 1 = $\overline{DCD}$ input is low	0 = Normal 1 = Framing Error	0 = Normal 1 = Overrun Error	0 = Normal 1 = Parity error	0 =Normal 1 =Change in $\overline{DCD}$ or transmit shift register is empty	0 = Receive holding register empty 1 = Receive holding register has data	0 = Transmit holding register busy 1 = Transmit holding register busy

The TxEMT/DSCHG bit, SR2, when set, indicates either a change of state of the $\overline{DCD}$ input (when CR2 or CR0 = 1) or that the transmit shift register has completed transmission of a character and no new character has been loaded into the transmit data holding register. TxEMT will not go active until at least one character has been transmitted. It is cleared by loading the transmit data holding register. The DSCHG condition is enabled when the TxEN = 1 or RxEN = 1. It is cleared when the status register is read by the CPU. If the status register is read twice and SR2 = 1 while SR6 remains unchanged, then a TxEMT condition exists. When SR2 is set, the $\overline{INTR}$ output is low.

SR3, when set, indicates a received parity error when parity is enabled by MR14. This bit is cleared when the receiver is disabled and by a reset error command, CR4.

The overrun error status bit, SR4, indicates that the previous character loaded into the receive holding register was not read by the CPU at the time a new received character was transferred into it. This bit is cleared when the receiver is disabled and by the reset error command, CR4.

Bit SR5 signifies that the received character was not framed by a stop bit; i.e., only the first stop bit is checked. If RHR = 0 when SR5 = 1, a break condition is present. The bit is reset when the receiver is disabled and when the reset error command is given.

SR6 reflects the condition of the DCD input. A low input sets the status bit and a high input clears it.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Input voltage						V
V_{IL}	Low				0.8	
V_{IH}	High		2.0			
Output voltage						V
V_{OL}	Low	I_{OL} = 2.2mA			0.4	
V_{OH}[7]	High	I_{OH} = −400μA	2.4			
I_{IL}	Input leakage current	V_{IN} = 0 to V_{CC}			10	μA
3-state output leakage current						μA
I_{LH}	Data bus high	V_O = 4.0V			10	
I_{LL}	Data bus low	V_O = 0.45V			10	
I_{CC}	Power supply current				150	mA

CAPACITANCE T_A = 25°C, V_{CC} = 0V

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Capacitance						pF
C_{IN}	Input	fc = 1MHz			20	
C_{OUT}	Output	Unmeasured pins tied to ground			20	
$C_{I/O}$	Input/Output				20	

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6]

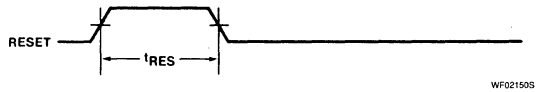
PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Pulse width						ns
t_{RES}	Reset		1000			
t_{CE}	Chip enable		250			
t_{CED}	CE to CE delay		600			
Set-up and hold time						ns
t_{AS}	Address set-up		10			
t_{AH}	Address hold		10			
t_{CS}	$\overline{R}/W$ control set-up		10			
t_{CH}	$\overline{R}/W$ control hold		10			
t_{DS}	Data set-up for write		150			
t_{DH}	Data hold for write		10			
t_{RXS}	Rx data set-up		300			
t_{RXH}	Rx data hold		350			
t_{DD}	Data delay time for read	$C_L = 150pF$			200	ns
t_{DF}	Data bus floating time for read	$C_L = 150pF$			100	ns
Input clock frequency						MHz
f_{BRG}	Baud rate generator		1.0	3.6864	4.0	
$f_{R/T}$[10]	$\overline{TxC}$ or $\overline{RxC}$		dc		1.0	
Clock state						ns
t_{BRH}[9]	Baud rate high		90			
t_{BRL}[9]	Baud rate low		90			
$t_{R/TH}$[10]	$\overline{TxC}$ or $\overline{RxC}$ high		480			
$t_{R/TL}$[10]	$\overline{TxC}$ or $\overline{RxC}$ low		480			
t_{TXD}	TxD delay from falling edge of $\overline{TxC}$	$C_L = 150pF$			650	ns
t_{TCS}	Skew between TxD changing and falling edge of $\overline{TxC}$ output[8]	$C_L = 150pF$		0		ns

NOTES:

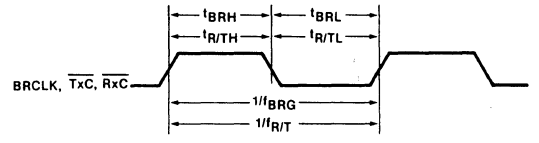
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground. All time measurements are at the 50% level for inputs (except t_{BRH} and t_{BRL}) and at 0.8V and 2.0V for outputs. Input levels swing between 0.4V and 2.4V, with a transition time of 20ns maximum.
6. Typical values are at +25°C, typical supply voltages and typical processing parameters.
7. $\overline{INTR}$ output is open drain.
8. Parameter applies when internal transmitter clock is used.
9. t_{BRH} and t_{BRL} measured at V_{IH} and V_{IL} respectively.
10. In asynchronous local loopback mode, using 1X clock, the following parameters apply:
    $f_{R/T} = 0.83MHz$ max
    $t_{R/TL} = 700ns$ min

TIMING DIAGRAMS

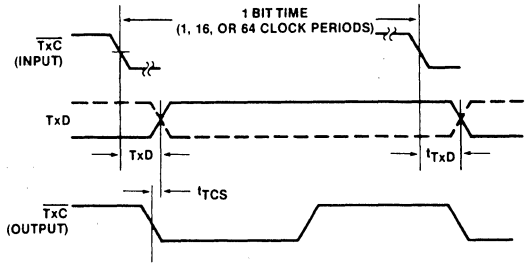
RESET



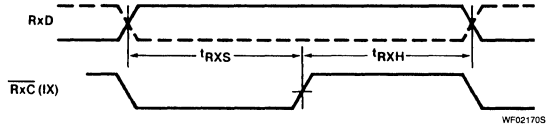
CLOCK



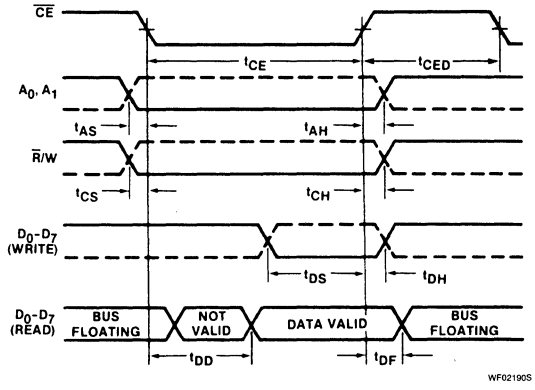
TRANSMIT



RECEIVE

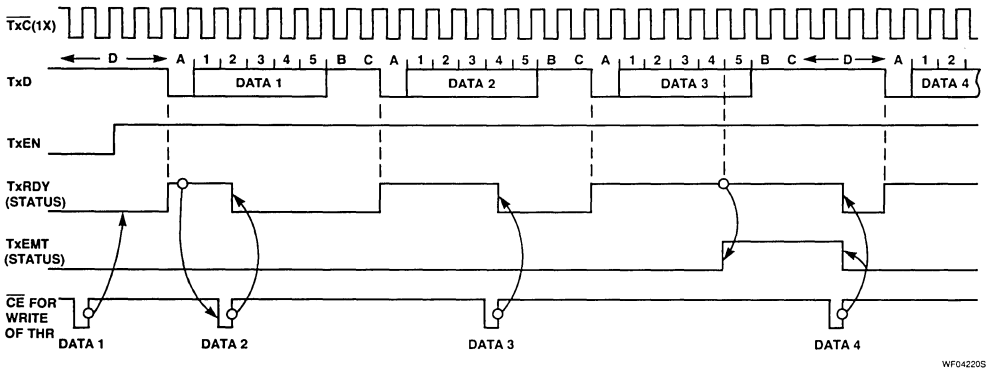


READ AND WRITE



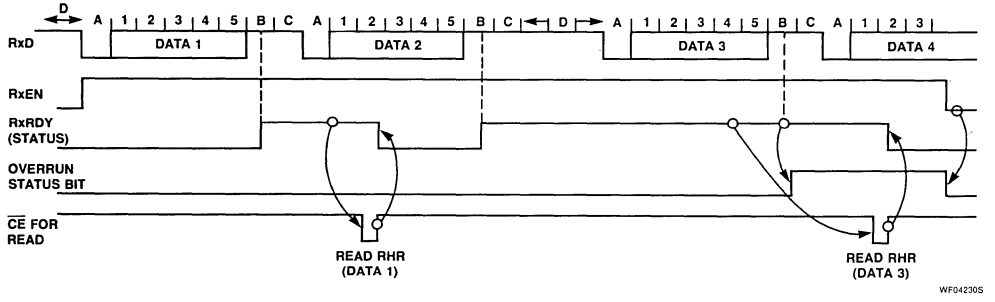
TIMING DIAGRAMS (Continued)

TxRDY, TxEMT (Shown for 5-bit characters, no parity, 2 stop bits)



NOTES:
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
TxEMT goes low at the beginning of the last data bit, or, if parity is enabled, at the beginning of the parity bit.

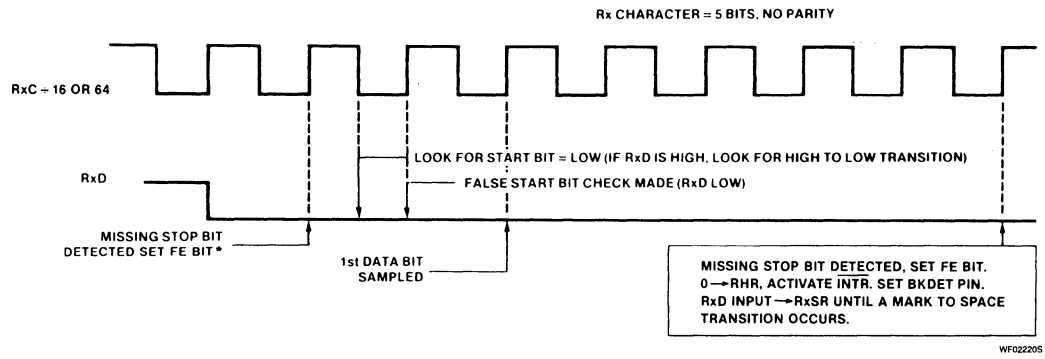
RxRDY (Shown for 5-bit characters, no parity, 2 stop bits)



NOTES:
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
Only one stop bit is detected.

TIMING DIAGRAMS (Continued)

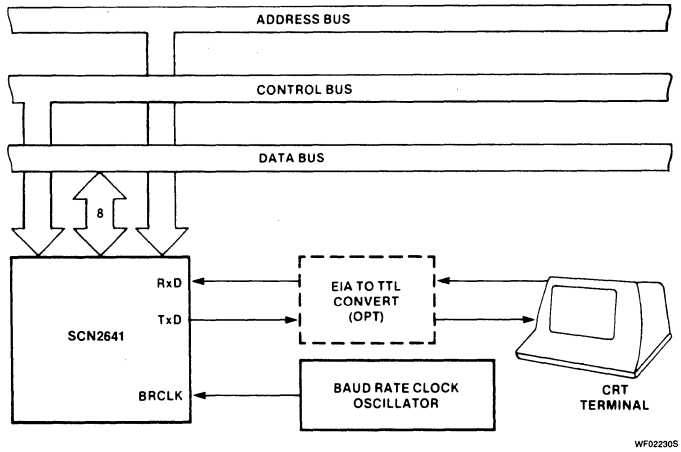
BREAK DETECTION TIMING



*If the stop bit is present, the start bit search will commence immediately.

TYPICAL APPLICATIONS

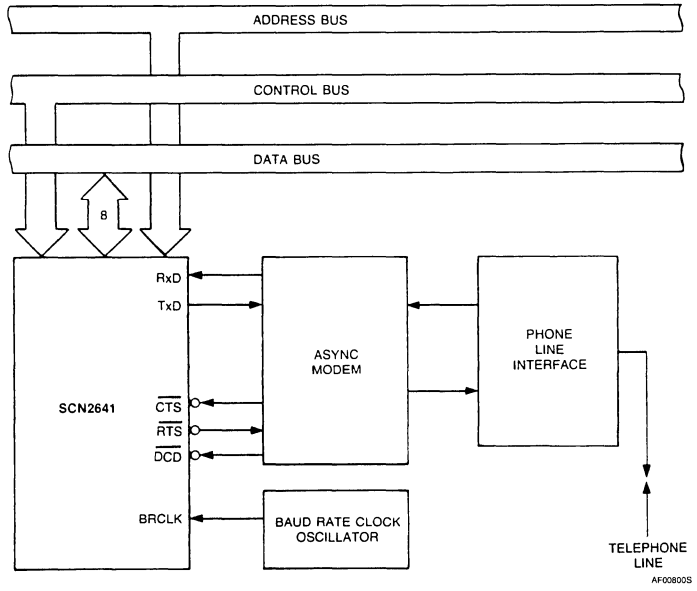
ASYNCHRONOUS INTERFACE TO CRT TERMINAL



2

TYPICAL APPLICATIONS (Continued)

ASYNCHRONOUS INTERFACE TO TELEPHONE LINES



Signetics

SCN2651 Programmable Communications Interface (PCI)

Microprocessor Products

Product Specification

DESCRIPTION

The Signetics SCN2651 PCI is a universal synchronous/asynchronous data communications controller chip designed for microcomputer systems. It interfaces directly to the Signetics SCN2650 microprocessor and may be used in a polled or interrupt driven system environment. The SCN2651 accepts programmed instructions from the microprocessor and supports many serial data communication disciplines, synchronous and asynchronous, in the full or half-duplex mode.

The PCI serializes parallel data characters received from the microprocessor for transmission. Simultaneously, it can receive serial data and convert it into parallel data characters for input to the microcomputer.

The SCN2651 contains a baud rate generator which can be programmed to either accept an external clock or to generate internal transmit or receive clocks. Sixteen different baud rates can be selected under program control when operating in the internal clock mode.

The PCI is constructed using Signetics n-channel silicon gate depletion load technology and is packaged in a 28-pin DIP.

FEATURES

- **Synchronous operation**
  - **5- to 8-bit characters**
  - **Single or double SYN operation**
  - **Internal character synchronization**
  - **Transparent or non-transparent mode**
  - **Automatic SYN or DLE-SYN insertion**
  - **SYN or DLE stripping**
  - **Odd, even, or no parity**
  - **Local or remote maintenance loopback mode**
  - **Baud rate: dc to 1M bps (1X clock)**
- **Asynchronous operation**
  - **5- to 8-bit characters**
  - **1, 1 1/2 or 2 stop bits**
  - **Odd, even, or no parity**
  - **Parity, overrun and framing error detection**
  - **Line break detection and generation**
  - **False start bit detection**
  - **Automatic serial echo mode**
  - **Local or remote maintenance loopback mode**
  - **Baud rate: dc to 1M bps (1X clock) dc to 62.5K bps (16X clock) dc to 15.625K bps (64X clock)**

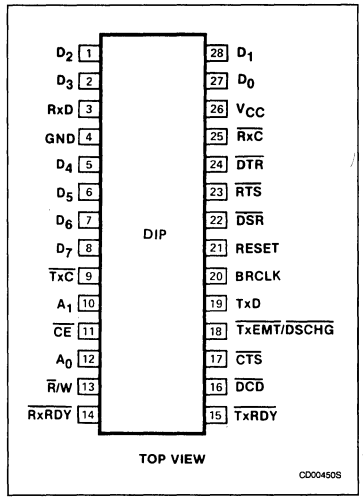
OTHER FEATURES

- **Internal or external baud rate clock**
- **16 Internal rates - 50 to 19,200 baud**
- **Double buffered transmitter and receiver**
- **Full or half duplex operation**
- **TTL compatible inputs and outputs**
- **Single 5V power supply**
- **No system clock required**
- **28-pin dual in-line package**

APPLICATIONS

- **Intelligent terminals**
- **Network processors**
- **Front-end processors**
- **Remote data concentrators**
- **Computer to computer links**
- **Serial peripherals**

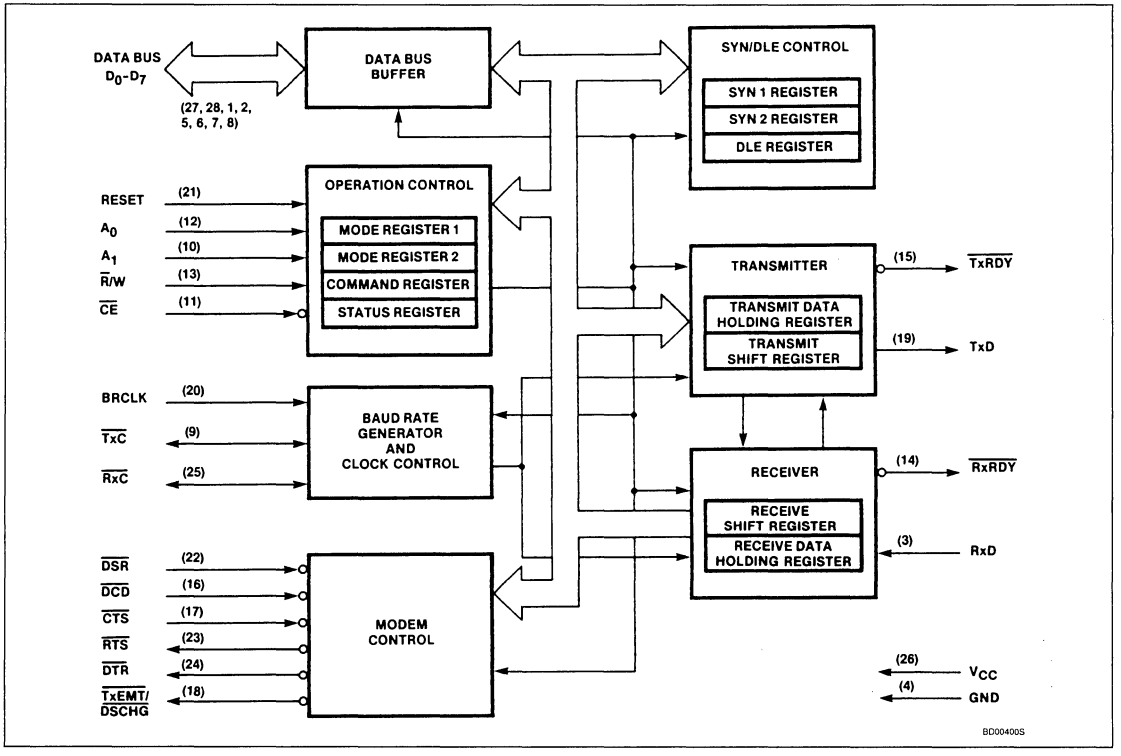
PIN CONFIGURATION



ORDERING CODE

PACKAGES	V_{CC} = 5V± 5%		
	Commercial	Automotive	Military
	0°C to +70°C	−40°C to + 85°C	−55°C to +125°C
Ceramic DIP	SCN2651CC1I28	SCN2651CA1I28	SCN2651CM1I28
Plastic DIP	SCN2651CC1N28	Contact Factory	Not available

BLOCK DIAGRAM



PIN DESCRIPTION

PIN NO.	SYMBOL	NAME AND FUNCTION	TYPE
27, 28, 1, 2, 5-8	$D_0 - D_7$	8-bit data bus	I/O
21	RESET	Reset	I
12,10	$A_0 - A_1$	Internal register select lines	I
13	$\overline{R}/W$	Read or write command	I
11	$\overline{CE}$	Chip enable input	I
22	$\overline{DSR}$	Data set ready	I
24	$\overline{DTR}$	Data terminal ready	O
23	$\overline{RTS}$	Request to send	O
17	$\overline{CTS}$	Clear to send	I
16	$\overline{DCD}$	Data carrier detected	I
18	$\overline{TxEMT/DSCHG}$	Transmitter empty or data set change	O
9	$\overline{TxC}$	Transmitter clock	I/O
25	$\overline{RxC}$	Receiver clock	I/O
19	TxD	Transmitter data	O
3	RxD	Receiver data	I
15	$\overline{TxRDY}$	Transmitter ready	O
14	$\overline{RxRDY}$	Receiver ready	O
20	BRCLK	Baud rate generator clock	I
26	V_{CC}	+5V supply	I
4	GND	Ground	I

Table 1. BAUD RATE GENERATOR CHARACTERISTICS CRYSTAL FREQUENCY = 5.0688MHz

BAUD RATE	THEORETICAL FREQUENCY 16X CLOCK	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
50	0.8KHz	0.8KHz	—	6336
75	1.2	1.2	—	4224
110	1.76	1.76	—	2880
134.5	2.152	2.1523	0.016	2355
150	2.4	2.4	—	2112
300	4.8	4.8	—	1056
600	9.6	9.6	—	528
1200	19.2	19.2	—	264
1800	28.8	28.8	—	176
2000	32.0	32.081	0.253	158
2400	38.4	38.4	—	132
3600	57.6	57.6	—	88
4800	76.8	76.8	—	66
7200	115.2	115.2	—	44
9600	153.6	153.6	—	33
19200*	307.2	316.8	3.125	16

NOTE:
*Error at 19200 can be reduced to zero by using crystal frequency 4.9152MHz.
16X clock is used in asynchronous mode. In synchronous mode, clock multiplier is 1X.

BLOCK DIAGRAM

The PCI consists of six major sections. These are the transmitter, receiver, timing, operation control, modem control and SYN/DLE control. These sections communicate with each other via an internal data bus and an internal control bus. The internal data bus interfaces to the microprocessor data bus via a data bus buffer.

Operation Control

This functional block stores configuration and operation commands from the CPU and generates appropriate signals to various internal sections to control the overall device operation. It contains read and write circuits to permit communications with the microprocessor via the data bus and contains mode registers 1 and 2, the command register, and the status register. Details of register addressing and protocol are presented in the PCI programming section of this data sheet.

Timing

The PCI contains a baud rate generator (BRG) which is programmable to accept external transmit or receive clocks or to divide an external clock to perform data communications. The unit can generate 16 commonly used baud rates, any one of which can be selected for full duplex operation. See table 1.

Receiver

The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU.

Transmitter

The transmitter accepts parallel data from the CPU, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin.

Modem Control

The modem control section provides interfacing for three input signals and three output signals used for "handshaking" and status indication between the CPU and a modem.

SYN/DLE Control

This section contains control circuitry and three 8-bit registers storing the SYN1, SYN2, and DLE characters provided by the CPU. These registers are used in the synchronous mode of operation to provide the characters required for synchronization, idle fill and data transparency.

INTERFACE SIGNALS

The PCI interface signals can be grouped into two types: the CPU-related signals (shown in table 2), which interface the SCN2651 to the microprocessor system, and the device-related signals (shown in table 3), which are used to interface to the communications device or system.

Table 2. CPU-RELATED SIGNALS

PIN NAME	PIN NO.	INPUT/OUTPUT	FUNCTION
V_{CC}	26	I	+5V supply input
GND	4	I	Ground
RESET	21	I	A high on this input performs a master reset on the SCN2651. This signal asynchronously terminates any device activity and clears the mode, command and status registers. The device assumes the idle state and remains there until initialized with the appropriate control words.
$A_1 - A_0$	10, 12	I	Address lines used to select internal PCI registers.
$\overline{R}/W$	13	I	Read command when low, write command when high.
$\overline{CE}$	11	I	Chip enable command. When low, indicates that control and data lines to the PCI are valid and that the operation specified by the $\overline{R}/W$, A_1 and A_0 inputs should be performed. When high, places the $D_0 - D_7$ lines in the tri-state condition.
$D_7 - D_0$	8, 7, 6, 5, 2, 1, 28, 27	I/O	8-bit, three-state data bus used to transfer commands, data and status between PCI and the CPU. D_0 is the least significant bit; D_7 the most significant bit.
$\overline{TxRDY}$	15	O	This output is the complement of status register bit SR0. When low, it indicates that the transmit data holding register (THR) is ready to accept a data character from the CPU. It goes high when the data character is loaded. This output is valid only when the transmitter is enabled. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{RxRDY}$	14	O	This output is the complement of status register bit SR1. When low, it indicates that the receive data holding register (RHR) has a character ready for input to the CPU. It goes high when the RHR is read by the CPU, and also when the receiver is disabled. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{TxEMT}/\overline{DSCHG}$	18	O	This output is the complement of status register bit SR2. When low, it indicates that the transmitter has completed serialization of the last character loaded by the CPU, or that a change of state of the $\overline{DSR}$ or $\overline{DCD}$ inputs has occurred. This output goes high when the status register is read by the CPU, if the TxEMT condition does not exist. Otherwise, the THR must be loaded by the CPU for this line to go high. It is an open drain output which can be used as an interrupt to the CPU.

OPERATION

The functional operation of the SCN2651 is programmed by a set of control words supplied by the CPU. These control words specify items such as synchronous or asynchronous mode, baud rate, number of bits per character, etc. The programming procedure is described in the PCI programming section of this data sheet.

After programming, the PCI is ready to perform the desired communications functions. The receiver performs serial to parallel conversion of data received from a modem or equivalent device. The transmitter converts parallel data received from the CPU to a serial bit stream. These actions are accomplished within the framework specified by the control words.

Receiver

The SCN2651 is conditioned to receive data when the $\overline{DCD}$ input is low and the RxEN bit in the command register is true. In the asynchronous mode, the receiver looks for a high to low transition of the start bit on the RxD input line. If a transition is detected, the state of the RxD line is sampled again after a delay of one-half of a bit time. If RxD is now high, the search for a valid start bit is begun again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input line at one bit time intervals until the proper number of data bits, the parity bit, and the stop bit(s) have been assembled. The data is then transferred to the receive data holding register, the RxRDY bit in the status register is set, and the $\overline{RxRDY}$ output is asserted. If the character length is less than 8 bits, the high order unused bits in the holding register are set to zero. The parity error, framing error, and overrun error status bits are strobed into the status register on the positive going edge of $\overline{RxC}$ corresponding to the received character boundary. If a break condition is detected (RxD is low for the entire character as well as the stop bit[s]), only one character consisting of all zeros (with the FE status bit set) will be transferred to the holding register. The RxD input must return to a high condition before a search for the next start bit begins.

When the PCI is initialized into the synchronous mode, the receiver first enters the hunt mode on a 0 to 1 transition of RxEN (CR2). In this mode, as data is shifted into the receiver shift register a bit at a time, the contents of the register are compared to the contents of the SYN1 register. If the two are not equal, the next bit is shifted in and the comparison is repeated. When the two registers match, the hunt mode is terminated and character assembly mode begins. If single SYN operation is programmed, the SYN detect status bit is set. If double SYN operation is programmed, the first character assembled after SYN1 must be SYN2 in order for the SYN detect bit to be set. Otherwise, the PCI returns to the hunt mode. (Note that the sequence SYN1 – SYN1 – SYN2 will not achieve synchronization). When synchronization has been achieved, the PCI continues to assemble characters and transfer them to the holding register, setting the RxRDY status bit and asserting the $\overline{RxRDY}$ output each time a character is transferred. The PE and OE status bits are set as appropriate. Further receipt of the appropriate SYN sequence sets the SYN detect status bit. If the SYN stripping mode is commanded, SYN characters are not transferred to the Holding Register. Note that the SYN characters used to establish initial synchronization are not transferred to the holding register in any case.

Table 3. DEVICE-RELATED SIGNALS

PIN NAME	PIN NO.	INPUT/OUTPUT	FUNCTION
BRCLK	20	I	5.0688MHz clock input to the internal baud rate generator. Not required if external receiver and transmitter clocks are used.
$\overline{RxC}$	25	I/O	Receiver clock. If external receiver clock is programmed, this input controls the rate at which the character is to be received. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. Data is sampled on the rising edge of the clock. If internal receiver clock is programmed, this pin becomes an output at 1X the programmed baud rate.*
$\overline{TxC}$	9	I/O	Transmitter clock. If external transmitter clock is programmed, this input controls the rate at which the character is transmitted. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. The transmitted data changes on the falling edge of the clock. If internal transmitter clock is programmed, the pin becomes an output at 1X the programmed baud rate.*
RxD	3	I	Serial data input to the receiver. "Mark" is high, "Space" is low.
TxD	19	O	Serial data output from the transmitter. "Mark" is high, "space" is low. Held in mark condition when the transmitter is disabled.
$\overline{DSR}$	22	I	General purpose input which can be used for data set ready or ring indicator condition. Its complement appears as status register bit SR7. Causes a low output on $\overline{TxEMT/DSCHG}$ when its state changes.
$\overline{DCD}$	16	I	Data carrier detect input. Must be low in order for the receiver to operate. Its complement appears as status register bit SR6. Causes a low output on $\overline{TxEMT/DSCHG}$ when its state changes.
$\overline{CTS}$	17	I	Clear to send input. Must be low in order for the transmitter to operate. If it goes high during transmission, the character in the transmit shift register will be transmitted before termination.
$\overline{DTR}$	24	O	General purpose output which is the complement of command register bit CR1. Normally used to indicate data terminal ready.
$\overline{RTS}$	23	O	General purpose output which is the complement of command register bit CR5. Normally used to indicate request to send.

NOTE:
*$\overline{RxC}$ and $\overline{TxC}$ outputs have short circuit protection max. C_L = 100pf

Transmitter

The PCI is conditioned to transmit data when the $\overline{CTS}$ input is low and the TxEN command register bit is set. The SCN2651 indicates to the CPU that it can accept a character for transmission by setting the TxRDY status bit and asserting the $\overline{TxRDY}$ output. When the CPU writes a character into the transmit data holding register, these conditions are negated. Data is transferred from the holding register to the transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again. Thus, one full character time of buffering is provided.

In the asynchronous mode, the transmitter automatically sends a start bit followed by the programmed number of data bits, the least significant bit being sent first. It then appends an optional odd or even parity bit and the programmed number of stop bits. If, following transmission of the data bits, a new character is not available in the transmit holding register, the TxD output remains in the marking (high) condition and the $\overline{TxEMT/DSCHG}$ output and its corresponding status bit are asserted. Transmission resumes when the CPU loads a new character into the holding register. The transmitter can be forced to output a continuous low (BREAK) condition by setting the send break command bit high.

In the synchronous mode, when the SCN2651 is initially conditioned to transmit, the TxD output remains high and the TxRDY condition is asserted until the first character to be transmitted (usually a SYN character) is loaded by the CPU. Subsequent to this, a continuous stream of characters is transmitted. No extra bits (other than parity, if commanded) are generated by the PCI unless the CPU fails to send a new character to the PCI by the time the transmitter has completed sending the previous character.

Since synchronous communication does not allow gaps between characters, the PCI asserts TxEMT and automatically "fills" the gap by transmitting SYN1s, SYN1 – SYN2 doublets, or DLE – SYN1 doublets, depending on the state of MR16 and MR17. Normal transmission of the message resumes when a new character is available in the transmit data holding register. If the send DLE bit in the command register is true, the DLE character is automatically transmitted prior to transmission of the message character in THR.

PCI PROGRAMMING

Prior to initiating data communications, the SCN2651 operational mode must be programmed by performing write operations to the mode and command registers. In addition, if synchronous operation is programmed, the appropriate SYN/DLE registers must be loaded. The PCI can be reconfigured at any time during program execution. However, if the change has an effect on the reception of a character the receiver should be disabled. Alternatively if the change is made 1½ RxC periods after RxRDY goes active it will affect the next character assembly. A flowchart of the initialization process appears in figure 1.

The internal registers of the PCI are accessed by applying specific signals to the $\overline{CE}$, $\overline{R}/W$, A_1 and A_0 inputs. The conditions necessary to address each register are shown in table 4.

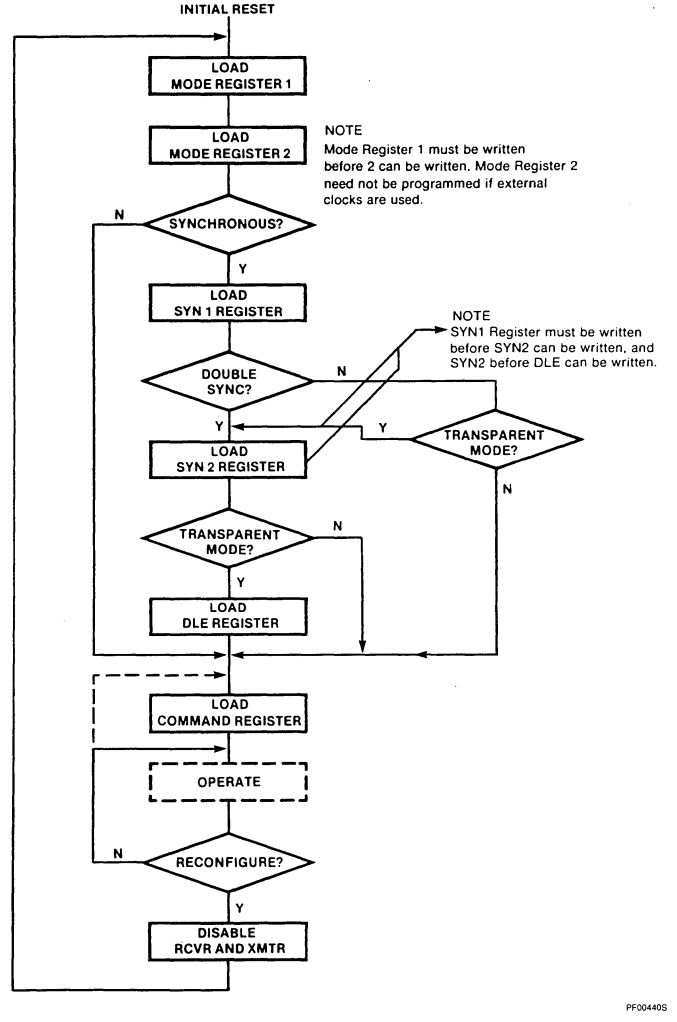


Figure 1. SCN2651 Initialization Flowchart

Table 4. SCN2651 REGISTER ADDRESSING

$\overline{CE}$	A_1	A_0	$\overline{R}/W$	FUNCTION
1	X	X	X	Tri-state data bus
0	0	0	0	Read receive holding register
0	0	0	1	Write transmit holding register
0	0	1	0	Read status register
0	0	1	1	Write SYN1/SYN2/DLE registers
0	1	0	0	Read mode registers ½
0	1	0	1	Write mode registers ½
0	1	1	0	Read command register
0	1	1	1	Write command register

NOTE:
See AC Characteristics section for timing requirements.

The SYN1, SYN2, and DLE registers are accessed by performing write operations with the conditions $A_1 = 0$, $A_0 = 1$, and $\overline{R}/W = 1$. The first operation loads the SYN1 register. The next loads the SYN2 register, and the third loads the DLE register. Reading or loading the mode registers is done in a similar manner. The first write (or read) operation addresses mode register 1, and a subsequent operation addresses mode register 2. If more than the required number of accesses are made, the internal sequencer recycles to point at the first register. The pointers are reset to SYN1 register and mode register 1 by a RESET input or by performing a "read command register" operation, but are unaffected by any other read or write operation.

The SCN2651 register formats are summarized in tables 5, 6, 7 and 8. Mode registers 1 and 2 define the general operational characteristics of the PCI, while the command register controls the operation within this basic framework. The PCI indicates its status in the status register. These registers are cleared when a RESET input is applied.

Mode Register 1 (MR1)

Table 5 illustrates mode register 1. Bits MR11 and MR10 select the communication format and baud rate multiplier. 00 specifies synchronous mode and 1X multiplier. 1X, 16X, and 64X multipliers are programmable for asynchronous format. However, the multiplier in asynchronous format applies only if the external clock input option is selected by MR24 or MR25.

MR13 and MR12 select a character length of 5, 6, 7, or 8 bits. The character length does not include the parity bit, if programmed, and does not include the start and stop bits in asynchronous mode.

MR14 controls parity generation. If enabled, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR15 selects odd or even parity when parity is enabled by MR14.

In asynchronous mode, MR17 and MR16 select character framing of 1, 1.5, or 2 stop bits. (If 1X baud rate is programmed, 1.5 stop bits default to 1 stop bit on transmit). In synchronous mode, MR17 controls the number of SYN characters used to establish synchronization and for character fill when the transmitter is idle. SYN1 alone is used if MR17 = 1, and SYN1 – SYN2 is used when MR17 = 0. If the transparent mode is specified by MR16, DLE – SYN1 is used for character fill and SYN detect, but the normal synchronization sequence is used. Also DLE stripping and DLE detect (with MR14 = 0) are enabled.

Mode Register 2 (MR2)

Table 6 illustrates mode register 2. MR23, MR22, MR21, and MR20 control the frequency of the internal baud rate generator (BRG). Sixteen rates are selectable. When driven by a 5.0688MHz input at the BRCLK input (pin 20), the BRG output has zero error except at 134.5, 2000, and 19,200 baud, which have errors of +0.016%, +0.235%, and +3.125% respectively.

MR25 and MR24 select either the BRG or the external inputs $\overline{TxC}$ and $\overline{RxC}$ as the clock source for the transmitter and receiver, respectively. If the BRG clock is selected, the baud rate factor in asynchronous mode is 16X regardless of the factor selected by MR11 and MR10. In addition, the corresponding clock pin provides an output at 1X the baud rate.

Table 5. MODE REGISTER 1 (MR1)

MR17	MR16	MR15	MR14	MR13	MR12	MR11	MR10
		Parity Type	**Parity Control**	**Character Length**		**Mode and Baud Rate Factor**	
Asynch: Stop bit length 00 = Invalid 01 = 1 Stop bit 10 = 1½ Stop bits 11 = 2 Stop bits		0 = Odd 1 = Even	0 = Disabled 1 = Enabled	00 = 5 Bits 01 = 6 Bits 10 = 7 Bits 11 = 8 Bits		00 = Synchronous 1X rate 01 = Asynchronous 1X rate 10 = Asynchronous 16X rate 11 = Asynchronous 64X rate	
Synch: Number of syn char 0 = Double SYN 1 = Single SYN	**Synch: Transparency control** 0 = Normal 1 = Transparent						

NOTE:
Baud rate factor in asynchronous applies only if external clock is selected. Factor is 16X if internal clock is selected. Mode must be selected (MR11, MR10) in any case.

Table 6. MODE REGISTER 2 (MR2)

MR27	MR26	MR25	MR24	MR23	MR22	MR21	MR20
		Transmitter Clock	**Receiver Clock**	**Baud Rate Selection**			
Not used		0 = External 1 = Internal	0 = External 1 = Internal	0000 = 50 Baud 0001 = 75 0010 = 110 0011 = 134.5 0100 = 150 0101 = 300 0110 = 600 0111 = 1200		1000 = 1800 Baud 1001 = 2000 1010 = 2400 1011 = 3600 1100 = 4800 1101 = 7200 1110 = 9600 1111 = 19,200	

Table 7. COMMAND REGISTER (CR)

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Operating Mode		**Request to Send**	**Reset Error**		**Receive Control (RxEN)**	**Data Terminal Ready**	**Transmit Control (TxEN)**
00 = Normal operation 01 = Asynch: automatic echo mode Synch: SYN and/or DLE stripping mode 10 = Local Loopback 11 = Remote Loopback		0 = Force $\overline{RTS}$ output high 1 = Force $\overline{RTS}$ output low	0 = Normal 1 = Reset error flag in status reg (FE, OE, PE/ DLE DETECT)	**Asynch: Force break** 0 = Normal 1 = Force break **Synch Send DLE** 0 = Normal 1 = Send DLE	0 = Disable 1 = Enable	0 = Force $\overline{DTR}$ output high 1 = Force $\overline{DTR}$ output low	0 = Disable 1 = Enable

Table 8. STATUS REGISTER (SR)

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
Data Set Ready	**Data Carrier Detect**	**FE/SYN Detect**	**Overrun**	**PE/DLE Detect**	**TxEMT/DSCHG**	**RxRDY**	**TxRDY**
0 = $\overline{DSR}$ Input is high 1 = $\overline{DSR}$ Input is low	0 = $\overline{DCD}$ Input is high 1 = $\overline{DCD}$ Input is low	**Asynch:** 0 = Normal 1 = Framing ERROR **Synch:** 0 = Normal 1 = SYN char detected	0 = Normal 1 = Overrun error	**Asynch:** 0 = Normal 1 = Parity error **Synch:** 0 = Normal 1 = Parity error or DLE char received	0 = Normal 1 = Change in $\overline{DSR}$ or $\overline{DCD}$, or transmit shift register is empty	0 = Receive holding reg empty 1 = Receive holding reg has data	0 = Transmit holding reg busy 1 = Transmit holding reg empty

2

Command Register (CR)

Table 7 illustrates command register. Bits CR0 (TxEN) and CR2 (RxEN) enable or disable the transmitter and receiver respectively. A 0 to 1 transition of CR2 forces start bit search (async mode) or hunt mode (sync mode) on the second $\overline{RxC}$ rising edge. Disabling the receiver causes $\overline{RxRDY}$ to go high (inactive). If the transmitter is disabled, it will complete the transmission of the character in the transmit shift register (if any) prior to terminating operation. The TxD output will then remain in the marking state (high) while the $\overline{TxRDY}$ and $\overline{TxEMT}$ will go high (inactive). If the receiver is disabled, it will terminate operation immediately. Any character being assembled will be neglected.

Bits CR1 (DTR) and CR5 (RTS) control the $\overline{DTR}$ and $\overline{RTS}$ outputs. Data at the outputs is the logical complement of the register data.

In asynchronous mode, setting CR3 will force and hold the TxD output low (spacing condition) at the end of the current transmitted character. Normal operation resumes when CR3 is cleared. The TxD line will go high for at least one bit time before beginning transmission of the next character in the transmit data holding register. In synchronous mode, setting CR3 causes the transmission of the DLE register contents prior to sending the character in the transmit data holding register. CR3 should be reset in response to the next TxRDY.

Setting CR4 causes the error flags in the status register (SR3, SR4, and SR5) to be cleared. This is a one time command. There is no internal latch for this bit.

The PCI can operate in one of four submodes within each major mode (synchronous or asynchronous). The operational submode is determined by CR7 and CR6. CR7 – CR6 = 00 is the normal mode, with the transmitter and receiver operating independently in accordance with the mode and status register instructions.

In asynchronous mode, CR7 – CR6 = 01 places the PCI in the automatic echo mode. Clocked, regenerated received data is automatically directed to the TxD line while normal receiver operation continues. The receiver must be enabled (CR2 = 1), but the transmitter need not be enabled. CPU to receiver communications continues normally, but the CPU to transmitter link is disabled. Only the first character of a break condition is echoed. The TxD output will go high until the next valid start is detected.

The following conditions are true while in automatic echo mode:

1. Data assembled by the receiver is automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. $\overline{TxRDY}$ output = 1.
4. The $\overline{TxEMT/DSCHG}$ pin will reflect only the data set change condition.
5. The TxEN command (CR0) is ignored.

In synchronous mode, CR7 – CR6 = 01 places the PCI in the automatic SYN/DLE stripping mode. The exact action taken depends on the setting of bits MR17 and MR16:

1. In the non-transparent, single SYN mode (MR17 – MR16 = 10), characters in the data stream matching SYN1 are not transferred to the receive data holding register (RHR).
2. In the non-transparent, double SYN mode (MR17 – MR16 = 00), characters in the data stream matching SYN1, or SYN2 if immediately preceded by SYN1, are not transferred to the RHR. However, only the first SYN1 of an SYN1 – SYN1 pair is stripped.
3. In transparent mode (MR16 = 1), characters in the data stream matching DLE, or SYN1 if immediately preceded by DLE, are not transferred to the RHR. However, only the first DLE of a DLE – DLE pair is stripped.

Note that automatic stripping mode does not affect the setting of the DLE detect and SYN detect status bits (SR3 and SR5).

Two diagnostic submodes can also be configured. In local loopback mode (CR7 – CR6 = 10), the following loops are connected internally:

1. The transmitter output is connected to the receiver input.
2. $\overline{DTR}$ is connected to $\overline{DCD}$ and $\overline{RTS}$ is connected to $\overline{CTS}$.
3. The receiver is clocked by the transmit clock.
4. The $\overline{DTR}$, $\overline{RTS}$ and $\overline{TxD}$ outputs are held high.
5. The $\overline{CTS}$, $\overline{DCD}$, $\overline{DSR}$ and RxD inputs are ignored.

Additional requirements to operate in the local loopback mode are that CR0 (TxEN), CR1 (DTR), and CR5 (RTS) must be set to 1. CR2 (RxEN) is ignored by the PCI.

The second diagnostic mode is the remote loopback mode (CR7 – CR6 = 11). In this mode:

1. Data assembled by the receiver is automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. No data is sent to the local CPU, but the error status conditions (PE, OE, FE) are set.
4. The $\overline{RxRDY}$, $\overline{TxRDY}$, and $\overline{TxEMT/DSCHG}$ outputs are held high.
5. CR0 (TxEN) is ignored.
6. All other signals operate normally.

Status Register

The data contained in the status register (as shown in table 8) indicate receiver and transmitter conditions and modem/data set status.

SR0 is the transmitter ready (TxRDY) status bit. It, and its corresponding output, are valid only when the transmitter is enabled. If equal to 0, it indicates that the transmit data holding register has been loaded by the CPU and the data has not been transferred to the transmit shift register. If set equal to 1, it indicates that the Holding Register is ready to accept data from the CPU. This bit is initially set when the transmitter is enabled by CR0, unless a character has previously been loaded into the holding register. It is not set when the auto-

matic echo or remote loopback modes are programmed. When this bit is set, the $\overline{TxRDY}$ output pin is low. In the automatic echo and remote loopback modes, the output is held high.

SR1, the receiver ready (RxRDY) status bit, indicates the condition of the receive data holding register. If set, it indicates that a character has been loaded into the holding register from the receive shift register and is ready to be read by the CPU. If equal to zero, there is no new character in the holding register. This bit is cleared when the CPU reads the receive data holding register or when the receiver is disabled by CR2. When set, the $\overline{RxRDY}$ output is low.

The TxEMT/DSCHG bit, SR2, when set, indicates either a change of state of the $\overline{DSR}$ or $\overline{DCD}$ inputs or that the transmit shift register has completed transmission of a character and no new character has been loaded into the transmit data holding register. Note that in synchronous mode this bit will be set even though the appropriate "fill" character is transmitted. TxEMT will not go active until at least one character has been transmitted. It is cleared by loading the transmit data holding register. The DSCHG condition is enabled when TxEN = 1 or RxEN = 1. If the status register is read twice and SR2 = 1 while SR6 and SR7 remain unchanged, then a TxEMT condition exists. It is cleared when the status register is read by the CPU. When SR2 is set, the TxEMT/DSCHG output is low.

SR3, when set, indicates a received parity error when parity is enabled by MR14. In synchronous transparent mode (MR16 = 1), with parity disabled, it indicates that a character matching the DLE register has been received. However, only the first DLE of two successive DLEs will set SR3. This bit is cleared when the receiver is disabled and by the reset error command, CR4.

The overrun error status bit, SR4, indicates that the previous character loaded into the receive holding register was not read by the CPU at the time a new received character was transferred into it. This bit is cleared when the receiver is disabled and by the reset error command, CR4.

In asynchronous mode, bit SR5 signifies that the received character was not framed by the programmed number of stop bits. (If 1.5 stop bits are programmed, only the first stop bit is checked.) If RHR = 0 when SR5 = 1 a break condition is present. In synchronous non-transparent mode (MR16 = 0), it indicates receipt of the SYN1 character is single SYN mode or the SYN1 – SYN2 pair in double SYN mode. In synchronous transparent mode (MR16 = 1), this bit is set upon detection of the initial synchronizing characters (SYN1 or SYN1 – SYN2) and, after synchronization has been achieved, when a DLE – SYN1 pair is received. The bit is reset when the receiver is disabled, when the reset error command is given in asynchronous mode, and when the status register is read by the CPU in the synchronous mode.

SR6 and SR7 reflect the conditions of the $\overline{DCD}$ and $\overline{DSR}$ inputs respectively. A low input sets its corresponding status bit and a high input clears it.

2

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	Note 4	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Input voltage						V
V_{IL}	Low				0.8	
V_{IH}	High		2.0			
Output voltage						V
V_{OL}	Low	I_{OL} = 1.6mA			0.4	
V_{OH}	High	I_{OH} = −100μA	2.4			
I_{IL}	Input leakage current	V_{IN} = 0 to 5.25V	−10		10	μA
Tristate output leakage current						μA
I_{LH}	Data bus high	V_O = 4.0V	−10		10	
I_{LL}	Data bus low	V_O = 0.45V	−10		10	
I_{CC}	Power supply current				150	mA

CAPACITANCE T_A = 25°C, V_{CC} = 0V

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Capacitance						pF
C_{IN}	Input	fc = 1MHz			20	
C_{OUT}	Output	Unmeasured pins tied			20	
$C_{I/O}$	Input/Output	to ground			20	

AC ELECTRICAL CHARACTERISTICS[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Pulse width						ns
t_{RES}	Reset		1000			
t_{CE}	Chip enable		300			
Set-up and hold time						ns
t_{AS}	Address set-up		20			
t_{AH}	Address hold		20			
t_{CS}	$\overline{R}$/W control set-up		20			
t_{CH}	$\overline{R}$/W control hold		20			
t_{DS}	Data set-up for write		225			
t_{DH}	Data hold for write		0			
t_{RXS}	Rx data set-up		300			
t_{RXH}	Rx data hold		350			
t_{DD}	Data delay time for read	C_L = 100pF			250	ns
t_{DF}	Data bus floating time for read	C_L = 100pF			150	ns
t_{CED}	$\overline{CE}$ to $\overline{CE}$ delay		700			ns
Input clock frequency						MHz
f_{BRG}	Baud rate generator		1.0	5.0688	5.0738	
$f_{R/T}$[10]	$\overline{TxC}$ or $\overline{RxC}$		dc		1.0	
Clock width						ns
t_{BRH}[9]	Baud rate high		70			
t_{BRL}[9]	Baud rate low		70			
$t_{R/TH}$	$\overline{TxC}$ or $\overline{RxC}$ high		500			
$t_{R/TL}$[10]	$\overline{TxC}$ or $\overline{RxC}$ low		500			
t_{TXD}	TxD delay from falling edge of $\overline{TxC}$	C_L = 100pF			650	ns
t_{TCS}	Skew between TxD changing and falling edge of $\overline{TxC}$ output[8]	C_L = 100pF		0		ns

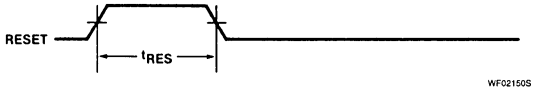
NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified. See ordering code table for applicable temperature range and operating supply range.
5. All voltage measurements are referenced to ground. All time measurements are at the 50% level for inputs (except t_{BRH} and T_{BRL}) and at 0.8V and 2.0V for outputs. Input levels for testing 0.45V and 2.4V.
6. Typical values are at +25°C, typical supply voltages and typical processing parameters.
7. TxRDY, RxRDY and TxEMT/DSCHG outputs are open drain.
8. Parameter applies when internal transmitter clock is used.
9. Under test conditions of 5.0688MHz, f_{BRG}, t_{BRH}, and t_{BRL} measured at V_{IH} and V_{IL} respectively.
10. $t_{R/T}$ and $t_{R/TL}$ shown for all modes except local loopback. For local loopback mode $f_{R/T}$ = 0.7MHz and $t_{R/TL}$ = 700ns min.

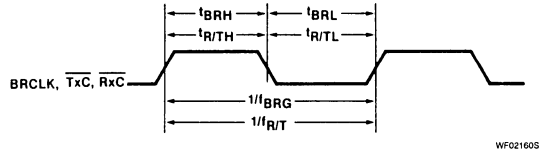
2

TIMING DIAGRAMS

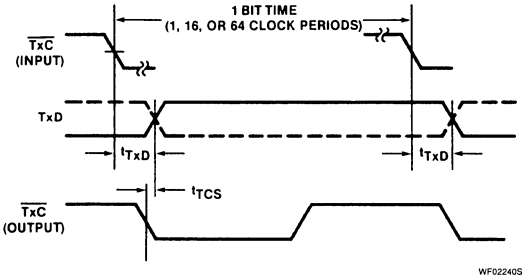
RESET



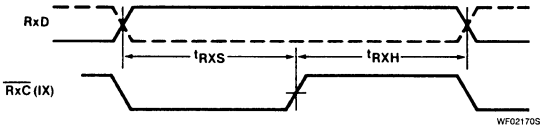
CLOCK



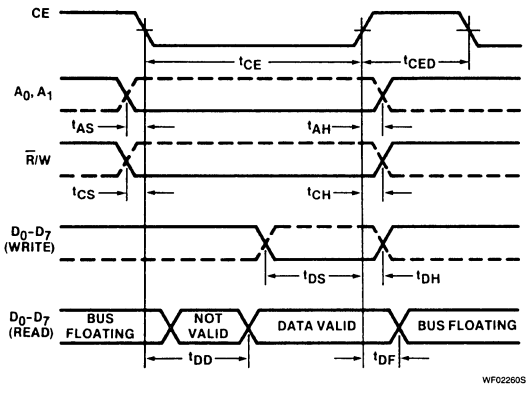
TRANSMIT



RECEIVE

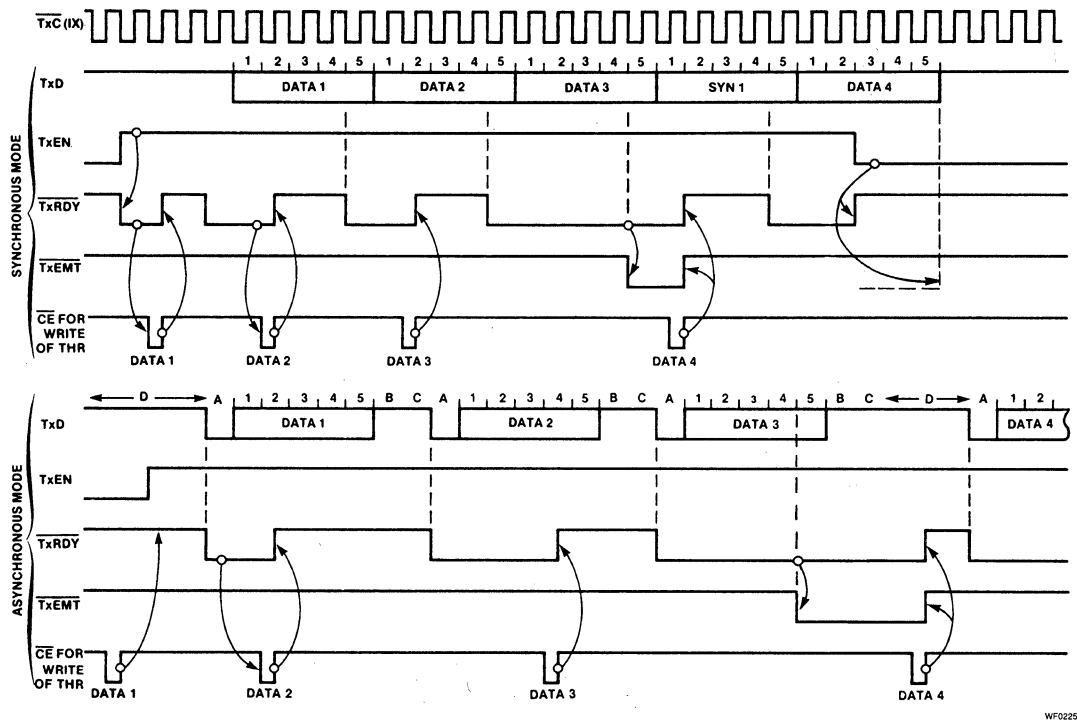


READ AND WRITE



TIMING DIAGRAMS (Continued)

TxRDY, TxEMT (Shown for 5-bit characters, no parity, 2 stop bits [in asynchronous mode])

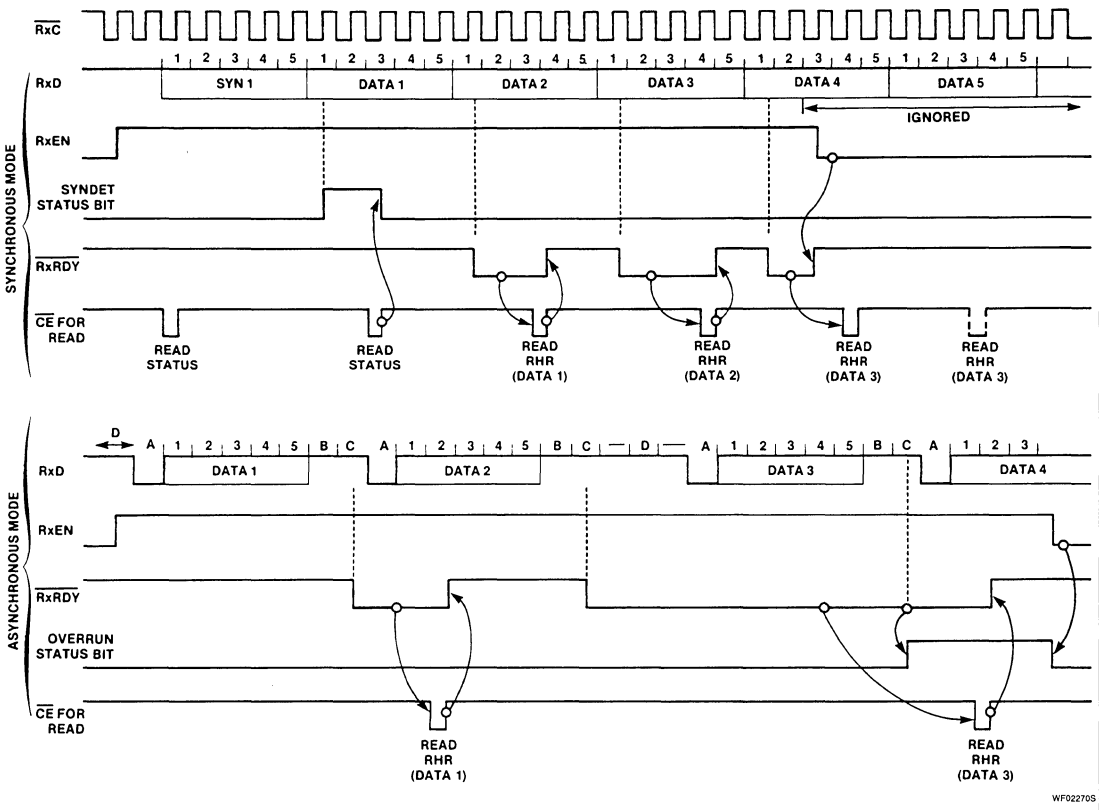


NOTES:
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
TxEMT goes low at the beginning of the last data bit, or, if parity is enabled, at the beginning of the parity bit.

2

TIMING DIAGRAMS (Continued)

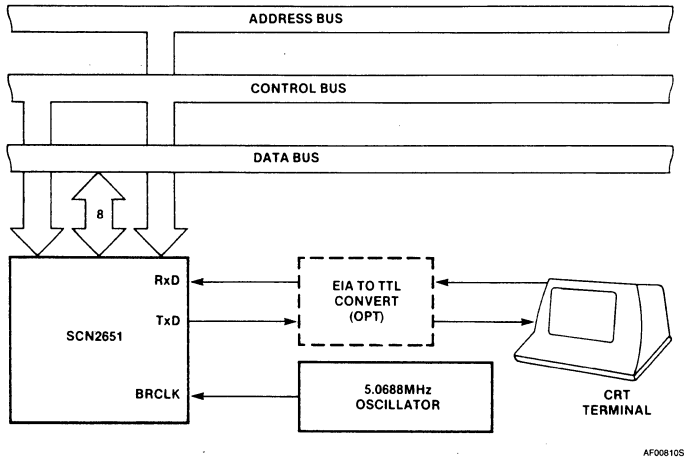
RxRDY (Shown for 5-bit characters, no parity, 2 stop bits [in asynchronous mode])



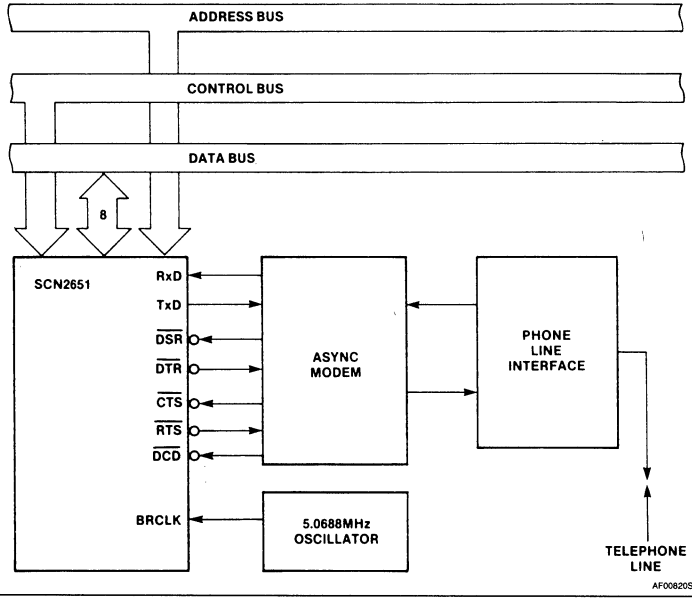
NOTES:
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition

TYPICAL APPLICATIONS

ASYNCHRONOUS INTERFACE TO CRT TERMINAL

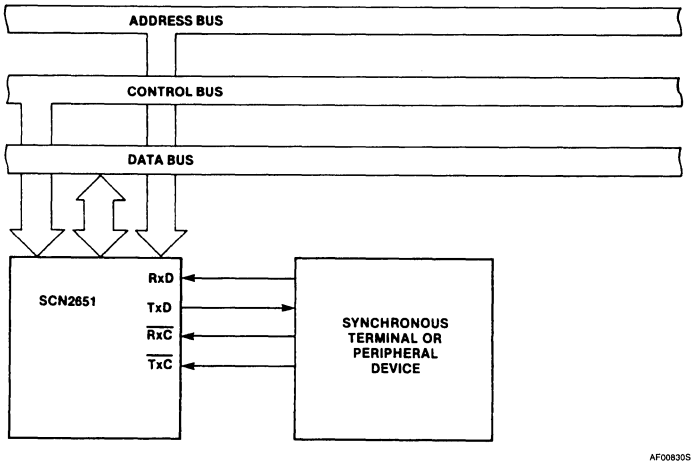


ASYNCHRONOUS INTERFACE TO TELEPHONE LINES

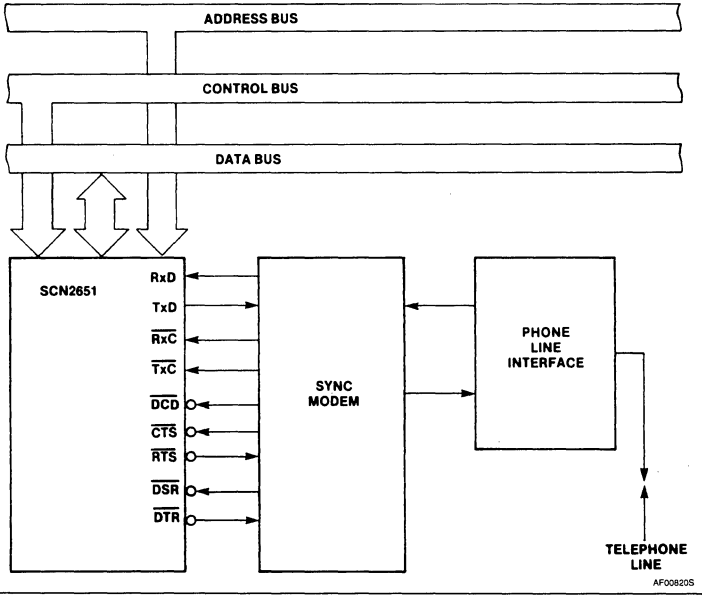


TYPICAL APPLICATIONS (Continued)

SYNCHRONOUS INTERFACE TO TERMINAL OR PERIPHERAL DEVICE



SYNCHRONOUS INTERFACE TO TELEPHONE LINES



SCN2652/SCN68652
Multi-Protocol Communications Controller (MPCC)

Product Specification

Microprocessor Products

DESCRIPTION

The SCN2652/68652 Multi-Protocol Communications Controller (MPCC) is a monolithic n-channel MOS LSI circuit that formats, transmits and receives synchronous serial data while supporting bit-oriented or byte control protocols. The chip is TTL compatible, operates from a single +5V supply, and can interface to a processor with an 8 or 16-bit bidirectional data bus.

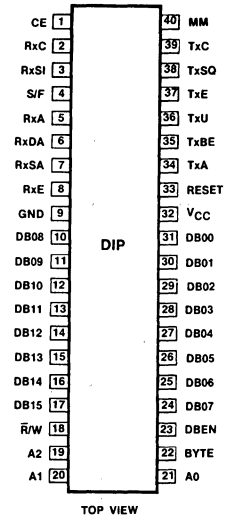
FEATURES

- **DC to 1Mbps or 2Mbps data rate**
- **Bit-oriented protocols (BOP): SDLC, ADCCP, HDLC**
- **Byte-control protocols (BCP): DDCMP, BISYNC (external CRC)**
- **Programmable operation**
  - **8 or 16-bit tri-state data bus**
  - **Error control – CRC or VRC or none**
  - **Character length – 1 to 8 bits for BOP or 5 to 8 bits for BCP**
  - **SYNC or secondary station address comparison for BCP-BOP**
  - **Idle transmission of SYNC/FLAG or MARK for BCP-BOP**
- **Automatic detection and generation of special BOP control sequences, i.e., FLAG, ABORT, GA**
- **Zero insertion and deletion for BOP**
- **Short character detection for last BOP data character**
- **SYNC generation, detection, and stripping for BCP**
- **Maintenance mode for self-testing**
- **TTL compatible**
- **Single +5V supply**

APPLICATIONS

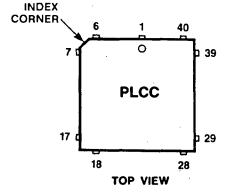
- **Intelligent terminals**
- **Line controllers**
- **Network processors**
- **Front end communications**
- **Remote data concentrators**
- **Communication test equipment**
- **Computer to computer links**

PIN CONFIGURATION



TOP VIEW

CD00160S



TOP VIEW

CD0044PS

Pin	Function	Pin	Function
1	NC	23	NC
2	CE	24	A0
3	RxC	25	BYTE
4	RxSI	26	DBEN
5	S/F	27	DB07
6	RxA	28	DB06
7	RxDA	29	DB05
8	RxSA	30	DB04
9	RxE	31	DB03
10	GND	32	DB02
12	NC	33	DB01
11	DB08	34	NC
13	DB09	35	DB00
14	DB10	36	V_{CC}
15	DB11	37	RESET
16	DB12	38	TxA
17	DB13	39	TxBE
18	DB14	40	TxU
19	DB15	41	TxE
20	$\overline{R}$/W	42	TxSQ
21	A2	43	TxC
22	A1	44	MM

NOTE: 00 is least significant bit, highest number (that is, DB15, A2) is most significant bit.

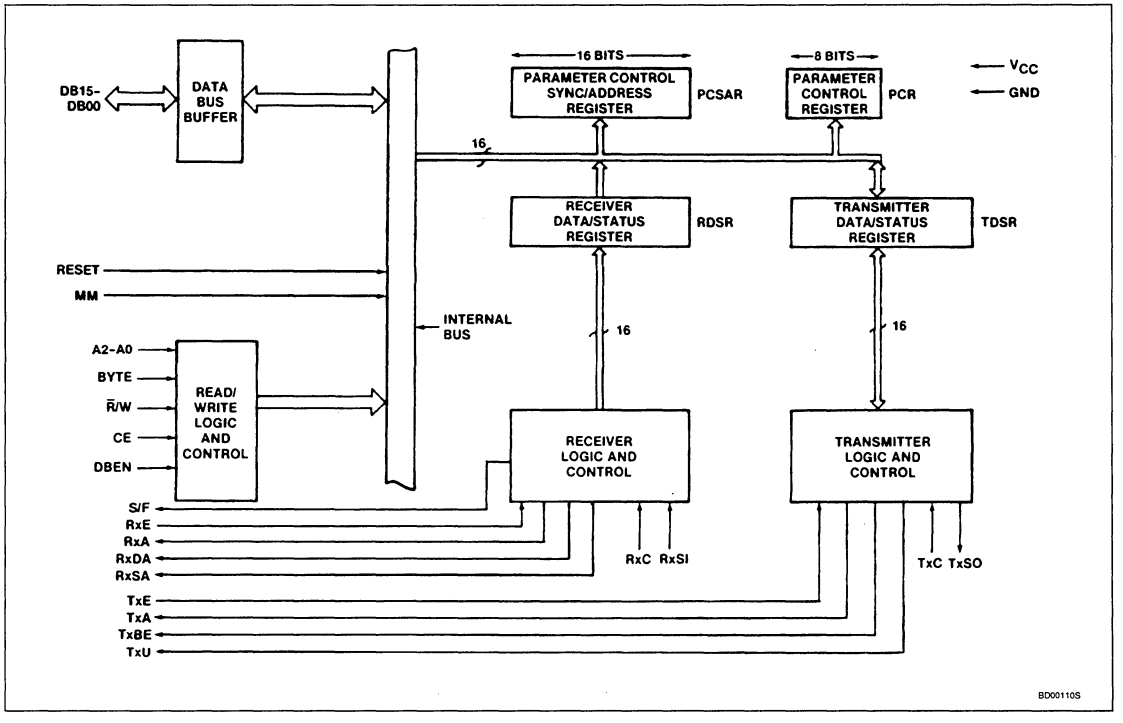
2

ORDERING CODE

PACKAGES		V_{CC} = 5V± 5%		
		Commercial	Automotive	Extended
		0°C to +70°C	−40°C to +85°C	−55°C to +125°C
Ceramic DIP	1MHz	SCN2652AC1I40	SCN2652AA1I40	SCN2652AM1I40
	2MHz	SCN2652AC2I40	SCN2652AA2I40	SCN2652AM2I40
Plastic DIP	1MHz	SCN2652AC1N40	Contact Factory	Not Available
	2MHz	SCN2652AC2N40	Contact Factory	Not Available
Plastic LCC	1MHz	SCN2652AC1A44	Contact Factory	Not Available
	2MHz	SCN2652AC2A44	Contact Factory	Not Available

NOTE:
SCN68652 is identical to SCN2652. Order using part numbers shown above.

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
DB15 – DB00	17 – 10 24 – 31	I/O	**Data Bus:** DB07 – DB00 contain bidirectional data while DB15 – DB08 contain control and status information to or from the processor. Corresponding bits of the high and low order bytes can be wire OR'ed onto an 8-bit bus. The data bus is floating if either CE or DBEN are low.
A2 – A0	19 – 21	I	**Address Bus:** A2 – A0 select internal registers. The four 16-bit registers can be addressed on a word or byte basis. See Register Address section.
BYTE	22	I	**Byte:** Single byte (8-bit) data bus transfers are specified when this input is high. A low level specifies 16-bit data bus transfers.
CE	1	I	**Chip Enable:** A high input permits a data bus operation when DBEN is activated.
$\overline{R}$/W	18	I	**Read/Write:** $\overline{R}$/W controls the direction of data bus transfer. When high, the data is to be loaded into the addressed register. A low input causes the contents of the addressed register to be presented on the data bus.
DBEN	23	I	**Data Bus Enable:** After A2 – A0, CE, BYTE and $\overline{R}$/W are set up, DBEN may be strobed. During a read, the 3-state data bus (DB) is enabled with information for the processor. During a write, the stable data is loaded into the addressed register and TxBE will be reset if TDSR was addressed.
RESET	33	I	**Reset:** A high level initializes all internal registers (to zero) and timing.
MM	40	I	**Maintenance Mode:** MM internally gates TxSO back to RxSI and TxC to RxC for off line diagnostic purposes. The RxC and RxSI inputs are disabled and TxSO is high when MM is asserted.
RxE	8	I	**Receiver Enable:** A high level input permits the processing of RxSI data. A low level disables the receiver logic and initializes all receiver registers and timing.
RxA	5	O	**Receiver Active:** RxA is asserted when the first data character of a message is ready for the processor. In the BOP mode this character is the address. The received address must match the secondary station address if the MPCC is a secondary station. In BCP mode, if strip-SYNC ($PCSAR_{13}$) is set, the first non-SYNC character is the first data character; if strip-SYNC is zero, the character following the second SYNC is the first data character. In the BOP mode, the closing FLAG resets RxA. In the BCP mode, RxA is reset by a low level at RxE.
RxDA*	6	O	**Receiver Data Available:** RxDA is asserted when an assembled character is in $RDSR_L$ and is ready to be presented to the processor. This output is reset when $RDSR_L$ is read.
RxC	2	I	**Receiver Clock:** RxC (1X) provides timing for the receiver logic. The positive going edge shifts serial data into the RxSR from RxSI.
S/F	4	O	**SYNC/FLAG:** S/F is asserted for one RxC clock time when a SYNC or FLAG character is detected.
RxSA*	7	O	**Receiver Status Available:** RxSA is asserted when there is a zero to one transition of any bit in $RDSR_H$ except for RSOM. It is cleared when $RDSR_H$ is read.
RxSI	3	I	**Receiver Serial Input:** RxSI is the received serial data. Mark = '1', space = '0'.
TxE	37	I	**Transmitter Enable:** A high level input enables the transmitter data path between $TDSR_L$ and TxSO. At the end of a message, a low level input causes TxSO = 1(mark) and TxA = 0 after the closing FLAG (BOP) or last character (BCP) is output on TxSO.
TxA	34	O	**Transmitter Active:** TxA is asserted after TSOM ($TDSR_8$) is set and TxE is raised. This output will reset when TxE is low and the closing FLAG (BOP) or last character (BCP) has been output on TxSO.
TxBE*	35	O	**Transmitter Buffer Empty:** TxBE is asserted when the TDSR is ready to be loaded with new control information or data. The processor should respond by loading the TDSR which resets TxBE.
TxU*	36	O	**Transmitter Underrun:** TxU is asserted during a transmit sequence when the service of TxBE has been delayed for one character time. This indicates the processor is not keeping up with the transmitter. Line fill depends on $PCSAR_{11}$. TxU is reset by RESET or setting of TSOM ($TDSR_8$), synchronized by the falling edge of TxC.
TxC	39	I	**Transmitter Clock:** TxC (1X) provides timing for the transmitter logic. The positive going edge shifts data out of the TxSR to TxSO.
TxSO	38	O	**Transmitter Serial Output:** TxSO is the transmitted serial data. Mark = '1', space = '0'.
V_{CC}	32	I	**+5V:** Power supply.
GND	9	I	**Ground:** 0V reference ground.

*Indicates possible interrupt signal

2

Table 1. GLOSSARY

REGISTERS		NO. OF BITS	DESCRIPTION*
Addressable			
PCSAR	Parameter control sync/ address register	16	$PCSAR_H$ and PCR contain parameters common to the receiver and transmitter. $PCSAR_L$ contains a programmable SYNC character (BCP) or secondary station address (BOP).
PCR	Parameter control register	8	
RDSR	Receive data/status register	16	$RDSR_H$ contains receiver status information. $RDSR_L$ = RxDB contains the received assembled character.
TDSR	Transmit data/status register	16	$TDSR_H$ contains transmitter command and status information. TDSRL = TxDB contains the character to be transmitted.
Internal			
CCSR	Control character shift register	8	These registers are used for character assembly (CSSR, HSR, RxSR), disassembly (TxSR), and CRC accumulation/generation (RxCRC, TxCRC).
HSR	Holding shift register	16	
RxSR	Receiver shift register	8	
TxSR	Transmitter shift register	8	
RxCRC	Receiver CRC accumulation register	16	
TxCRC	Transmitter CRC generation register	16	

NOTES:
*H = High byte - bits 15-8
L = Low byte - bits 7-0

FUNCTIONAL DESCRIPTION

The MPCC can be functionally partitioned into receiver logic, transmitter logic, registers that can be read or loaded by the processor, and data bus control circuitry. The register bit formats are shown in figure 1 while the receiver and transmitter data paths are depicted in figures 2 and 3.

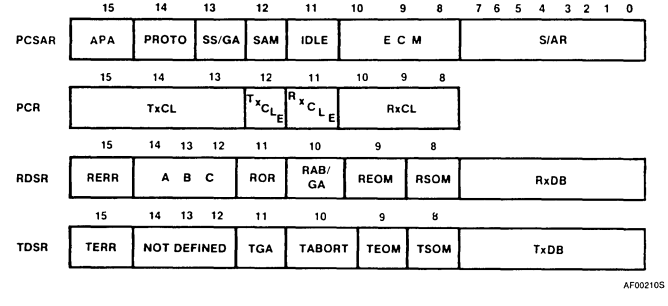
Table 2. ERROR CONTROL

CHARACTER	DESCRIPTION
FCS	Frame check sequence is transmitted/received as 16 bits following the last data character of a BOP message. The divisor is usually CRC-CCITT ($X^{16} + X^{12} + X^5 + 1$) with dividend preset to 1's but can be other wise determined by ECM. The inverted remainder is transmitter as the FCS.
BCC	Block check character is transmitted/received as two successive characters following the last data character of a BCP message. The polynomial is CRC-16 ($X^{16}+X^{15}+X^2+1$) or CRC-CCITT with dividend preset to 0's (as specified by ECM). The true remainder is transmitted as the BCC.

Table 3. SPECIAL CHARACTERS

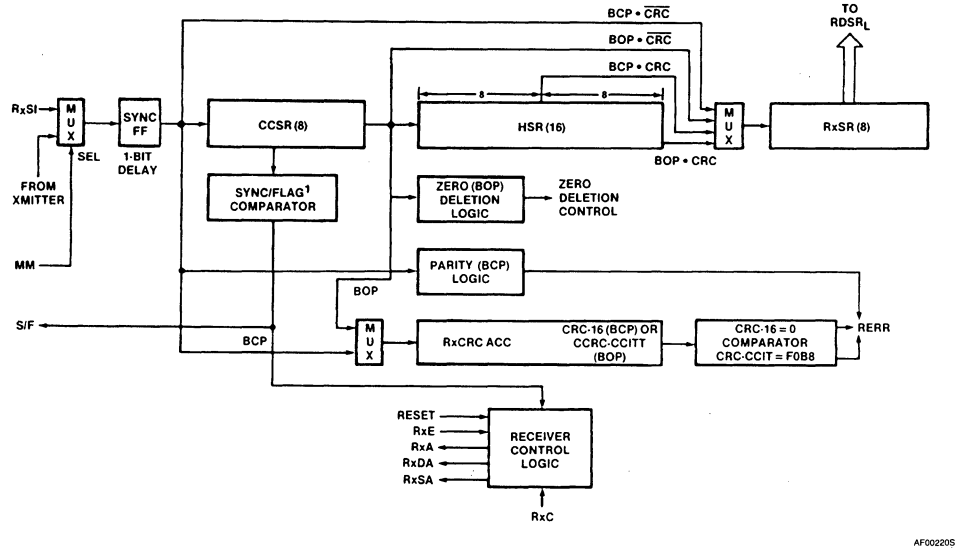
OPERATION	BIT PATTERN	FUNCTION
BOP		
FLAG	01111110	Frame message
ABORT	11111111 generation 01111111 detection	Terminate communication
GA	01111111	Terminate loop mode repeater function
Address	$(PCSAR_L)$[1]	Secondary station address
BCP		
SYNC	$(PCSAR_L)$ or (TxDB)[2] generation	Character synchronization

NOTES:
1. ( ) = contents of.
2. For IDLE = 0 or 1 respectively.



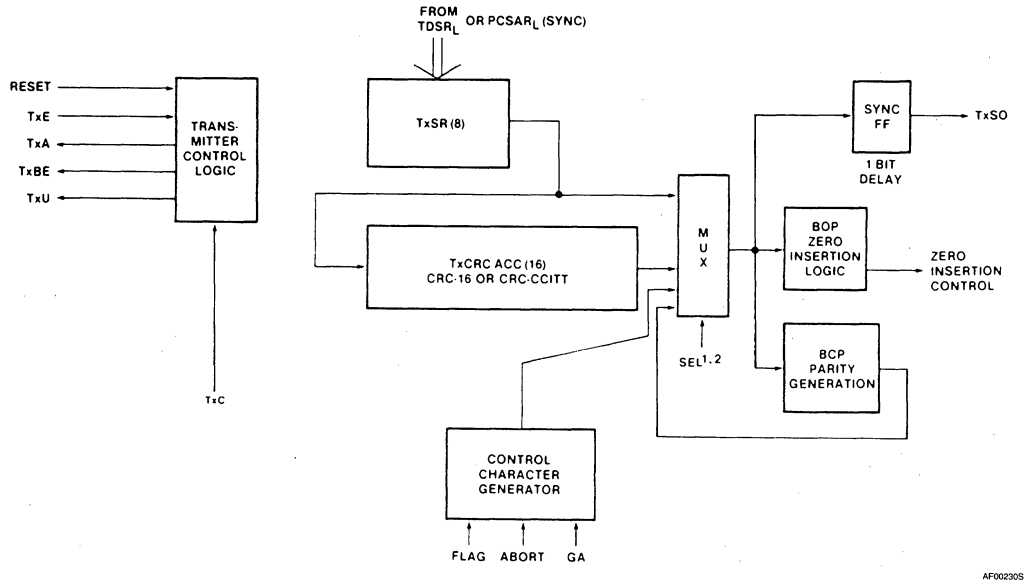
NOTE:
Refer to Register Formats for mnemonics and description.

Figure 1. Short Form Register Bit Formats



NOTES:
1. Detected in SYNC FF and 7 MS bits of CCSR.
2. In BOP mode, a minimum of two data characters must be received to turn the receiver active.

Figure 2. MPCC Receiver Data Path



NOTES:
1. TxCRC selected if TEOM = 1 and the last data character has been shifted out of TxSR.
2. In BCP parity selected will be generated after each character is shifted out of TxSR.

Figure 3. MPCC Transmitter Data Path

2

RECEIVER OPERATION

General

After initializing the parameter control registers (PCSAR and PCR), the RxE input must be set high to enable the receiver data path. The serial data on the RxSI is synchronized and shifted into an 8-bit Control Character Shift Register (CCSR) on the rising edge of RxC. A comparison between CCSR contents and the FLAG (BOP) or SYNC (BCP) character is made until a match is found. At that time, the S/F output is asserted for one RxC time and the 16-bit Holding Shift Register (HSR) is enabled. The receiver then operates as described below.

BOP Operation

A flowchart of receiver operation in BOP mode appears in figure 4. Zero deletion (after five ones are received) is implemented on the received serial data so that a data character will not be interpreted as a FLAG, ABORT, or GA. Bits following the FLAG are shifted through the CCSR, HSR, and into the Receiver Shift Register (RxSR). A character will be assembled in the RxSR and transferred to the $RDSR_L$ for presentation to the processor. At that time the RxDA output will be asserted and the processor must take the character no later than one RxC time after the next character is assembled in the RxSR. If not, an overrun ($RDSR_{11} = 1$) will occur and succeeding characters will be lost.

The first character following the FLAG is the secondary station address. If the MPCC is a secondary station ($PCSAR_{12} = 1$), the contents of RxSR are compared with the address stored in $PCSAR_L$. A match indicates the forthcoming message is intended for the station; the RxA output is asserted, the character is loaded into $RDSR_L$, RxDA is asserted and the Receive Start of Message bit (RSOM) is set. No match indicates that another station is being addressed and the receiver searches for the next FLAG.

If the MPCC is a primary station, ($PCSAR_{12} = 0$), no secondary address check is made; RxA is asserted and RSOM is set once the first non-FLAG character has been loaded into $RDSR_L$ and RxDA has been asserted. Extended address field can be supported by software if $PCSAR_{12} = 0$.

When the 8 bits following the address character have been loaded into $RDSR_L$ and RxDA has been asserted, RSOM will be cleared. The processor should read this 8-bit character and interpret it as the Control field.

Received serial data that follows is read and interpreted as the information field by the processor. It will be assembled into character lengths as specified by PCR_{8-10}. As before, RxDA is asserted each time a character has been transferred into $RDSR_L$ and is cleared

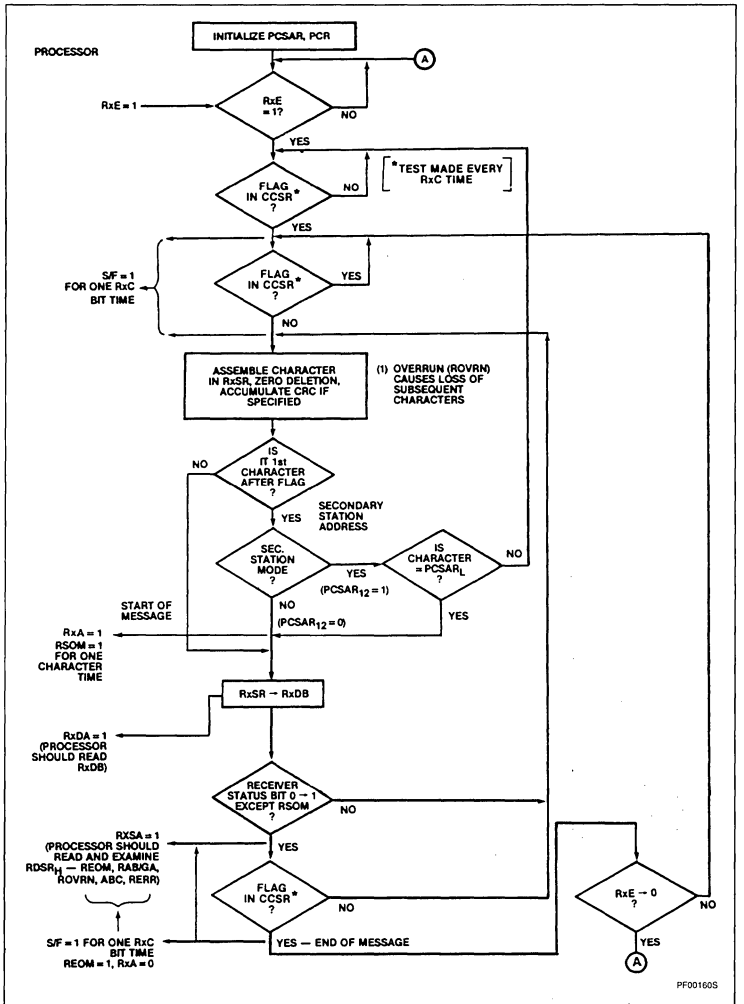


Figure 4. BOP Receive

when $RDSR_L$ is read by the processor. $RDSR_H$ should only be read when RxSA is asserted. This occurs on a zero to one transition of any bit in $RDSR_H$ except for RSOM. RxSA and all bits in $RDSR_H$ except RSOM are cleared when $RDSR_H$ is read. The processor should check $RDSR_{9-15}$ each time RxSA is asserted. If $RDSR_9$ is set, then $RDSR_{12-15}$ should be examined.

Receiver character length may be changed dynamically in response to RxDA: read the character in RxDB and write the new character length into RxCL. The character length will be changed on the next receiver character boundary. A received residual (short) character will be transferred into RxDB after the previous character in RxDB has been read, i.e. there will not be an overrun. In general the last two characters are protected from overrun.

The CRC-CCITT, if specified by $PCSAR_{8-10}$, is accumulated in RxCRC on each character following the FLAG. When the closing FLAG is detected in the CCSR, the received CRC is in the 16-bit HSR. At that time, the Receive End of Message bit (REOM) will be set; RxSA and RxDA will be asserted. The processor should read the last data character in $RDSR_L$ and the receiver status in $RDSR_{9-15}$. If $RDSR_{15} = 1$, there has been a transmission error; the accumulated CRC-CCITT is incorrect. If $RDSR_{12-14} \neq 0$, the last data charac-

ter is not of prescribed length. Neither the received CRC nor closing FLAG are presented to the processor. The processor may drop RxE or leave it active at the end of the received message.

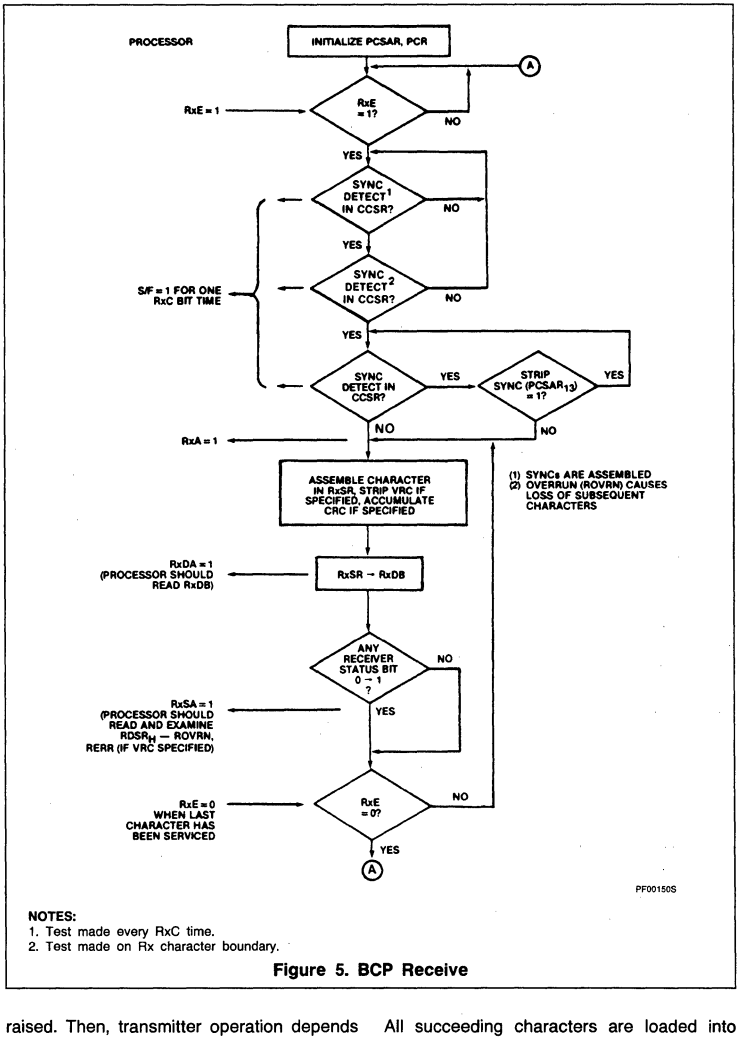
BCP Operation

The operation of the receiver in BCP mode is shown in figure 5. The receiver initially searches for two successive SYNC characters, of length specified by PCR_{8-10}, that match the contents of $PCSAR_L$. The next non-SYNC character or next SYNC character, if stripping is not specified ($PCSAR_{13} = 0$), causes RxA to be asserted and enables the receiver data path. Once enabled, all characters are assembled in RxSR and loaded into $RDSR_L$. RxDA is active when a character is available in $RDSR_L$. RxSA is active on a 0 to 1 transition of any bit in $RDSR_H$. The signals are cleared when $RDSR_L$ or $RDSR_H$ are read respectively.

If CRC-16 error control is specified by $PCSAR_{8-10}$, the processor must determine the last character received prior to the CRC field. When that character is loaded into $RDSR_L$ and RxDA is asserted, the received CRC will be in CCSR and HSR_L. To check for a transmission error, the processor must read the receiver status ($RDSR_H$) and examine $RDSR_{15}$. This bit will be set for one character time if an error free message has been received. If $RDSR_{15} = 0$, the CRC-16 is in error. The state of $RDSR_{15}$ in BCP CRC mode does not set RxSA. Note that this bit should be examined only at the end of a message. The accumulated CRC will include all characters starting with the first non-SYNC character if $PCSAR_{13} = 1$, or the character after the opening two SYNCs if $PCSAR_{13} = 0$. This necessitates external CRC generation/checking when supporting IBM's BISYNC. This can be accomplished using the Signetics SCN2653 Polynomial Generator/Checker. See Typical Applications.

If VRC has been selected for error control, parity (odd or even) is regenerated on each character and checked when the parity bit is received. A discrepancy causes $RDSR_{15}$ to be set and RxSA to be asserted. This must be sensed by the processor. The received parity bit is stripped before the character is presented to the processor.

When the processor has read the last character of the message, it should drop RxE which disables the receiver logic and initializes all receiver registers and timing.



NOTES:
1. Test made every RxC time.
2. Test made on Rx character boundary.

Figure 5. BCP Receive

TRANSMITTER OPERATION

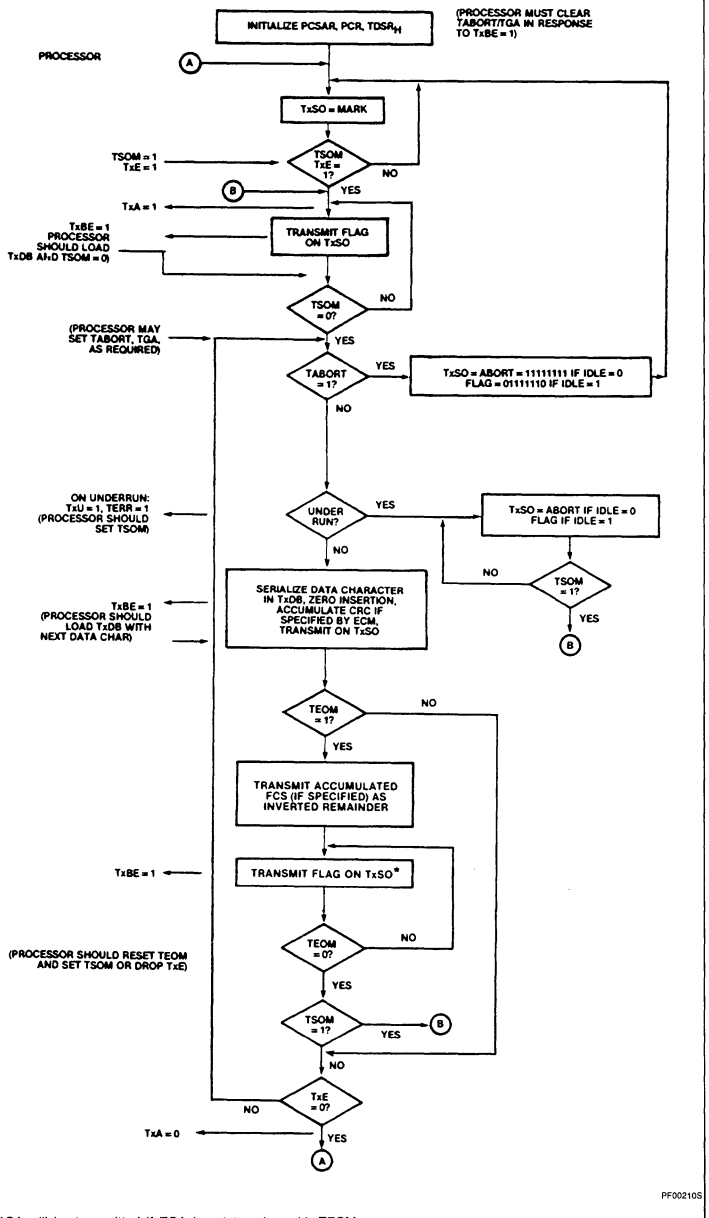
General

After the parameter control registers (PCSAR and PCR) have been initialized, TxSO is held at mark until TSOM ($TDSR_8$) is set and TxE is raised. Then, transmitter operation depends on protocol mode.

BOP Operation

Transmitter operation for BOP is shown in figure 6. A FLAG is sent after the processor sets the Transmit Start of Message bit (TSOM) and raises TxE. The FLAG is used to synchronize the message that follows. TxA will also be asserted. When TxBE is asserted by the MPCC, the processor should load $TDSR_L$ with the first character of the message. TSOM should be cleared at the same time $TDSR_L$ is loaded (16-bit data bus) or immediately thereafter (8-bit data bus). FLAGs are sent as long as TSOM = 1. For counting the number of FLAGs, the processor should reassert TSOM in response to the assertion of TxBE.

All succeeding characters are loaded into $TDSR_L$ by the processor when TxBE = 1. Each character is serialized in TxSR and transmitted on TxSO. Internal zero insertion logic stuffs a "0" into the serial bit stream after five successive "1s" are sent. This insures a data character will not match a FLAG, ABORT, or GA reserved control character. As each character is transmitted, the Frame Check Sequence (FCS) is generated as specified by Error Control Mode ($PCSAR_{8-10}$). The FCS should be the CRC-CCITT polynomial ($X^{16} + X^{12} + X^5 + 1$) preset to 1s. If an underrun occurs (processor is not keeping up with the transmitter), TxU and TERR ($TDSR_{15}$) will be asserted with ABORT or FLAG used as the TxSO line fill depending on the state of IDLE ($PCSAR_{11}$). The proces-



*GA will be transmitted if TGA is set together with TEOM.

Figure 6. BOP Transmit

sor must set TSOM to reset the underrun condition. To retransmit the message, the processor should proceed with the normal start of message sequence.

A residual character of 1 to 7 bits may be transmitted at the end of the information field. In response to TxBE, write the residual character length into TxCL and load TxDB with the residual character. Dynamic alteration of character length should be done in exactly the same sequence. The character length will be changed on the next transmit character boundary.

After the last data character has been loaded into $TDSR_L$ and sent to TxSR (TxBE = 1), the processor should set TEOM ($TDSR_9$). The MPCC will finish transmitting the last character followed by the FCS and the closing FLAG. The processor should clear TEOM and drop TxE when the next TxBE is asserted. This corresponds to the start of closing FLAG transmission. When TxE has been dropped. TxA will be low 1½ bit times after the last bit of the closing FLAG has been transmitted. TxSO will be marked after the closing FLAG has been transmitted.

If TxE and TEOM are high, the transmitter continues to send FLAGs. The processor may initiate the next message by resetting TEOM and setting TSOM, or by loading $TDSR_L$ with a data character and then simply resetting TSOM (without setting TSOM).

BCP Operation

Transmitter operation for BCP mode is shown in figure 7. TxA will be asserted after TSOM = 1 and TxE is raised. At that time SYNC characters are sent from $PCSAR_L$ or $TDSR_L$ (IDLE = 0 or 1) as long as TSOM = 1. TxBE is asserted at the start of transmission of the first SYNC character. For counting the number of SYNCs, the processor should reassert TSOM in response to the assertion of TxBE. When TSOM = 0 transmission is from $TDSR_L$, which must be loaded with characters from the processor each time TxBE is asserted. If this loading is delayed for more than one character time, an underrun results: TxU and TERR are asserted and the TxSO line fill depend on IDLE ($PCSAR_{11}$). The processor must set TSOM and retransmit the message to recover. This is not compatible with IBM's BISYNC, so that the user must not underrun when supporting that protocol.

CRC-16, if specified by $PCSAR_{8-10}$, is generated on each character transmitted from $TDSR_L$ when TSOM = 0. The processor must set TEOM = 1 after the last data character has been sent to TxSR (TxBE = 1). The MPCC will finish transmitting the last data character and the CRC-16 field before sending SYNC characters which are transmitted as long as TEOM = 1. If SYNCs are not desired after CRC-16 transmission, the processor should clear TEOM and lower TxE when the TxBE corresponding to the start of CRC-16 transmission is asserted. When TEOM = 0, the line is marked and a new

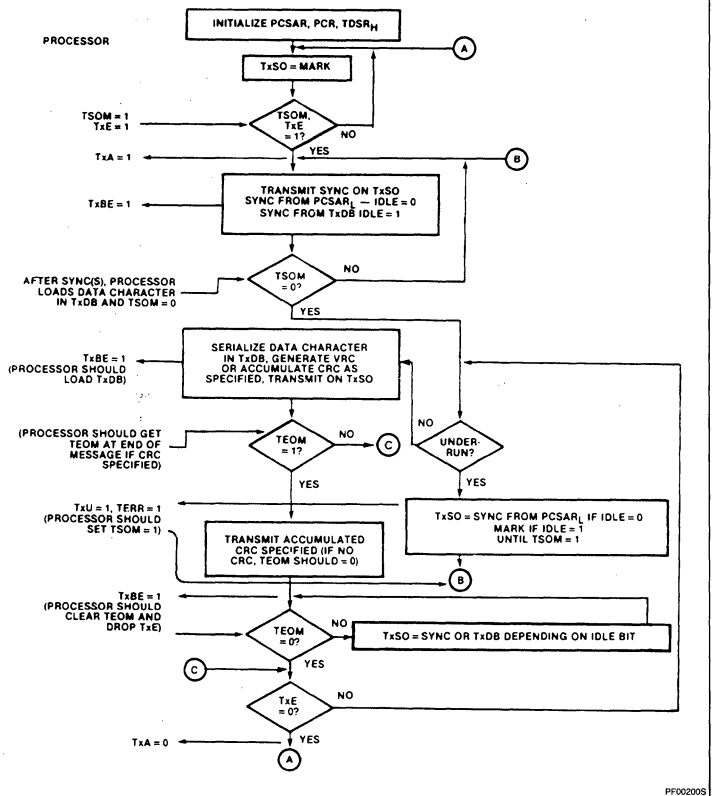


Figure 7. BCP Transmit

message may be initiated by setting TSOM and raising TxE.

If VRC is specified, it is generated on each data character and the data character length must not exceed 7 bits. For software LRC or CRC, TEOM should be set only if SYNC's are required at the end of the message block.

Special Case

The capability to transmit 16 spaces is provided for line turnaround in half duplex mode or for a control recovery situation. This is achieved by setting TSOM and TEOM, clearing TEOM when TxBE = 1, and proceeding as required.

PROGRAMMING

Prior to initiating data transmission or reception, PCSAR and PCR must be loaded with control information from the processor. The contents of these registers (see Register Format section) will configure the MPCC for the user's specific data communication environment. These registers should be loaded during power-on initialization and after a reset operation. They can be changed at any time that the respective transmitter or receiver is disabled.

The default value for all registers is zero. This corresponds to BOP, primary station mode, 8-bit character length, FCS = CRC-CCITT preset to 1s.

For BOP mode the character length register (PCR) may be set to the desired values during system initialization. The address and control fields will automatically be 8-bits. If a residual character is to be transmitted, TxCL should be changed to the residual character length prior to transmission of that character.

DATA BUS CONTROL

The processor must set up the MPCC register address (A2 – A0), chip enable (CE), byte select (BYTE), and read/write ($\overline{R}$/W) inputs before each data bus transfer operation.

During a read operation ($\overline{R}$/W = 0), the leading edge of DBEN will initiate an MPCC read cycle. The addressed register will place its contents on the data bus. If BYTE = 1, the 8-bit byte is placed on DB15-08 or DB07-00 depending on the H/L status of the register addressed. Unused bits in RDSR$_L$ are zero. If BYTE = 0, all 16 bits (DB15-00) contain MPCC information. The trailing edge of DBEN will reset RxDA and/or RxSA if RDSR$_L$ or RDSR$_H$ is addressed respectively.

DBEN acts as the enable and strobe so that the MPCC will not begin its internal read cycle until DBEN is asserted.

During a write operation ($\overline{R}$/W = 1), data must be stable on DB$_{15-08}$ and/or DB$_{07-00}$ prior to the leading edge of DBEN. The stable data is strobed into the addressed register by DBEN. TxBE will be cleared if the addressed register was TDSR$_H$ or TDSR$_L$.

2

Table 4. MPCC REGISTER ADDRESSING

	A2	A1	A0	REGISTER
BYTE = 0	16-BIT DATA BUS = DB_{15} - DB_{00}			
	0	0	X	RDSR
	0	1	X	TDSR
	1	0	X	PCSAR
	1	1	X	PCR*
BYTE = 1	8-BIT DATA BUS = DB_{7-0} or DB_{15-8}**			
	0	0	0	$RDSR_L$
	0	0	1	$RDSR_H$
	0	1	0	$TDSR_L$
	0	1	1	$TDSR_H$
	1	0	0	$PCSAR_L$
	1	0	1	$PCSAR_H$
	1	1	0	PCR_L*
	1	1	1	PCR_H

NOTES:
* PCR lower byte does not exist. It will be all "0"s when read.
** Corresponding high and low order pins must be tied together.

Table 5. PARAMETER CONTROL REGISTER (PCR) – (R/W)

BIT	NAME	MODE	FUNCTION
00 – 07	Not Defined		
08 – 10	RxCL	BOP/BCP	Receiver character length is loaded by the processor when RxCLE = 0. The character length is valid after transmission of single byte address and control fields have been received.
11	RxCLE	BOP/BCP	Receiver character length enable should be zero when the processor loads RxCL. The remaining bits of PCR are not affected during loading. Always 0 when read.
12	TxCLE	BOP/BCP	Transmitter character length enable should be zero when the processor loads TxCL. The remaining bits of PCR are not affected during loading. Always 0 when read.
13 – 15	TxCL	BOP/BCP	Transmitter character length is loaded by the processor when TxCLE = 0. Character bit length specification format is identical to RxCL. It is valid after transmission of single byte address and control fields.

10	9	8	Char length (bits)
0	0	0	8
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Table 6. PARAMETER CONTROL SYNC/ADDRESS REGISTER (PCSAR)-(R/W)

BIT	NAME	MODE	FUNCTION
00 – 07	S/AR	BOP	SYNC/address register. Contains the secondary station address if the MPCC is a secondary station. The contents of this register is compared with the first received non-FLAG character to dertermine if the message is meant for this station.
		BCP	SYNC character is loaded into this register by the processor. It is used for receive and transmit bit synchronization with bit length specified by RxCL and TxCL.
08 – 10	ECM	BOP/BCP	**Error Control Mode** / **10** / **9** / **8** / **Suggested Mode** / **Char. length** CRC-CCITT preset to 1's / 0 / 0 / 0 / BOP / 1 – 8 CRC-CCITT preset to 0's / 0 / 0 / 1 / BCP / 8 Not used / 0 / 1 / 0 / — / CRC-16 preset to 0's / 0 / 1 / 1 / BCP / 8 VRC odd / 1 / 0 / 0 / BCP / 5 – 7 VRC even / 1 / 0 / 1 / BCP / 5 – 7 Not used / 1 / 1 / 0 / — / No error control / 1 / 1 / 1 / BCP/BOP / 5 – 8 ECM should be loaded by the processor during initialization or when both data paths are idle.
11	IDLE		Determines line fill character to be used if transmitter underrun occurs (TxU asserted and TERR set) and transmission of special characters for BOP/BCP.
		BOP	IDLE = 0, transmit ABORT characters during underrun and when TABORT = 1. IDLE = 1, transmit FLAG characters during underrun and when TABORT = 1.
		BCP	IDLE = 0 transmit initial SYNC characters and underrun line fill characters from the S/AR. IDLE = 1 transmit initial SYNC characters from TxDB and marks TxSO during underrun.
12	SAM	BOP	Secondary Address Mode = 1 if the MPCC is a secondary station. This facilitates automatic recognition of the received secondary station address. When transmitting, the processor must load the secondary address into TxDB. SAM = 0 inhibits the received secondary address comparison which serves to activate the receiver after the first non-FLAG character has been received.
13	SS/GA		Strip SYNC/Go Ahead. Operation depends on mode.
		BOP	SS/GA = 1 is used for loop mode only and enables GA detection. When a GA is detected as a closing character, REOM and RAB/GA will be set and the processor should terminate the repeater function. SS/GA = 0 is the normal mode which enables ABORT detection. It causes the receiver to terminate the frame upon detection of an ABORT or FLAG.
		BCP	SS/GA = 1, causes the receiver to strip SYNC's immediately following the first two SYNC's detected. SYNC's in the middle of a message will not be stripped. SS/GA = 0, presents any SYNC's after the initial two SYNC's to the processor.
14	PROTO		Determines MPCC Protocol mode
		BOP	PROTO = 0
		BCP	PROTO = 1
15	APA	BOP	All parties address. If this bit is set, the receiver data path is enabled by an address field of '11111111' as well as the normal secondary station address.

Table 7. TRANSMIT DATA/STATUS REGISTER (TDSR) (R/W EXCEPT TDSR15)

BIT	NAME	MODE	FUNCTION
00 – 07	TxDB	BOP/BCP	Transmit data buffer. Contains processor loaded characters to be serialized in TxSR and transmitted on TxSO.
08	TSOM		Transmitter start of message. Set by the processor to initate message transmission provided TxE = 1.
		BOP	TSOM = 1 generates FLAGs. When TSOM = 0 transmission is from TxDB and FCS generation (if specified) begins. FCS, as specified by $PCSAR_{8-10}$, should be CRC-CCITT preset to 1's.
		BCP	TSOM = 1 generates SYNCs from $PCSAR_L$ or transmits from TxDB for IDLE = 0 or 1 respectively. When TSOM = 0 transmission is from TxDB and CRC generation (if specified) begins.
09	TEOM		Transmit end of message. Used to terminate a transmitted message.
		BOP	TEOM = 1 causes the FCS and the closing FLAG to be transmitted following the transmission of the data character in TxSR. FLAGs are transmitted until TEOM = 0. ABORT or GA are transmitted if TABORT or TGA are set when TEOM = 1.
		BCP	TEOM = 1 causes CRC-16 to be transmitted (if selected) followed by SYNCs from $PCSAR_L$ or TxDB (IDLE = 0 or 1). Clearing TEOM prior to the end of CRC-16 transmission (when TxBE = 1) causes TxSO to be marked following the CRC – 16. TxE must be dropped before a new message can be initiated. If CRC is not selected, TEOM should not be set.
10	TABORT	BOP	Transmitter abort = 1 will cause ABORT or FLAG to be sent (IDLE = 0 or 1) after the current character is transmitted. (ABORT = 11111111)
11	TGA	BOP	Transmit go ahead (GA) instead of FLAG when TEOM = 1. This facilitates repeater termination in loop mode. (GA = 01111111)
12 – 14	Not Defined		
15	TERR	Read only	Transmitter error = 1 indicates the TxDB has not been loaded in time (one character time-½ TxC period after TxBE is asserted) to maintain continuous transmission. TxU will be asserted to inform the processor of this condition. TERR is cleared by setting TSOM. See timing diagram.
		BOP	ABORT's or FLAG's are sent as fill characters (IDLE = 0 or 1)
		BCP	SYNC's or MARK's are sent as fill characters (IDLE = 0 or 1). For IDLE = 1 the last character before underrun is not valid.

Table 8. RECEIVER DATA/STATUS REGISTER (RDSR) – (READ ONLY)

BIT	NAME	MODE	FUNCTION
00 – 07	RxDB	BOP/BCP	Receiver data buffer. Contains assembled characters from the RxSR. If VRC is specified, the parity bit is stripped.
08	RSOM	BOP	Receiver start of message = 1 when a FLAG followed by a non-FLAG has been received and the latter character matches the secondary station if SAM = 1. RxA will be asserted when RSOM = 1. RSOM resets itself after one character time and has no affect on RxSA.
09	REOM	BOP	Receiver end of message = 1 when the closing FLAG is detected and the last data character is loaded into RxDB or when an ABORT/GA character is received. REOM is cleared on reading $RDSR_H$, reset operation, or dropping of RxE.
10	RAB/GA	BOP	Received ABORT or GA character = 1 when the receiver senses an ABORT character if SS/GA = 0 or a GA character if SS/GA = 1. RAB/GA is cleared on reading $RDSR_H$, reset operation, or dropping of RxE. A received abort does not set RxDA.
11	ROR	BOP/BCP	Receiver overrun = 1 indicates the processor has not read last character in the RxDB within one character time + ½ RxC period after RxDA is asserted. Subsequent characters will be lost. ROR is cleared on reading $RDSR_H$, reset operation, or dropping of RxE.
12 – 14	ABC	BOP	Assembled bit count. Specifies the number of bits in the last received data character of a message and should be examined by the processor when REOM = 1(RxDA and RxSA asserted). ABC = 0 indicates the message was terminated (by a flag or GA) on a character boundary as specified by PCR_{8-10}. Otherwise, ABC = number of bits in the last data character. ABC is cleared when $RDSR_H$ is read, reset operation, or dropping RxE. The residual character is right justified in $RDSR_L$.
15	RERR	BOP/BCP	Receiver error indicator should be examined by the processor when REOM = 1 in BOP, or when the processor determines the last data character of the message in BCP with CRC or when RxSA is set in BCP with VRC. CRC-CCITT preset to 1's/0's as specified by $PCSAR_{8-10}$: RERR = 1 indicates FCS error (CRC ≠ F0B8 or ≠ 0) RERR = 0 indicates FCS received correctly (CRC = F0B8 or = 0) CRC-16 preset to 0's on 8-bit characters specified by $PSCAR_{8-10}$: RERR = 1 indicates CRC-16 received correctly (CRC = 0). RERR = 0 indicates CRC-16 error (CRC≠0) VRC specified by $PCSAR_{8-10}$: RERR = 1 indicates VRC error RERR = 0 indicates VRC is correct.

ABSOLUTE MAXIMUM RATINGS[1]

	PARAMETER	RATING	UNIT
T_A	Operating ambient temperature[2]	Note 4	°C
T_{STG}	Storage temperature	−65 to +150	°C
	Input or output voltages with respect to GND[3]	−0.3 to +15	V
V_{CC}	With respect to GND	−0.3 to +7	V

2

DC ELECTRICAL CHARACTERISTICS[4,5]

PARAMETER	TEST CONDITIONS	LIMITS			UNIT
		Min	Typ	Max	
Input voltage					V
V_{IL} Low				0.8	
V_{IH} High		2.0			
Output voltage					V
V_{OL} Low	I_{OL} = 1.6mA			0.4	
V_{OH} High	I_{OH} = −100μA	2.4			
I_{CC} Power supply current	V_{CC} = 5.25V, T_A = 0°C			150	mA
Leakage current					μA
I_{IL} Input	V_{IN} = 0 to 5.25V			10	
I_{OL} Output	V_{OUT} = 0 to 5.25V			10	
Capacitance					pF
C_{IN} Input	V_{IN} = 0V, f = 1MHz			20	
C_{OUT} Output	V_{OUT} = 0V, f = 1MHz			20	

AC ELECTRICAL CHARACTERISTICS[4,5,6]

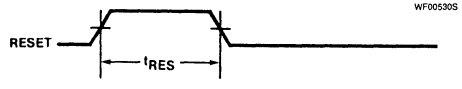
PARAMETER	1MHz CLOCK VERSION			2MHz CLOCK VERSION			UNIT
	Min	Typ	Max	Min	Typ	Max	
Set-up and hold time							ns
t_{ACS} Address/control set-up	50			50			
t_{ACH} Address/control hold	0			0			
t_{DS} Data bus set-up (write)	50			50			
t_{DH} Data bus hold (write)	0			0			
t_{RXS} Receiver serial data set-up	150			150			
t_{RxH} Receiver serial data hold	150			150			
Pulse width							ns
t_{RES} RESET	250			250			
t_{DBEN} DBEN	250		m[7]	200		m[7]	
Delay Time							ns
t_{DD} Data bus (read)			200			170	
t_{TxD} Transmit serial data			325			250	
t_{DBEND} DBEN to DBEN delay	200			200			
t_{DF} Data bus float time (read)			150			150	ns
f Clock (RxC,TxC) frequency			1.0			2.0	MHz
t_{CLK1} Clock high (MM = 0)	340			165			ns
t_{CLK2} Clock high (MM = 1)	490			240			
t_{CLK0} Clock low	490			240			

NOTES:

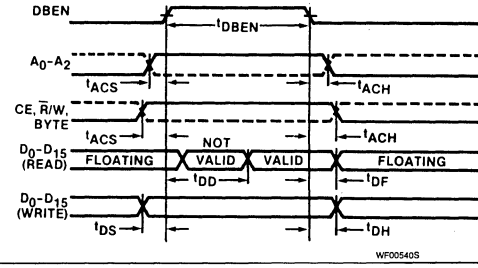
1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation sections of this specification is not implied.
2. For operating at elevated temperatures the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified. See ordering code table for applicable temperature range and operating supply range.
5. All voltage measurements are referenced to ground. All time measurements are at 0.8V or 2.0V. Input voltage levels for testing are 0.4V and 2.4V.
6. Output load C_L = 100pF.
7. m = TxC low and applies to writing to $TDSR_H$ only.

TIMING DIAGRAMS

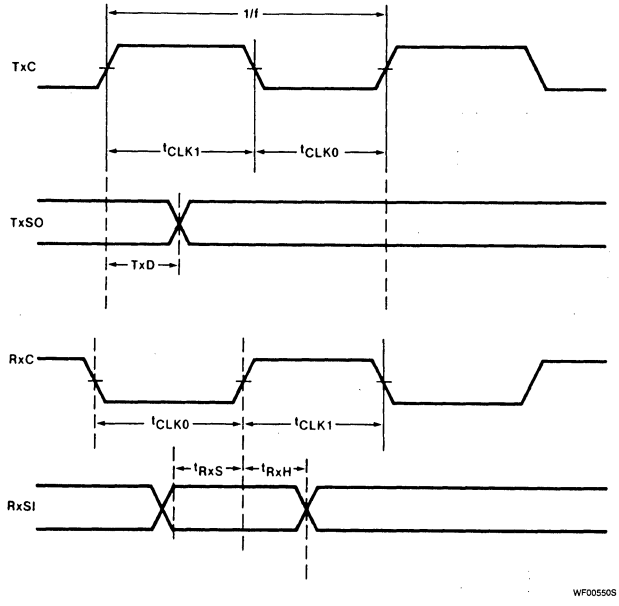
RESET



READ AND WRITE DATA BUS



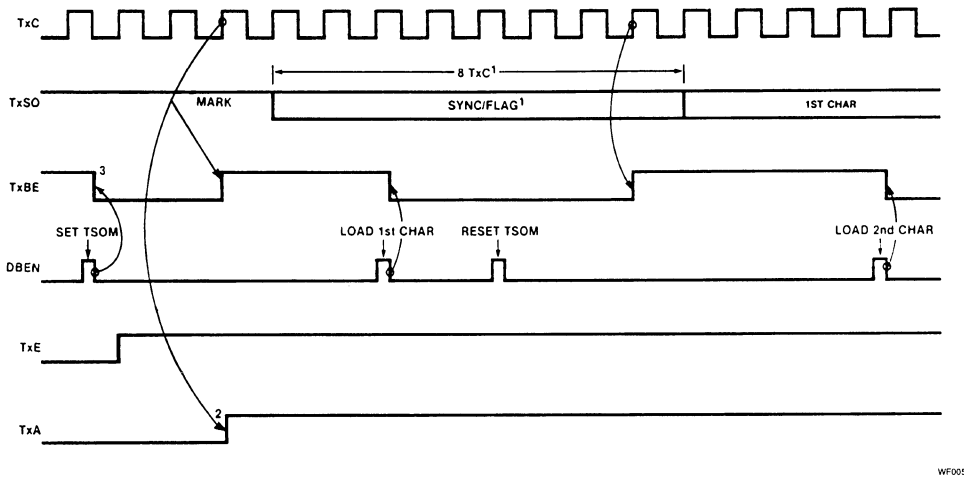
CLOCK



2

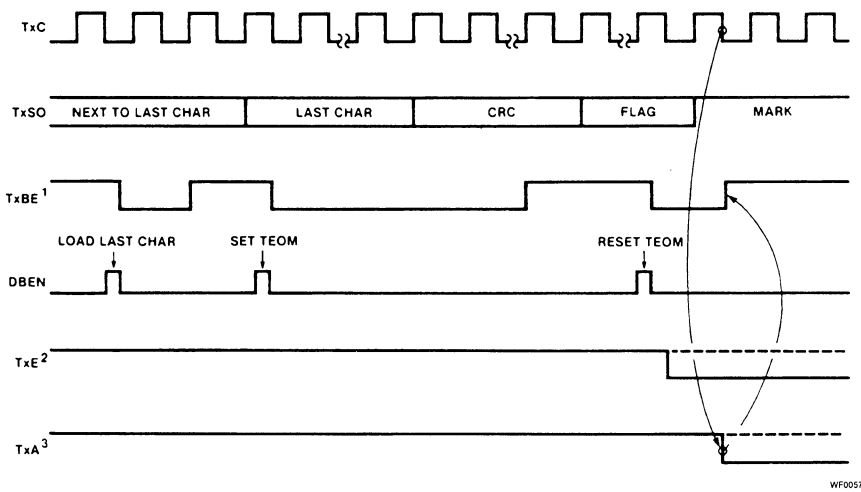
TIMING DIAGRAMS (Continued)

TRANSMIT – START OF MESSAGE



NOTES:
1. SYNC may be 5 to 8 bits and will contain parity bit as specified.
2. TxA goes high relative to TxC rising edge after TSOM has been set and TxE has been raised.
3. TxBE goes low relative to DBEN falling edge on the first write transfer into TDSR. It is reasserted 1 TxC time before the first bit of the transmitted SYNC/FLAG. TxBE then goes low relative to DBEN falling edge when writing into $TDSR_H$ and/or $TDSR_L$. It is reasserted on the rising edge of the TxC that corresponds to the transmission of the last bit of each character, except in BOP mode when the CRC is to be sent as the next character (see Transmit Timing – End of Message).

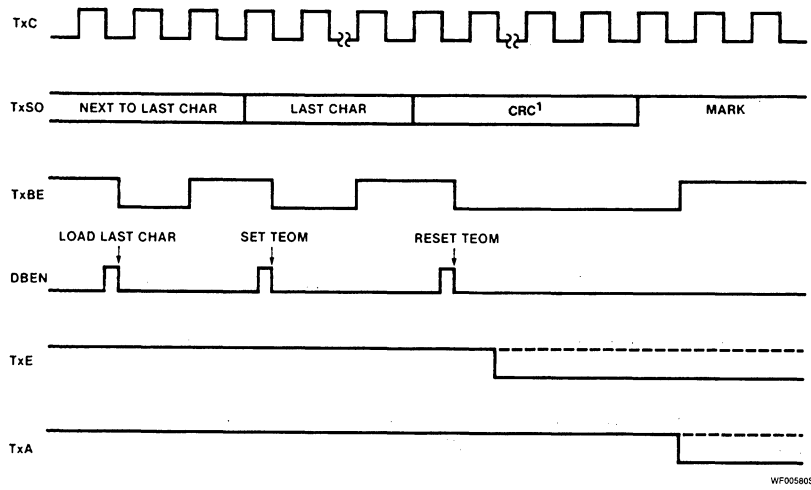
TRANSMIT — END OF BOP MESSAGE



NOTES:
1. TxBE goes low relative to the falling edge of DBEN corresponding to loading $TDSR_{H/L}$. It goes high one TxC before character transmission begins and also when TxA has been dropped.
2. TxE can be dropped before resetting TEOM if TxBE (corresponding to the closing FLAG) is high. Alternatively TxE can remain high and a new message initiated.
3. TxA goes low after TxE has been dropped and 1½ TxC's after the last bit of the closing FLAG has been transmitted.

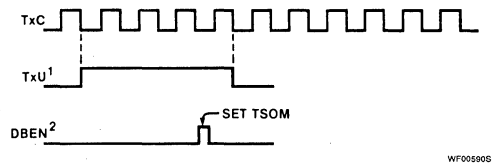
TIMING DIAGRAMS (Continued)

TRANSMIT TIMING — END OF BCP MESSAGE



NOTES:
1. When SCN2652 generated CRC is not required, TEOM should only be set if SYNCs are to follow the message block. In that case, TxE should be dropped in response to TxBE (which corresponds to the start of transmission of the last character). When CRC is required, TxE must be dropped before CRC transmission is complete. Otherwise, the contents of TxDB will be shifted out on TxSO. This facilitates transmission of contiguous messages.

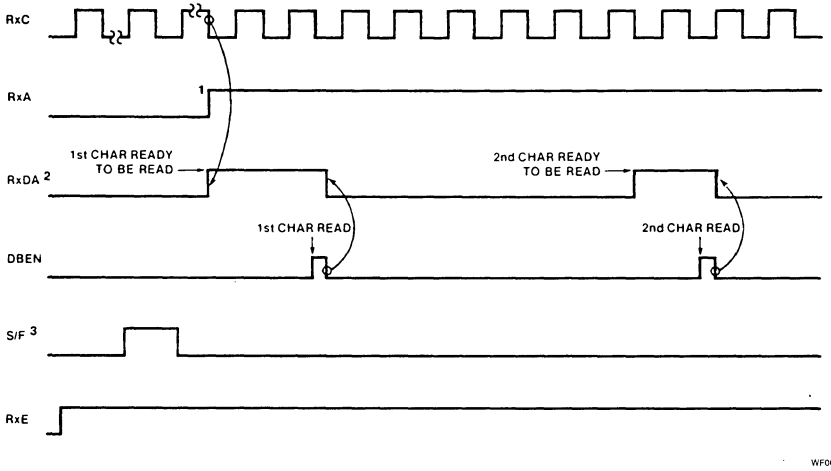
TRANSMIT UNDERRUN



NOTES:
1. TxU goes active relative to TxC falling edge if TxBE has not been serviced after n-½ TxC times (where n = transmit character length). TxU is reset on the TxC falling edge following assertion of the TSOM command.
2. An underrun will occur at the next character boundary if TEOM is reset and the transmitter remains enabled, unless the TSOM command is asserted or a character is loaded into the TxDB.

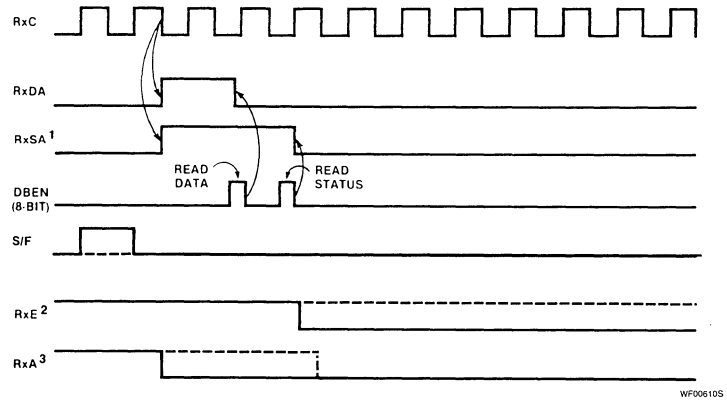
TIMING DIAGRAMS (Continued)

RECEIVE — START OF MESSAGE



NOTES:
1. RxA goes high relative to falling edge of RxC when RxE is high and: a. A data character following two SYNC's is in RxDB (BCP mode). b. Character following FLAG is in RxDB (BOP primary station mode). c. Character following FLAG is in RxDB and character matches the secondary station address or all parties address (BOP secondary station mode).
2. RxDA goes high on RxC falling edge when a character in RxDB is ready to be read. It comes up before RxSA and goes low on the falling edge of DBEN when RxDB is read.
3. S/F goes high relative to rising edge of RxC anytime a SYNC (BCP) or FLAG (BOP) is detected.

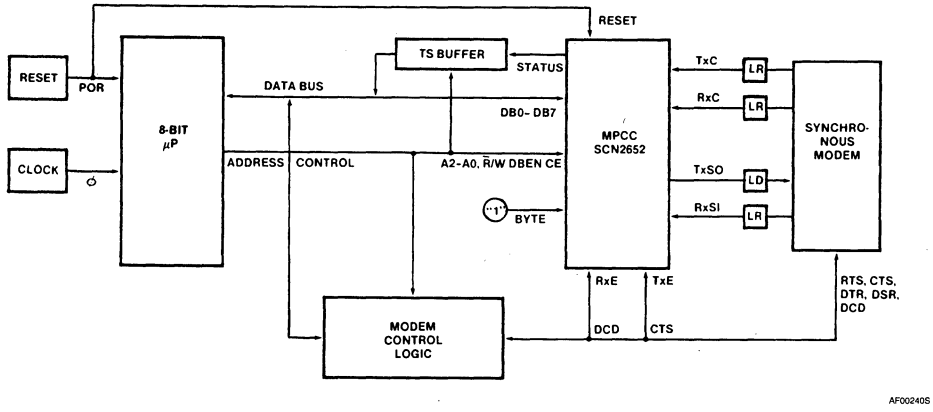
RECEIVE END OF MESSAGE



NOTES:
1. At the end of a BOP message, RxSA goes high when FLAG detection (S/F 1) forces REOM to be set. Processor should read the last data character ($RDSR_L$) and status ($RDSR_H$) which resets RxDA and RxSA respectively. For BCP end of message, RxSA may not be set and S/F = 0. The processor should read the last data character and status.
2. RxE must be dropped for BCP with non-contiguous messages. It may be left on at the end of a BOP message (see BOP Receive Operation).
3. RxA is reset relative to the falling edge of RxC after the closing FLAG of a BOP message (REOM = 1 and RxSA active.) or when RxE is dropped.

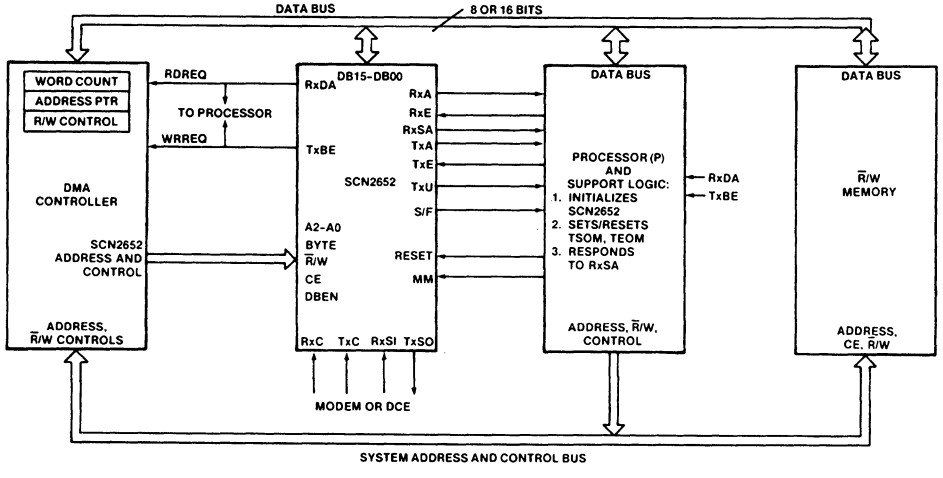
TYPICAL APPLICATIONS

SCN2652 MPCC MICROPROCESSOR INTERFACE



NOTES:
1. Possible μP interrupt requests are: RxDA RxSA TxBE TxU
2. Other SCN2652 status signals and possible uses are: S F line idle indicator, frame delimiter. RxA handshake on RxE, line turn around control. TxA handshake on TxE, line turn around control.
3. Line drivers/receivers (LD/LR) convert EIA to TTL voltages and vice-versa.
4. RTS should be dropped after the CRC (BCP) or FLAG (BOP) has been transmitted. This forces CTS low and TxE low.
5. Corresponding high and low order bits of DB must be OR tied.

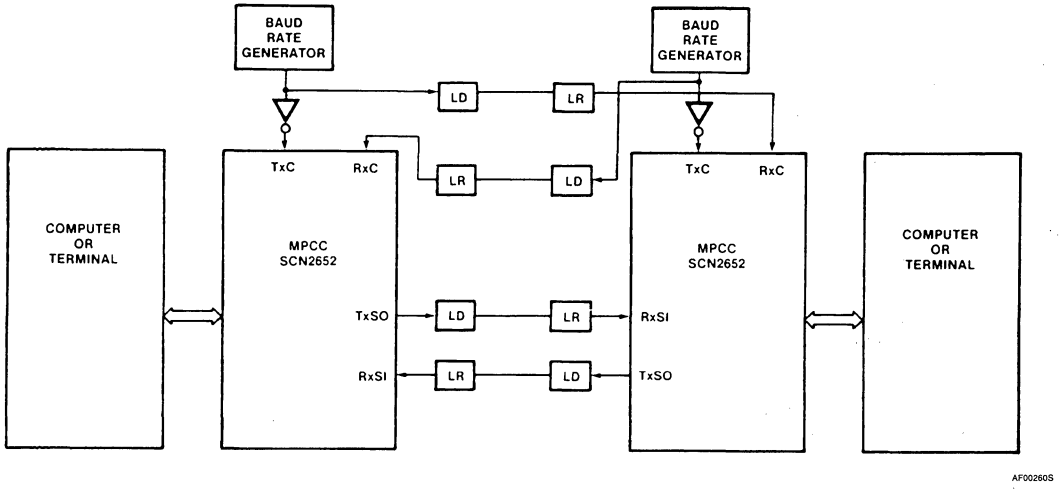
DMA/PROCESSOR INTERFACE



For non-DMA operation TxBE and RxDA are sent to the processor which then loads or reads data characters as required.

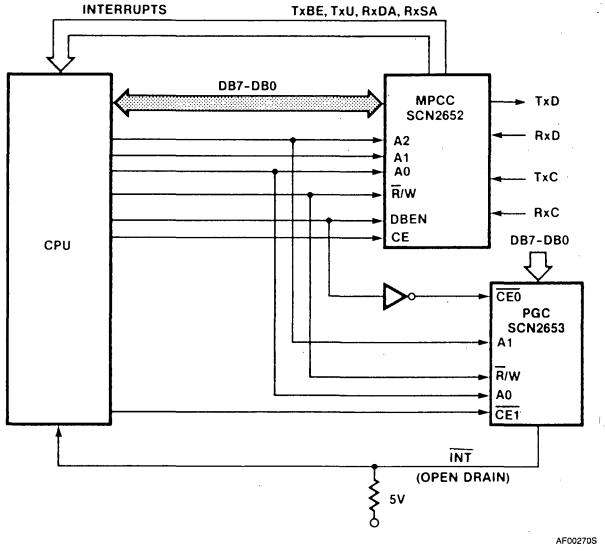
TYPICAL APPLICATIONS (Continued)

CHANNEL INTERFACE



No Modem — DC Baseband Transmission

SCN2652/SCN2653 INTERFACE TYPICAL PROTOCOLS:
BISYNC, DDCMP, SDLC, HDLC



Signetics

SCN2653/SCN68653
Polynomial Generator Checker (PGC)

Product Specification

Microprocessor Products

DESCRIPTION

The Signetics SCN2653/68653 Polynomial Generator Checker (PGC) is a polynomial generator checker/character comparator circuit that complements a receiver/transmitter (R/T or USART/USRT/UART) in the support of character oriented data link controls. Table 1 defines many of the more commonly used PGC terms and abbreviations.

Parallel data characters transferred between the CPU and R/T are monitored by the PGC which performs block check character (BCC) and parity (VRC) generation/checking, single character detection, and two character sequence detection. Since the PGC operates on parallel characters, the data transmission format may be serial (synchronous or asynchronous) or parallel.

There are four modes of BCC accumulation and each mode can select one of three polynomials to compute the BCC. In the BISYNC normal and transparent modes, the PGC determines which characters are to be accumulated and which characters are to be excluded from the accumulation. The block terminating characters and the initiation and termination of BISYNC transparent text can be detected and an interrupt generated. The single interrupt output represents the inclusive OR of four maskable status conditions.

In the automatic accumulation mode, all characters are accumulated while the single accumulate mode requires a specific accumulation command for each character to be accumulated.

Character accumulation control and character comparisons are facilitated by a character class array which places each of 128 characters into one of four character classes. The four classes are normal, SYN/BISYNC not included, block terminating character (BTC)/search character (SC), and secondary search character (SSC).

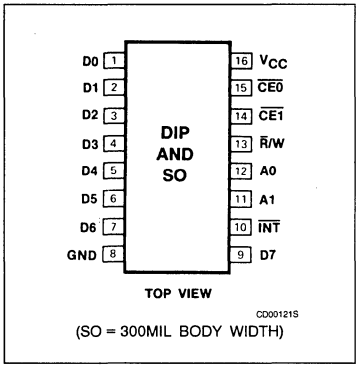
FEATURES

- **Parallel Block Check Character accumulation/checking: CRC-16, CRC-12, LRC-8**
- **BISYNC normal and transparent modes**
- **Automatic or single character accumulation modes**
- **Character detection - up to 128 characters**
- **Two character sequence detection; examples: DLE-STX, ACK0, ACK1, WACK, RVI, DISC, WBT**
- **6, 7, or 8-bit characters**
- **VRC generation/checking on data bus**
- **Four maskable interrupt conditions**
- **Four classes of characters**
- **Internal power-on reset**
- **Maximum character accumulation rate of 500kHz (4Mbps)**
- **Directly compatible with Signetics SCN2651, SCN2652 and SCN2661**
- **No system clock required**
- **TTL compatible inputs and outputs**
- **Single 5V supply**
- **16-pin dual in line package**

APPLICATIONS

- **Character oriented data link control:**
  - **dedicated to one USART/USRT**
  - **multiplexed among several USART/USRTs**
- **Automated BISYNC with 2661 (minimal software intervention)**
- **BCC and VRC generation/detection on a block of memory or peripheral data**
- **Programmable character array comparator**

PIN CONFIGURATION



(SO = 300MIL BODY WIDTH)

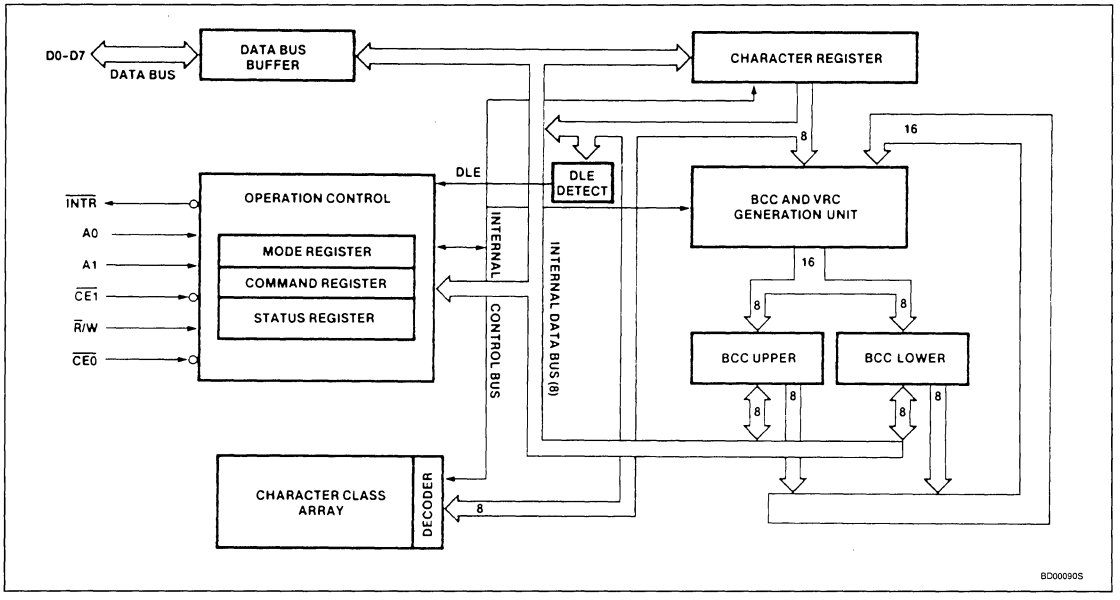
Polynomial Generator Checker (PGC) SCN2653/SCN68653

ORDERING CODE

PACKAGES	V_{CC} = 5V± 5%, T_A = 0°C to +70°C
Ceramic DIP	SCN2653AC4I16
Plastic DIP	SCN2653AC4N16
Small Outline (SO)	SCN2653ACD16

NOTE:
SCN68653 is identical to SCN2653. Order using numbers shown above.

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{CC}	16	I	**+5V:** Power supply
GND	8	I	Ground
A1-A0	11,12	I	**Address Lines:** Used to select internal PGC registers or character class array.
R/W	13	I	**Read/Write:** Read command when low, write command when high.
CE0	15	I	**Chip Enable:** Connected to chip enable input of a receiver/transmitter (R/T) circuit. It is used to strobe data being transferred between the CPU and the R/T into the PGC character register.
CE1	14	I	**Chip Enable:** Used in conjunction with the R/W signal to enable the transfer of data between the PGC and the CPU or DMA controller and to initialize the PGC registers.
D7-D0	9,7-1	I/O	**Data Bus:** 8-bit three-state bidirectional bus used to transfer data to or from the PGC via CE0 or CE1. All data, mode words, command words, and status information are transferred on this bus. D0 is the least significant bit; D7 is the most significant bit.
INT	10	O	**Interrupt:** Open drain active low interrupt output that signals the CPU that one or more maskable conditions are true: BCC error, VRC error, BTC/SC detect, SSC detect. The true conditions can be determined by reading the status register which in turn deactivates INT. A power on, clear BCC, or master reset command causes INT to be inactive (high).

Additional PGC applications include off-line R/T operation where the BCC is generated on data not sent to the R/T, BCC multiplexing by sharing the PGC among several R/Ts and reading/writing the partial BCC accumulation on a character by character basis, VRC generation/checking on characters appearing on a bidirectional data bus, and programmable character comparisons or searches.

PGC operation is half duplex (either receive or transmit, one way or two way alternate). Full duplex (two way simultaneous) is achieved by using two PGCs. The device is directly compatible with the Signetics SCN2651 Programmable Communications interface (PCI) and SCN2661 Enhanced Programmable Communications interface (EPCI). When used in BISYNC modes with the SCN2661, software requirements are minimized by the SCN2653-SCN2661 control character comparisons, character sequence comparisons, and automatic DLE insertion/detection.

Other bus oriented R/Ts can be interfaced to the PGC with a minimum of external circuitry. See figure 1 for a typical system configuration.

This NMOS LSI circuit is TTL compatible, operates from a single + 5V supply and is contained in a 16 pin dual in line package.

BLOCK DIAGRAM

The PGC consists of six major sections. These are the operation control, character class array, DLE ROM, character register, BCC and parity generators, and BCC registers. These sections communicate with each other via an internal data bus and an internal control bus. The internal data bus interfaces to the CPU data bus via a data bus buffer.

Operation Control Unit

This functional block stores configuration and operation instructions from the CPU and generates appropriate signals to control the device operation. It also contains read and write circuits to permit communications between the CPU and the PGC registers via the data bus. The mode, command, and status registers are in this logic block.

Character Register

Characters to be considered for BCC generation, parity generation and checking, or character comparisons are loaded into this register by either $\overline{CE0}$ or $\overline{CE1}$. This register serves as an input to the BCC and VRC generator, where the accumulation and parity generation takes place. The character register also serves as the input for character class array and DLE comparisons.

Table 1. GLOSSARY

TERM/ABBREVIATION	DEFINITION
BCC	Block check character
BTC	Block terminating character
SC	Search character
SSC	Second search character (preceded by DLE)
CRC-16	$X^{16} + X^{15} + X^2 + 1$ divisor, dividend pre-cleared
CRC-12	$X^{12} + X^{11} + X^3 + X^2 + X + 1$ divisor, dividend pre-cleared
LRC-8	Horizontal parity on least significant 7 bits; vertical parity on most significant bit
VRC	Vertical redundancy check (character parity)
R/T	Receiver/transmitter circuit. Also known as USART/USRT/UART/PCI/MPCC
BISYNC	IBM binary synchronous communications (BSC), ANSI X3.28, ISO 1745
MSB	Most significant bit
LSB	Least significant bit
Rx	Receive
Tx	Transmit

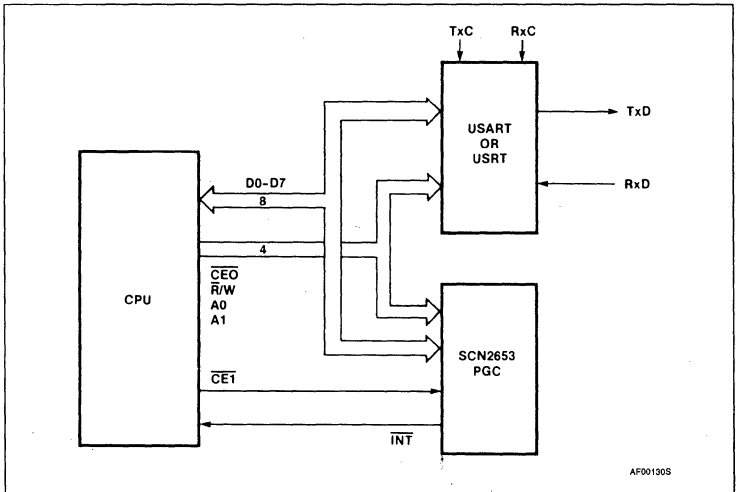


Figure 1. Typical System Configuration

Character Class Array

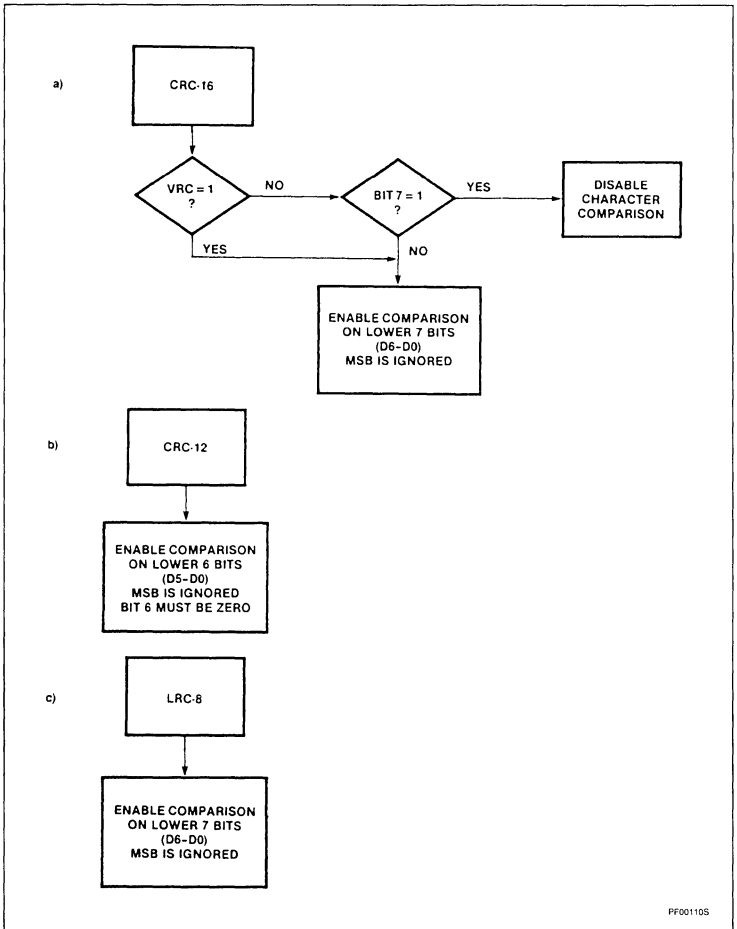
This 128 x 2 array holds the character class associated with each of 128 possible 7-bit characters. The array is zero after a master reset. When the character class array is loaded (see PGC Addressing), the character on the data bus is placed in the class specified by the contents of command register bits CR2 and CR3. The PGC uses these two command bits to represent four different character classes. These are:

1. Normal class (included in the accumulation)
2. SYN character/BISYNC not included class
3. Block terminating character/search class
4. Second search character class (preceded by DLE)

These encoded character classes are used by the PGC:

1. To control the BCC accumulation of associated characters in BISYNC modes only. BCC accumulation in automatic or single accumulation modes is carried out independent of the character classes.
2. To detect characters and two character sequences in all modes of accumulation and to set the control character detect bits in the status register.

It should be noted that any number of characters (up to 128 for CRC-16 or LRC-8; up to 64 for CRC-12) can be put into any one class.



NOTE:
Transparent mode always disables the SSC comparison.

Figure 2. Character Class Array Comparison Operation

If VRC is specified along with CRC-16 then the least significant 7 bits of the character are used for character array comparison. If VRC is not enabled, but CRC-16 is, the MSB of the character then determines whether a character comparison is to take place. If the MSB is 0, the comparison takes place; if the MSB is 1, the comparison does not take place and the character is processed as though it were in the normal class. This enables the PGC to detect all communication control characters and DLE-SSC sequences.

Only the first 64 locations of the array are accessed if CRC-12 is selected. The user should right justify each six bit character (D0-D5) to be written into the character class array. Bit 6 must be zero.

If VRC is enabled, the generated parity becomes the most significant bit of the character to be compared. VRC is not allowed in BISYNC transparent mode.

The method in which the character register contents is compared against the character class array depends on the BCC polynomial chosen. Figure 2 illustrates the comparison process.

DLE Read Only Memory

The DLE characters are stored internally and are selected by the error polynomial as follows:

CRC-12: 01 1111

LCR-8 or CRC-16:
No VRC or odd VRC: 0001 0000
Even VRC: 1001 0000

BCC and Parity Generator

This functional block performs all the necessary computation to generate and update the BCC accumulation on a character by character busis. It contains the three generator polynomials (CRC-16, CRC-12, and LRC-8) that can be selected to compute the BCC. This block also checks and generates odd or even parity for 7-bit (ASCII) characters.

BCC Registers

This block consists of two 8-bit registers (BCC upper and BCC lower) which contain the high and low order bytes of the BCC accumulation. The result of the accumulation from the BCC and parity generator is stored in these registers. A recirculating register address pointer is initialized by a power on, master reset, or clear BCC command. The pointer alternately selects BCC upper and lower on successive BCC register accesses for CRC-16 or CRC-12. For LRC-8, BCC upper is always selected.

BCC upper and lower are cleared by a clear BCC or master reset command. The highest term of the BCC polynomial is always represented by bit 0 of BCC upper; the lowest term is always represented by bit 7 of BCC lower (see figure 3, Orientation of BCC Polynomials.)

The length of the block check character depends on the error checking polynomial that is selected. If LRC-8 is chosen, the BCC result is stored entirely in BCC upper. The BCC lower remains unchanged from previous setting. Both BCC registers are used when CRC-16 is specified. When CRC-12 is selected, the block check character is 12 bits long. The six least significant bits of the BCC are stored in the least significant bits of the BCC lower. The remaining upper six bits of the BCC are stored in least significant bits of BCC upper. The two most significant bits in each BCC register are filled with zero.

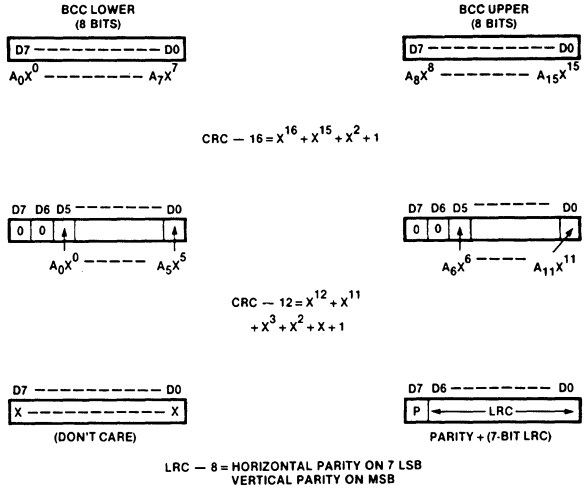
The BCC register(s) are read by the CPU after the last data character is transmitted. They can then be sent to the R/T to complete a transmitted block of data. These registers are read and loaded when one PGC is time-shared by several R/Ts. Refer to Applications Information—Multiplexed PGC.

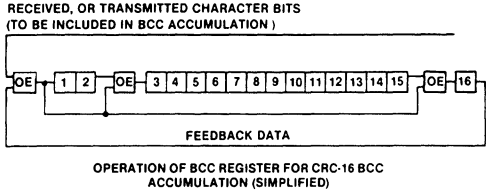
PGC Addressing

All internal registers and the character class array are selected by the unique address codes shown in table 2.

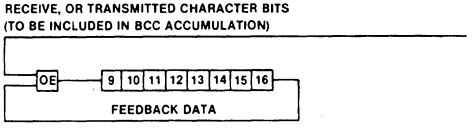
$$\frac{M(X)}{G(X)} = Q(X) + \frac{R(X)}{G(X)}, \text{ WHERE } R(X) = A_n X^n + A_{n-1}X^{n-1} \cdots + A_0 X^0$$

M (X) : BINARY POLYNOMIAL (DATA STREAM)
G (X) : FIXED DIVISOR TO GENERATE BCC
Q (X) : QUOTIENT AFTER BCC GENERATION
R (X) : REMAINDER AFTER BCC GENERATION





OPERATION OF BCC REGISTER FOR CRC-16 BCC ACCUMULATION (SIMPLIFIED)



OPERATION OF BCC REGISTER FOR LRC BCC ACCUMULATION (SIMPLIFIED)

AF00140S

Figure 3. Orientation of BCC Polynomials

Table 2. ADDRESS CODES

$\overline{CE0}$	$\overline{CE1}$	A1	A0	$\overline{R}/W$	FUNCTION
0	0	X	X	X	Operation not guaranteed
0	1	0	0	0	If MR2 = 0 load data bus into character register If MR2 = 1 PGC not selected[1]
0	1	0	0	1	If MR2 = 1 load data bus into character register If MR2 = 0 PGC not selected[1]
0	1	0	1	X	PGC not selected[1]
0	1	1	0	X	PGC not selected[1]
0	1	1	1	X	PGC not selected[1]
1	0	0	0	0	Read character register
1	0	0	0	1	Load data bus into character register if MR1,0 $\neq 00$[2]; write character class array using CR3, CR2 class code if MR 1,0 = 00[3,4]
1	0	0	1	0	Read Status register
1	0	0	1	1	Write command register
1	0	1	0	0	Read mode register
1	0	1	0	1	Write mode register
1	0	1	1	0	Read BCC upper/lower[5]
1	0	1	1	1	Write BCC upper/lower[5]
1	1	X	X	X	PGC not selected[1]

NOTES:
1. Data bus is 3-state
2. Character will not be accumulated unless MR3 = 1.
3. Character will not be accumulated even if MR3 = 1.
4. The mode bits MR1 and MR0 are cleared to 00 by power-on-reset, master reset, or by loading the mode register bits MR1 and MR0.
5. Recirculating internal pointer selects BCC upper on first access, BCC lower on next access for all BCCs except for LRC-8; in case of LRC-8, the pointer only selects BCC upper.

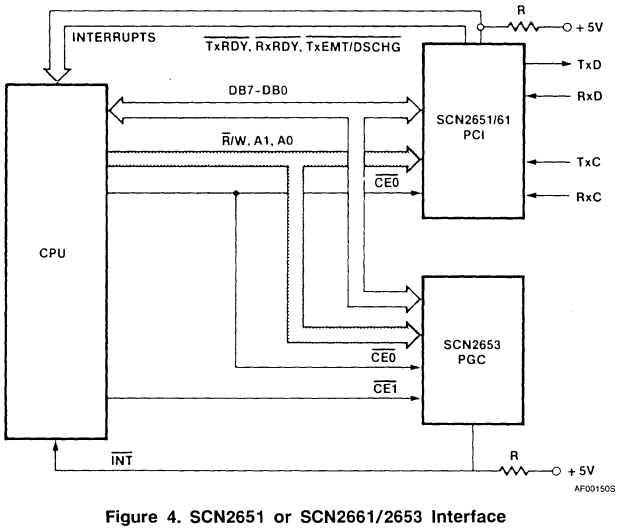


Figure 4. SCN2651 or SCN2661/2653 Interface

INTERFACE SIGNALS AND TIMING

PGC data transfers are controlled by A1, A0, and $\overline{R}/W$ which must be stable prior to the active low going chip enable pulse. $\overline{CE0}$ is used for PGC monitoring of data transfers between a CPU/DMA controller and a R/T; $\overline{CE1}$ is used for direct CPU-to-PGC transfers. MR3 must be set prior to loading the character register in order to accumulate or compare characters via $\overline{CE1}$. The active low (leading) edge of chip enable initiates a PGC read/write cycle; the rising (trailing) edge ends the cycle and also serves as a write strobe.

When loading the character, mode, or command register, the data bus is strobed into the selected register on the trailing (rising) edge of the appropriate $\overline{CE}$. When writing into the character class array, the data on the bus (the special character) is placed in the class specified by command register bits CR3 and CR2.

Characters are transferred into the character register when $\overline{CE0}$ is active (low) depending on the state of MR2 and the $\overline{R}/W$ input. Characters (from the R/T are loaded into the character register when in receive mode (MR2 = 0 and $\overline{R}/W = 0$) while CPU/DMA characters are loaded into the character register when in transmit mode (MR2 = 1 and $\overline{R}/W = 1$). The time between consecutive chip enables is given by t_{CEC} or t_{CED}.

The open drain active low interrupt signal ($\overline{INT}$) goes active whenever one or more of four maskable status conditions (SR0-SR3) are true (= 1). A status read deactivates $\overline{INT}$.

The same techniques used in interfacing the SCN2651 PCI to 8-bit microprocessors can be used to interface the SCN2653 PGC (consult Application Note M22). Note that when addressing the R/T's holding registers, the PGC pins must have A1, A0 = 00 and that the address and $\overline{R}/W$ signals must be stable (set up) prior to the active low chip enable. When using the SCN2651 or SCN2661 as the R/T, the PGC's A1, A0, $\overline{R}/W$, and $\overline{CE0}$ are directly connected to comparable SCN2651 or SCN2661 signals. Schematics of an SCN2653 monitoring data transfers to/from the Signetics SCN2651/2661 and SCN2652 are shown in figures 4 and 5.

An alternate interfacing technique is to treat the PGC as an independent peripheral device. This necessitates a write character register instruction after the CPU reads or writes a character to or from the R/T.

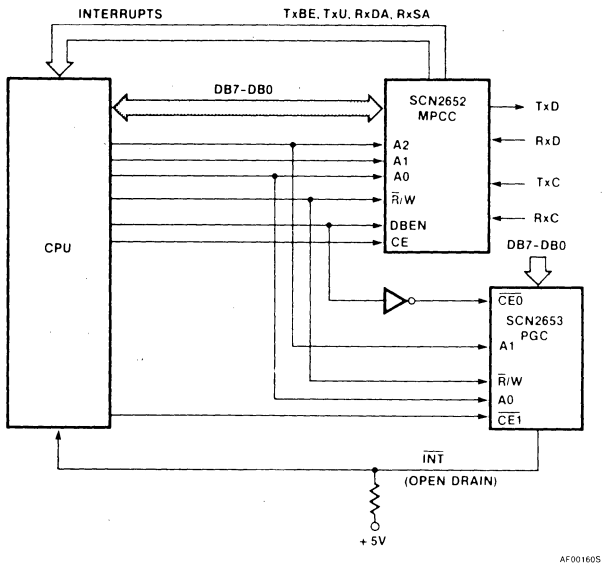


Figure 5. SCN2652/2653 Interface – Typical Protocols: *BISYNC, DDCMP, SDLC, HDLC

PGC PROGRAMMING

The PGC operational mode must be initially programmed by the CPU (see figure 6). The mode register, command register and character class array should be written into, after a power-on-reset or a master reset command. The character class array should be programmed only for the classes pertinent to the application. After a master reset, the character class array is zero which places all characters in the normal class (included in the BCC accumulation).

OPERATION

The PGC should be initially configured by the CPU (via $\overline{CE1}$) prior to systems operation. This is done by loading the mode register, command register and character class array (see PGC PROGRAMMING). Characters may then be loaded into the character register for BCC accumulation, VRC generation/checking, BTC/SC and DLE-SSC comparisons. See table 3 for a summary of BCC accumulation modes.

BCC accumulation depends on the mode selected.

BISYNC Normal

In BISYNC normal mode, all characters loaded into the character register are accumulated except those in the SYN/BISYNC not included class. During receive (MR2 = 0), a BTC/SC match will cause the BCC accumulation to stop after the next one (LRC-8) or two (CRC-12 or CRC-16) characters have been accumulated. At that time, if the BCC accumulation does not equal zero, the BCC error bit (SR0) will be set and $\overline{INT}$ will go active if the corresponding mask bit (CR4) is enabled (= 1). In transmit (MR2 = 1), the BCC accumulation is automatically stopped once the BTC/SC character has been accumulated. The CPU must read the BCC upper and BCC lower (CRC-12 or CRC-16) register(s) and transmit them to the R/T.

Note that the received BCCs are not subject to VRC if CRC-16 is selected. If LRC-8 is selected, the received BCC is subject to VRC. An incorrect result will set the VRC error bit (SR1). After its accumulation, the least significant 7 bits of BCC upper are checked and a non-zero result will set the BCC error bit (SR0). BCCs are not checked against the character class array nor are they compared to the DLE ROM.

Second search character (SSC) detection is enabled so that a DLE-STX or two character communication control sequence can be detected.

BISYNC Transparent

BISYNC transparent mode should be used for data blocks beginning with DLE-STX if the DLEs are transferred between CPU and R/T ($\overline{CE0}$) or CPU and PGC ($\overline{CE1}$), i.e., DLEs are not stripped. VRC should be disabled in this mode. Characters excluded from the BCC accumulation are the first DLE of a DLE-non SYN sequence pair and the DLE-SYN sequence if not preceded by an odd number of DLEs. For example, consider the following transparent mode character string:

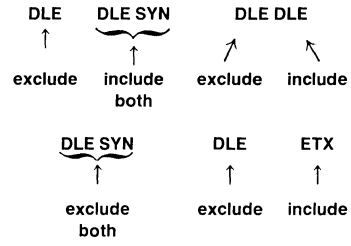


Table 3. SUMMARY OF BCC ACCUMULATION MODES

ACCUMULATION MODES	START ACCUMULATION	STOP ACCUMULATION	CHARACTERS EXCLUDED FROM ACCUMULATION
BISYNC normal and BISYNC transparent	Clear BCC registers command Mode register is loaded with BISYNC or automatic mode Start accumulation command Load BCC registers	After BTC has been detected and received BCC is accumulated After transmitted BTC has been accumulated Single mode is selected	SYN/BISYNC not included class in normal mode DLE-SYN/not included class and first DLE of a DLE non SYN pair in transparent mode. These characters are not excluded if preceded by an odd number of DLEs
Automatic	Same as above	Single mode selected	None
Single	Start accumulation command	After each character has been accumulated	Up to user who must generate start accumulation command for each character to be included

In receive and transmit modes, the termination of BCC accumulation works exactly as in BISYNC normal, except that the BTC/SC must be immediately preceded by an odd number of DLEs to be identified as a BTC/SC.

Second search character detection is not enabled in BISYNC transparent.

After a BTC/SC class character is detected by the PGC when receiving in either BISYNC mode, the following one or two characters are accumulated (depending on LRC-8 or CRC-12/16, respectively) and the PGC will automatically stop further accumulation. However, the PGC can continue the accumulation if a start accumulate command is issued or either BISYNC mode is loaded into the mode register. The start accumulate command should be given to the PGC before loading the character that follows the detected BTC/SC. This procedure enables a special search character to be detected (the BTC/SC detect bit (SR2) will be set and an interrupt generated if CR6 = 1) with the BCC accumulation continuing (see figures 7 and 8).

Automatic Accumulate

All characters loaded into the character register are accumulated, BTC/SC and SSC detection is enabled. The BCC accumulation is not automatically terminated. (The CPU must use single accumulate mode to stop the accumulation). When in receive mode, the BCC error bit (SR0) is set/reset after accumulating each character so that the CPU must examine this bit after the last character is accumulated. SR0 = 0 if the accumulated remainder in the BCC register(s) is zero; otherwise SR0 = 1. Examples of use of automatic accumulate mode usage include an R/T (SCN2651/SCN2661) in transparent DLE/SYN strip mode and asynchronous/synchronous/parallel DDCMP.

Single Accumulate

All characters for which a start accumulate command (CR1, CR0 = 01) is given are accumulated and compared against the character class array. If not given, the BCC accumulation is not updated and BTC/SC and SCC detection is disabled. Operation in this mode is otherwise identical to automatic accumulate.

Single accumulate mode can be used to selectively accumulate characters under CPU control or to accumulate characters that were unintentionally excluded in one of the other modes.

Polynomial Selection and DLE Comparison

The BCC polynomial may be CRC-16, CRC-12 or LRC-8. The cyclic redundancy check (CRC) is generated by dividing the binary value of a character in the character register by the selected polynomial. The quotient is discarded and the remainder is used as the BCC (two 6-bit characters for CRC-12, two 8-bit characters for CRC-16). CRC-16 uses all 8 bits of each BCC register. CRC-12 uses the least significant 6 bits of the BCC registers. The two most significant bits of the BCC registers are cleared to zero whenever CRC-12 is selected (see figure 3).

When the PGC is in receive mode (MR2 = 0), the received BCC will be accumulated. The result will be zero for an error free message.

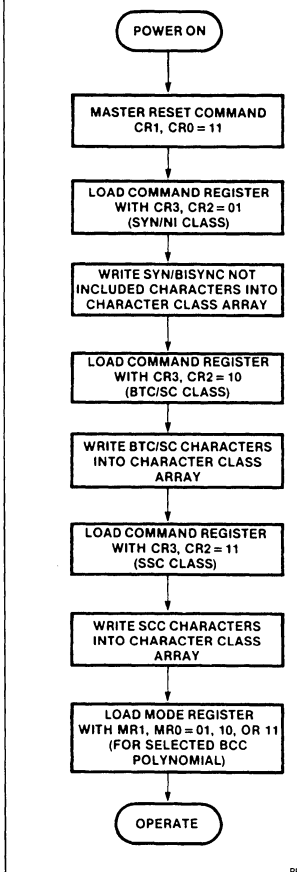
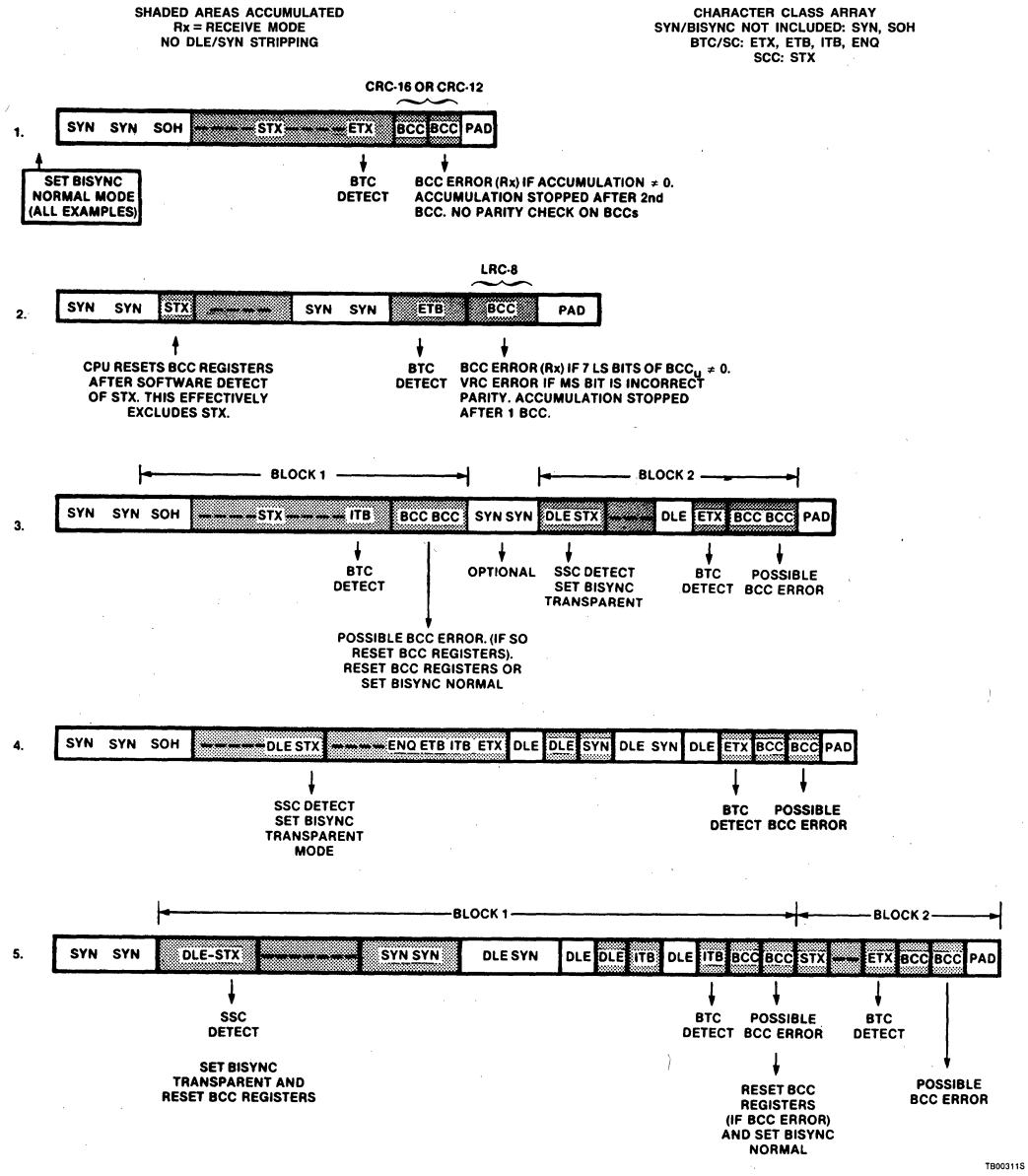
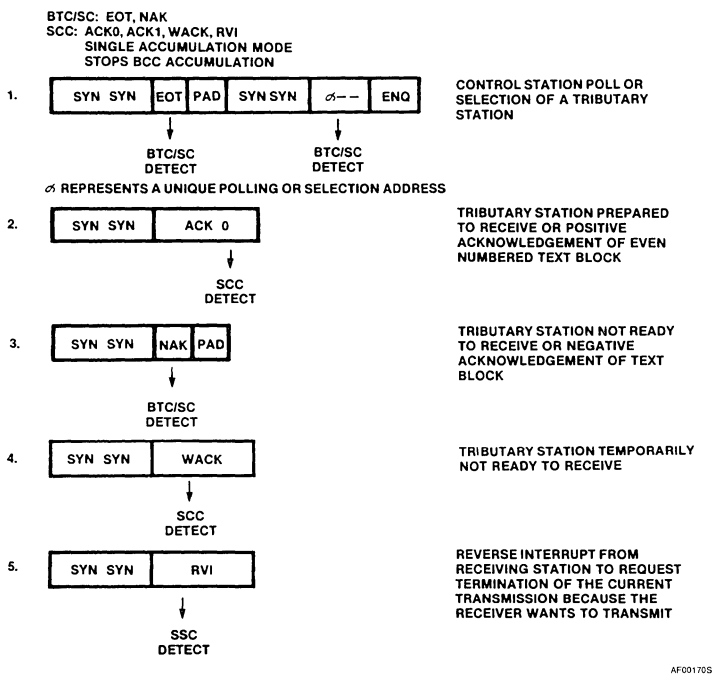


Figure 6. PGC Programming



NOTES:
1. BCC error only for receive mode. In transmit mode, CPU must respond to BTC detect by reading the BCC register(s) and sending them to the R/T. The accumulation is stopped after the BTC is accumulated.
2. ENQ (DLE – ENQ) in a text message should be treated as an abort.
3. Opening SYNs may be stripped by the R/T.
4. The single accumulate mode and command can be used to accumulate a character that inadvertently was excluded. (For example, the DLE of a DLE – STX if the PGC was in transparent mode and there was not a line turnaround prior to the DLE). The single accumulation should be done using $\overline{CE1}$ after the BCC(s) have been accumulated.

Figure 7. Examples – Bisync Text Message BCC Accumulation



NOTES:
1. BCC accumulation should be ignored for control messages. This can be effected by single accumulate mode without single accumulate commands.
2. Characters programmed as SSCs should be the binary equivalent of the second character of the DLE – SSC sequence.

Figure 8. Examples – Bisync Control Messages

CRC-12 is used with 6-bit codes. The internal 6-bit transcode DLE character hex 1F is selected by CRC-12. VRC should be disabled (MR4 = 0) for CRC-12 operation. The two most significant bits of the character register are ignored when compared to the internal 6-bit DLE. When the character is checked against the character class array, the MSB is ignored and the next MSB (bit 6) is assumed to be zero. If CRC-12 is specified, the user must write to the character class array with bit 6 cleared.

CRC-16 or LRC-8 implies the use of ASCII or EBCDIC although any 7-bit plus parity or 8-bit no parity code may be used (with DLE = hex 10 or hex 90). The DLE character compare is on an 8-bit basis with the generated parity (if VRC is enabled) as the MSB. When the character is compared against the character class array, the MSB is not used. This may result in a false BTC or SSC detection if there is a VRC error. However, the VRC error bit (SR1) will be set under that condition.

The LRC-8 is generated by the exclusive OR of the 7 least significant bits of the character register and the BCC upper. The most significant bit of the LRC-8 check character is a vertical odd/even parity bit (MR5 = 0/1), which is generated on the least significant bits of that character. The selection of LRC-8 implies VRC is enabled and that only the BCC upper is used for the BCC accumulation. The BCC lower remains unchanged from previous setting.

VRC Generation and Detection

Parity (VRC) is enabled by MR4 and specified as odd or even by MR5. VRC should be disabled when in BISYNC transparent mode and whenever CRC-12 or CRC-16 (EBCDIC) is selected as the BCC polynomial. MR4 = 1 enables VRC generation and detection for both receive and transmit operations. Characters loaded into the character register will have VRC generated on the least significant 7 bits with the generated parity bit written into the character register MSB. If the generated parity does not match the MSB of the loaded character, the VRC error bit (SR1) is set and $\overline{INT}$ asserted if the corresponding mask bit was enabled (CR5 = 1). Thus, if 7-bit characters are to be transmitted with VRC, CR5 should be zero and SR1 ignored. 8-bit characters with a VRC bit in the MSB position are parity checked by the PGC in both transmit (to R/T) and receive (from R/T) modes, i.e., the PGC operates as a data bus parity checker.

2

CHARACTER CLASSES

Normal (Included in the Accumulation)

Any character that belongs to this class is normal data, i.e., the character is not a communication control or other special character. Characters in this class are always accumulated in BISYNC, automatic and single accumulation modes.

SYN Character/BISYNC Not Included

SYN characters are never accumulated in BISYNC normal accumulation mode. In BISYNC transparent accumulation mode, the DLE-SYN character pair is not accumulated, but a SYN not preceded by a DLE is accumulated. (DLE is implied as an odd number of DLEs).

Block Termination Character (BTC)/Search Character (SC)

BTC/SC characters have two functions in the PGC: termination of BCC accumulation and character detection. In BISYNC transparent mode, a BTC/SC must be preceded by an odd number of DLEs to be recognized.

Termination of BCC Accumulation

In BISYNC normal and transparent accumulation modes, the PGC will stop the accumulation upon the detection of the BTC/SC character. Examples of BTCs are ETX, ETB, ITB, ENQ.

In receive mode, the accumulation is stopped after the following one (LRC-8) or two (CRC-12, CRC-16) character(s) have been accumulated. In transmit mode, the accumulation is stopped after the BTC/SC character has been accumulated. The BTC/SC character is always accumulated in all of the accumulation modes.

Character Detection

BTC/SC characters will be detected in any of the four accumulation modes when that character is being accumulated. The BTC/SC status bit (SR2) is set on detection. Since detection also stops BISYNC BCC accumulation, the BISYNC accumulation must be restarted if the character is not a BTC. This can be effected by loading BISYNC mode into the mode register or generating a start accumulation command.

Second Search Character Class (SSC)

Control functions in character oriented data link control procedures can be represented by a sequence of two characters, the first character being a DLE. Examples include ACK0, ACK1, WACK, RVI, DISC, WBT and the initiation of transparent text (DLE-STX). The PGC will detect such sequences, except in BISYNC transparent mode, when an SSC class character is being accumulated after being immediately preceded by an odd number of DLEs. Under those conditions, the SSC status bit (SR3) will be set.

The SSC character is always accumulated in all of the accumulation modes.

REGISTER BIT DESCRIPTION

The operation of the PGC is determined by programming the mode register and the command register. The status register provides feedback on potential interrupt conditions. Formats of these registers are shown in table 4.

Table 4. PGC REGISTER BIT FORMATS

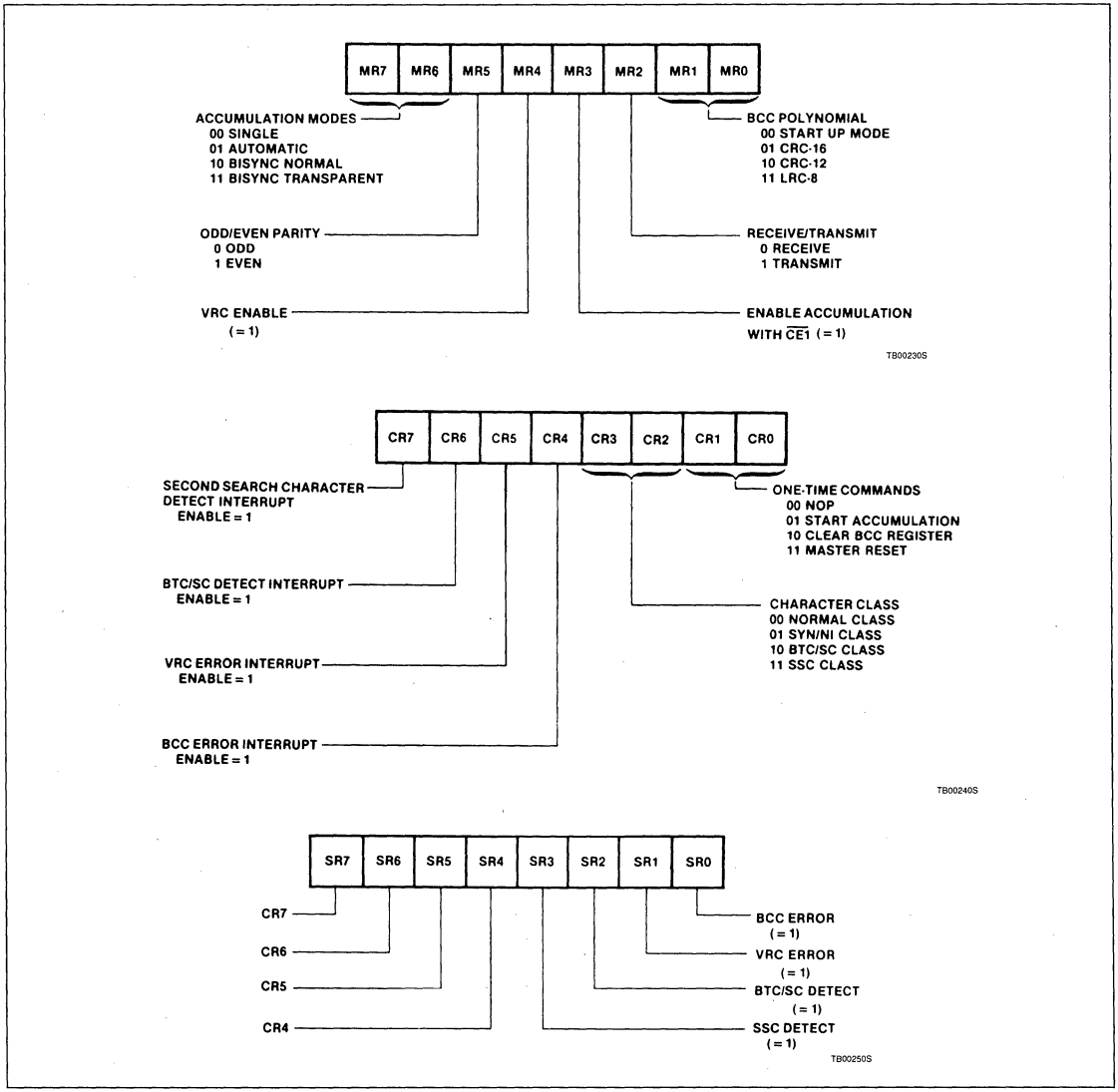


Table 5. BCC ACCUMULATION BY CHARACTER CLASS

CR3	CR2	CLASS	BISYNC NORMAL	BISYNC TRANSPARENT	AUTOMATIC ACCUM	SINGLE ACCUM
0	0	Normal	Yes	Yes	Yes	Yes
0	1	SYN/BISYNC not included	No	Yes, unless preceded by an odd number of DLEs	Yes	Yes
1	0	BTC/SC	Yes	Yes	Yes	Yes
1	1	SSC*	Yes	Yes	Yes	Yes

NOTE:
*Preceded by DLE

Mode Register

The mode register defines general PGC operation characteristics. MR1 and MRO = 00 permit the character class array to be programmed. These bits will be zero after a power on or master reset command. After the character class array is programmed, these bits should be set to 01,10, or 11 to select the CRC-16, CRC-12 or LRC-8 polynomials.

MR2(Tx/$\overline{Rx}$) determines whether or not the PGC is to generate (Tx) or generate and check (Rx) the BCC. It is used with $\overline{R}$/W to determine if the data bus is to be loaded into the character register when $\overline{CE0}$, $\overline{CE1}$, A1, A0 = 0100.

If MR2 = 1: 1) the PGC will generate the BCC but will never set the BCC error bit (SR0). 2) If the $\overline{R}$/W pin is high when $\overline{CE0}$, $\overline{CE1}$, A1, A0 = 0100, then the data bus will be loaded into the character register. If $\overline{R}$/W is low under these conditions, the PGC is not selected.

If MR2 = 0: 1) the PGC will accumulate the BCC and set the BCC error bit (SR0) when appropriate. 2) If the $\overline{R}$/W pin is low when $\overline{CE0}$, $\overline{CE1}$, A1, A0 = 0100, then the data bus will be loaded into the character register. If $\overline{R}$/W is high under these conditions, the PGC is not selected.

MR3 is a $\overline{CE1}$ accumulate/compare enable bit. If MR3 = 0, characters loaded into the character register by $\overline{CE1}$ are not accumulated, checked against the character class array, or compared to the DLE ROM. Parity will be generated and checked if VRC is enabled (MR4 = 1). The primary use of MR3 = 0 is to generate parity on a 7-bit character which is to be transmitted to an R/T. The CPU loads the character register with the 7-bit character and reads the 8-bit VRC generated character via $\overline{CE1}$. This 8-bit character is then transferred to the R/T via $\overline{CE0}$. Another application of MR3 = 0 is for a CPU to interleave parity checking on memory data ($\overline{CE1}$) with on line R/T data transfer ($\overline{CE0}$).

If MR3 = 1, characters loaded into the character register by $\overline{CE1}$ will be accumulated (according to the BCC accumulation mode selected) and compared against the character class array and DLE ROM. This bit setting should be used when the CPU/DMA controller sends data characters to be accumulated or compared to the PGC and the R/T is inactive (off line). If the R/T were active, then a DLE or BTC loaded into the character register via $\overline{CE0}$ would cause incorrect accumulation and character comparisons if the next character was loaded via $\overline{CE1}$.

Table 6. BTC/SC AND SSC DETECTION CONDITIONS

CLASS	BISYNC NORMAL	BISYNC TRANSPARENT	AUTO ACCUM	SINGLE ACCUM
BTC/SC	Yes	Yes*	Yes	Yes †
SSC	Yes*	No	Yes*	Yes*†

NOTES:
* Only if immediately preceded by an odd number of DLEs.
† Start accumulate command necessary for detection.

MR4 is a VRC enable bit. If MR4 = 1, VRC is enabled as odd/even by MR5. VRC is generated on the 7 LS bits of the character and the MS bit is checked against the generated parity. If not equal, SR1 is set. If MR4 = 0, VRC is not enabled. MR4 = 0 is used for BISYNC transparent mode with ASCII code, and for both BISYNC modes for EBCDIC and SBT.

MR5 is an odd/even VRC bit. If MR5 = 1, the total number of 1 bits in the character including the parity bit is even. If MR5 = 0, the total number of bits is odd. This bit is ignored if MR4 = 0.

MR7, MR6 select the BCC accumulation mode. These modes have been previously discussed in the operation section.

Command Register

The command register contains four interrupt enables, a 2-bit character class code used when programming the character class array, and 2 bits that specify three one time commands and a NOP.

CR1, CR0 = 00 is a NOP. This bit setting is used when changing CR7-CR2 without affecting any of the 3 one time commands.

CR1, CR0 = 01 is a start BCC accumulation command. In single accumulation mode, the character accumulated is the character that is in the character register at the time the command is given. The accumulation stops immediately after the character has been accumulated. If the command is given in either of the BISYNC or automatic accumulation modes, it enables the PGC to accumulate the BCC starting with the next character loaded into the character register. This is a means of restarting a BISYNC normal accumulation after detection of a BTC/SC that is not a valid BTC (example; CR, LF, TAB). In all accumulation modes, a previously detected DLE will not be cancelled by this command.

CR1, CR0 = 10 is a clear BCC registers command. Both BCC registers are cleared along with the associated internal pointer and SR0-SR3. The pointer points to BCC upper. $\overline{INT}$ is forced high. This command permits BCC accumulation, starting with the next character loaded into the character register in BISYNC or auto modes. Single accumulate mode requires a start BCC accumulation command.

CR1, CR0 = 11 is a master reset command. All internal registers (except the character register), the internal pointer, and the entire character class array are cleared. $\overline{INT}$ is forced high.

CR3 and CR2 are used for programming the character class array. During a write character class array instruction, the character corresponding to the 7 LS bits of the data bus is placed in the class contained in CR3 and CR2. The encoded character classes control the accumulation of the associated character as shown in table 5.

Detection operates under the conditions shown in table 6.

CR7, CR6, CR5, CR4 are interrupt enables that individually enable/disable $\overline{INT}$ when the corresponding status register condition is true (set). Each bit is set in order to enable $\overline{INT}$ upon the condition. Each bit is reset to disable $\overline{INT}$ upon the condition. The state of

these bits may be read via the status register (SR7, SR6, SR5, SR4).

The corresponding status bits (SR3, SR2, SR1, SR0) are set independent of the interrupt enables. The bit assignments are:

CR4–BCC error interrupt enable

CR5–VRC error interrupt enable

CR6–BTC/SC detect interrupt enable

CR7–DLE-SSC detect interrupt enable

Status Register

This register reflects the status of the 4 conditions that are potential interrupt ($\overline{INT}$) sources and the 4 interrupt enables in the command register. A status register read clears SR0, SR1, SR2, SR3 and deactivates $\overline{INT}$. These bits are also cleared by a master reset or clear BCC command.

SR0 is a BCC error bit. This bit can only be set in receive mode (MR2 = 0). In BISYNC normal and BISYNC transparent modes, SR0 will be set/reset once the accumulation has been stopped by the detection of the BTC/SC character and accumulation of the BCC(s).

In automatic and single accumulate modes, SR0 is set/reset after each character in the character register has been accumulated.

The rules for the detection of a BCC error are:

SR0 = 1 LRC-8: 7 least significant bits of BCC upper $\neq 0$
CRC-12, CRC-16: BCC upper or BCC lower $\neq 0$

SR0 = 0 LRC-8: 7 least significant bits of BCC upper = 0
CRC-12, CRC-16: BCC upper and BCC lower = 0

SR1 is a VRC error bit. When set, this bit reports a character parity error (on receive or transmit) when parity is enabled (MR4 = 1). Parity is odd/even as specified by MR5. The parity bit will be regenerated in the character register.

SR2 is a BTC/SC detect bit. When set, this bit indicates the character being accumulated is of the BTC/SC class for BISYNC normal, automatic and single accumulate modes. In BISYNC transparent mode, the BTC/SC character being accumulated must be immediately preceded by an odd number of DLEs for this bit to be set.

SR3 is a DLE SSC detect bit. This bit can only be set when in BISYNC normal, auto, or single accumulated modes. When set, it indicates that the character being accumulated is of the SSC class when that character was immediately preceded by an odd number of DLEs.

SR7, SR6, SR5, SR4 are interrupt enables. These 4 bits reflect the state of the interrupt enable command bits CR7, CR6, CR5, and CR4, as follows:

SR4–BCC error

SR5–VRC error

SR6–BTC/SC detect

SR7–SSC detect

APPLICATIONS INFORMATION

Dedicated PGC

The most efficient use of the 2653 is to dedicate one to each R/T for two way alternate (half duplex) operation or two to each R/T for two way simultaneous (full duplex) operation (see figure 9). The CPU configures each PGC (using $\overline{CE1}$) by initializing the the mode register, command register, and character class array. Data transfers to or from the R/T can then be on a DMA basis with each receiver holding register ready signal used as a read request (RREQ) and each transmit holding register available signal used as a write request (WREQ) to the DMA controller. The CPU needs only to respond to enable interrupts from each of the PGCs. The individual $\overline{INT}$ outputs can be wire-OR'd into single CPU interrupt ($\overline{INTRPT}$) with one pull up resistor. Each PGC in this system has a unique address that is decoded into the respective chip enables.

The CPU or DMA controller could send a block of memory data to the PGC to be error checked without sending that data to the R/T. In that case, $\overline{CE1}$ is used.

Multiplexed PGC

One PGC may be time-shared among a few R/Ts if the CPU saves and restores the mode register and partial BCC result in the BCC registers. These registers are accessed via $\overline{CE1}$. There must be a separate save area for each R/T (serial channel) and a channel pointer indicating the last R/T that transferred or received a data character (see figure 10).

The loading of the BCC registers will clear SR0-SR3 and all previously detected special characters, i.e., DLE, BTC/SC, BCC (BISYNC modes). The BCC accumulation will start again when the next character is loaded into the character register in all accumulation modes except single. That mode requires a start accumulation command.

Figures 11 and 12 represent software flow diagrams for transmit and receive service requests. Note that interrupts from all other R/Ts must be masked during a read or write to the BCC registers so as not to affect the internal BCC address pointer. It is recommended that all R/T interrupts be masked while servicing an interrupt that accesses any PGC register.

BISYNC Operation

Table 7 is a concise listing of SCN2651/SCN2661 operating modes with recommended corresponding SCN2653 BCC accumulation modes.

Character Comparator

The PGC can be used as a programmable data bus character comparator which monitors data bus transfers (CPU $\leftrightarrow$ peripheral, CPU $\leftrightarrow$ CPU, CPU $\leftrightarrow$ memory, memory $\leftrightarrow$ peripheral (via DMA)). The user selectively loads the character class array with BTC/SC and SSC characters to be compared. Status bits will be set and an interrupt can be generated upon SC and DLE–SSC detection. A match on one to 128 different characters or DLE–SSC sequences can be programmed.

Figure 13 depicts an arrangement where the DMA controller or slave CPU handles data bus transfers, the PGC interrogates the data bus, and the host CPU responds to PGC interrupts.

2

Table 7. BISYNC (ANSI 3.28, ISO 1745) MODES FOR SCN2651/SCN2661 AND SCN2653

SCN2651/SCN2661 OPERATING MODES	SCN2653 BCC ACCUMULATION MODE
Sync normal non-strip	BISYNC normal
Sync transparent non-strip	BISYNC transparent
Normal SYN/DLE strip[1]	BISYNC normal
Transparent SYN/DLE strip[1]	Automatic accumulate[2]
Async (with SYN/DLE characters)	BISYNC normal

NOTES:
1. CPU should switch to non-strip mode after BTC detect. Otherwise a received BCC could be inadvertently stripped.
2. SSC detect should be ignored.

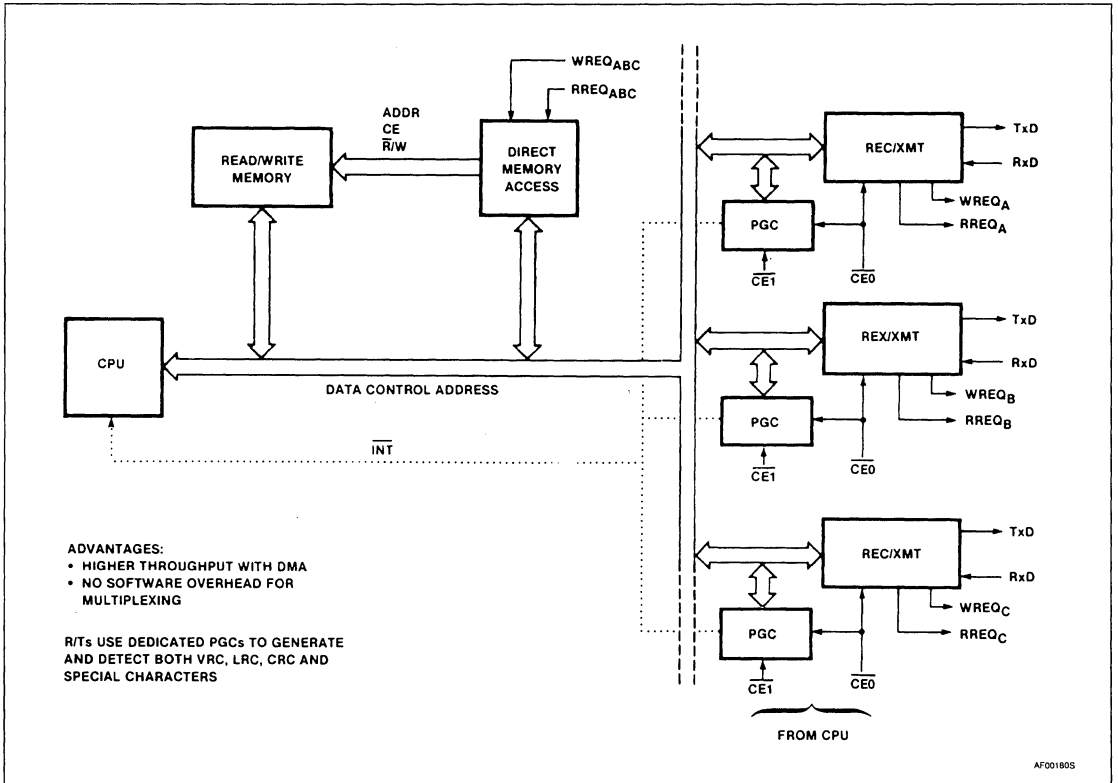


Figure 9. One PGC Per Receiver/Transmitter

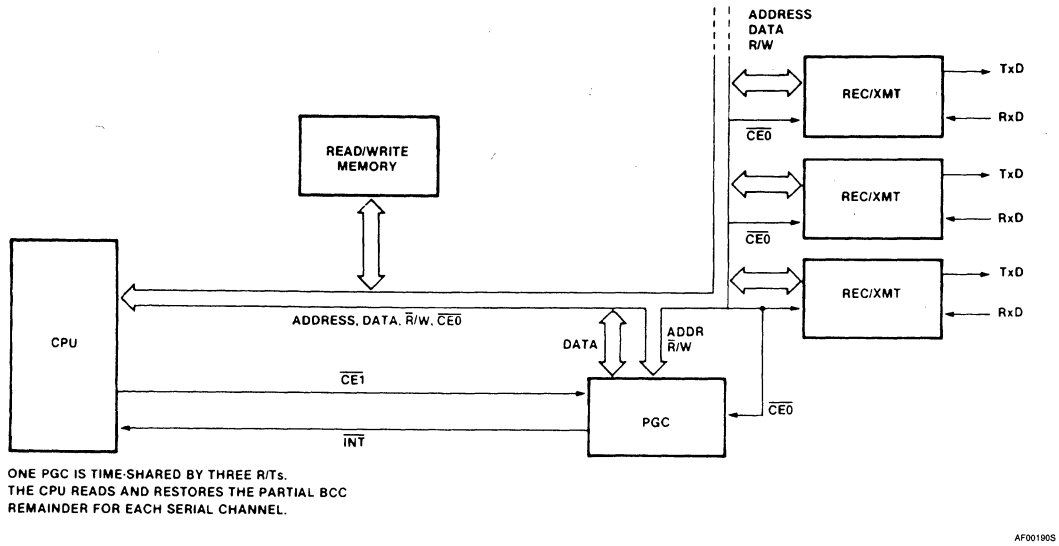


Figure 10. PGC Services Multiple Receivers/Transmitters

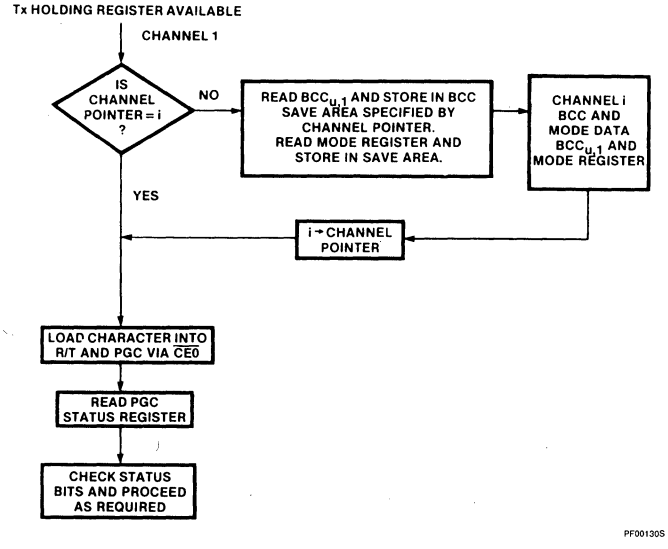


Figure 11. Multiplexed PGC—Transmit to R/T

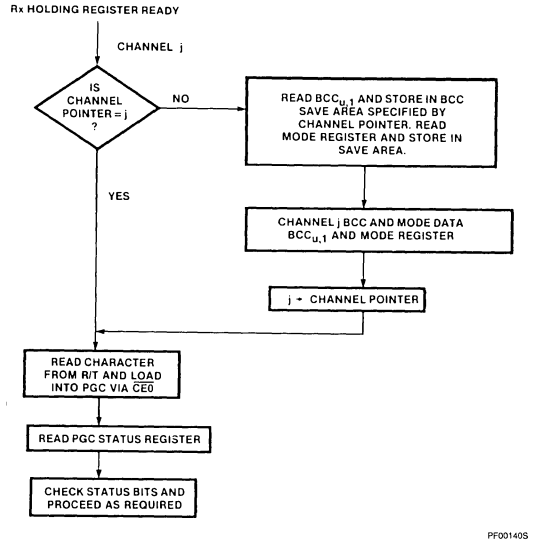
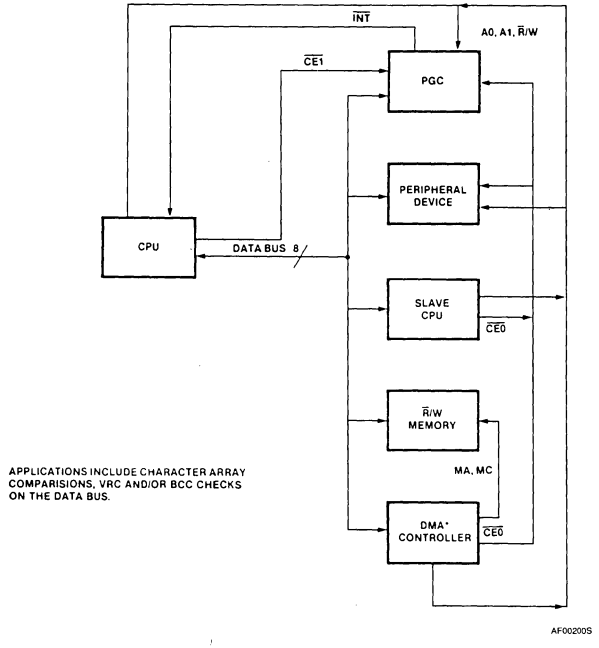


Figure 12. Multiplexed PGC—Receiver from R/T



NOTE:
* CPU initializes DMA controller for each block transfer of data.

Figure 13. PGC Data Bus Monitoring with DMA Transfers

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operation sections of this specification is not implied.
2. For operating at elevated temperatures the device must be derated based on + 150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. However, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to 70°C, $V_{CC} = 5.0V \pm 5\%$

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Input voltage						V
V_{IL}	Low				0.8	
V_{IH}	High		2.0			
Output voltage						V
V_{OL}	Low	$I_{OL} = 2.2mA$		0.25	0.45	
V_{OH}	High	$I_{OH} = -400\mu A$	2.4	2.8		
I_{IL}	Input load current	$V_{IN} = 0$ to 5.5V			10	μA
Output leakage current						μA
I_{LD}	Data bus	$V_{OUT} = 4.0V$			10	
I_{LO}	Open drain	$V_{OUT} = 4.0V$			10	
I_{CC}	Power supply current			45	75	mA

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to 70°C, $V_{CC} = 5.0V \pm 5\%$[1, 2, 3]

PARAMETER		LIMITS		UNIT
		Min	Max	
t_{CE}	Chip enable pulse width	250		ns
t_{CED}	Chip enable period D	1750		ns
t_{CEC}[4]	Chip enable period C	1750		ns
t_{AS}	Address set-up	10		ns
t_{AH}	Address hold	10		ns
t_{CS}	Control set-up	10		ns
t_{CH}	Control hold	10		ns
t_{DS}[5]	Data set-up	150		ns
t_{DH}	Data hold	15		ns
t_{DD}[6]	Data delay time for read		200	ns
t_{DF}[6]	Data bus floating time for read		100	ns
t_{INTL}[7]	Interrupt low delay		1600	ns
t_{INTH}[7]	Interrupt high delay		600	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. All voltage measurements are referenced to ground. All time measurements are at 50% level for inputs and at the 0.8V or 2.0V level for outputs. Input levels for testing are 0.45V and 2.4V.
3. Typical values are at +25°C, typical supply voltages and typical processing parameters.
4. $t_{CEC} = 600ns$ during PGC initialization when no BCC accumulation is in progress.
5. $t_{DS} = 50ns$ whenever $\overline{CE0}$ is used.
6. Test conditions: $C_L = 150pF$.
7. $\overline{INT}$ is an open drain output.

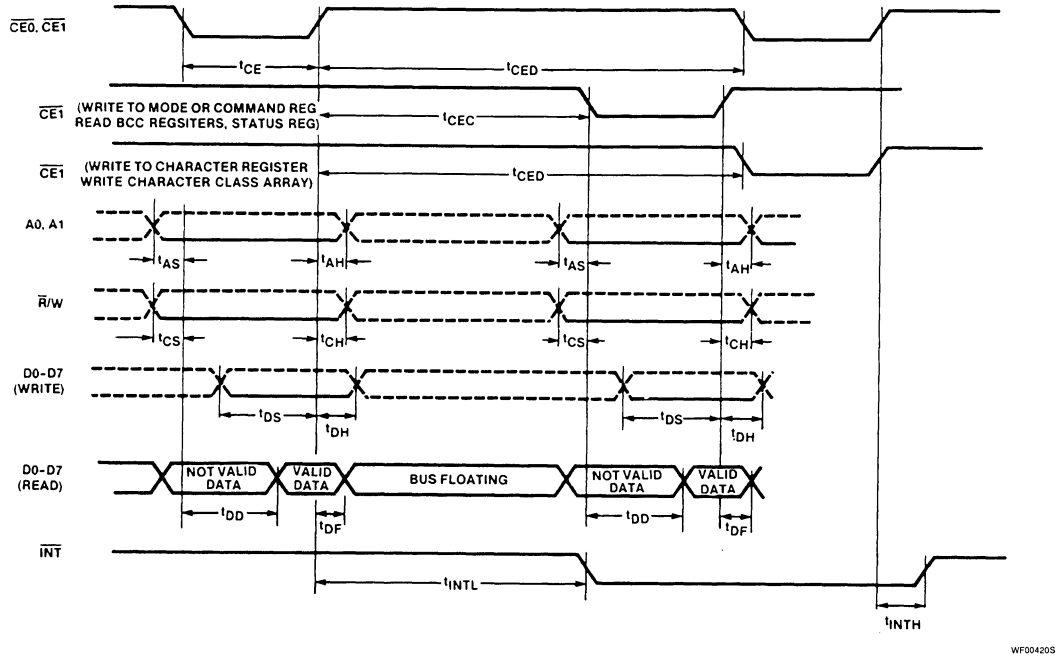


Figure 14. PGC Timing

Signetics

SCN2661/SCN68661
Enhanced Programmable Communications Interface (EPCI)

Microprocessor Products

Product Specification

DESCRIPTION

The Signetics SCN2661 EPCI is a universal synchronous/asynchronous data communications controller chip that is an enhanced version of the SCN2651. It interfaces easily to all 8-bit and 16-bit microprocessors and may be used in a polled or interrupt driven system environment. The SCN2661 accepts programmed instructions from the microprocessor while supporting many serial data communications disciplines – synchronous and asynchronous – in the full or half-duplex mode. Special support for BISYNC is provided.

The EPCI serializes parallel data characters received from the microprocessor for transmission. Simultaneously, it can receive serial data and convert it into parallel data characters for input to the microcomputer.

The SCN2661 contains a baud rate generator which can be programmed to either accept an external clock or to generate internal transmit or receive clocks. Sixteen different baud rates can be selected under program control when operating in the internal clock mode. Each version of the EPCI (A, B, C) has a different set of baud rates.

FEATURES

- **Synchronous operation**
  - **5- to 8-bit characters plus parity**
  - **Single or double SYN operation**
  - **Internal or external character synchronization**
  - **Transparent or non-transparent mode**
  - **Transparent mode DLE stuffing (Tx) and detection (Rx)**
  - **Automatic SYN or DLE-SYN insertion SYN, DLE and DLE-SYN stripping**
  - **Odd, even, or no parity**
  - **Local or remote maintenance loopback mode**
  - **Baud rate: dc to 1M bps (1X clock)**
- **Asynchronous operation**
  - **5- to 8-bit characters plus parity**
  - **1, 1½ or 2 stop bits transmitted**
  - **Odd, even, or no parity**
  - **Parity, overrun and framing error detection**
  - **Line break detection and generation**
  - **False start bit detection**
  - **Automatic serial echo mode (echoplex)**
  - **Local or remote maintenance loopback mode**
  - **Baud rate: dc to 1M bps (1X clock)**
    dc to 62.5K bps (16X clock)
    dc to 15.625K bps (64X clock)

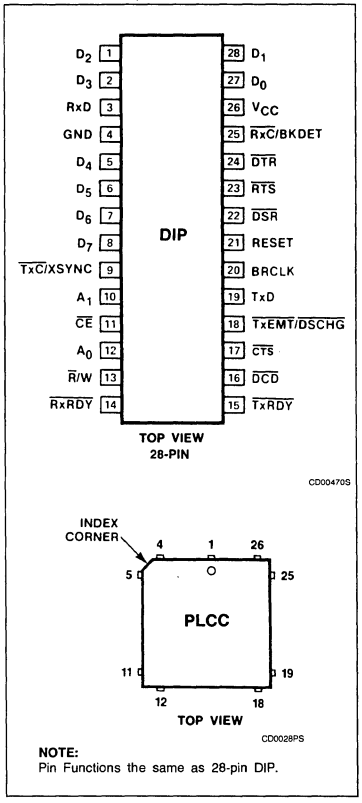
OTHER FEATURES

- **Internal or external baud rate clock**
- **3 baud rate sets**
- **16 internal rates for each set**
- **Double buffered transmitter and receiver**
- **Dynamic character length switching**
- **Full or half-duplex operation**
- **TTL compatible inputs and outputs**
- **RxC and TxC pins are short circuit protected**
- **Single 5V power supply**
- **No system clock required**

APPLICATIONS

- **Intelligent terminals**
- **Network processors**
- **Front-end processors**
- **Remote data concentrators**
- **Computer-to-computer links**
- **Serial peripherals**
- **BISYNC adaptors**

PIN CONFIGURATION



NOTE:
Pin Functions the same as 28-pin DIP.

Enhanced Programmable Communications Interface (EPCI)

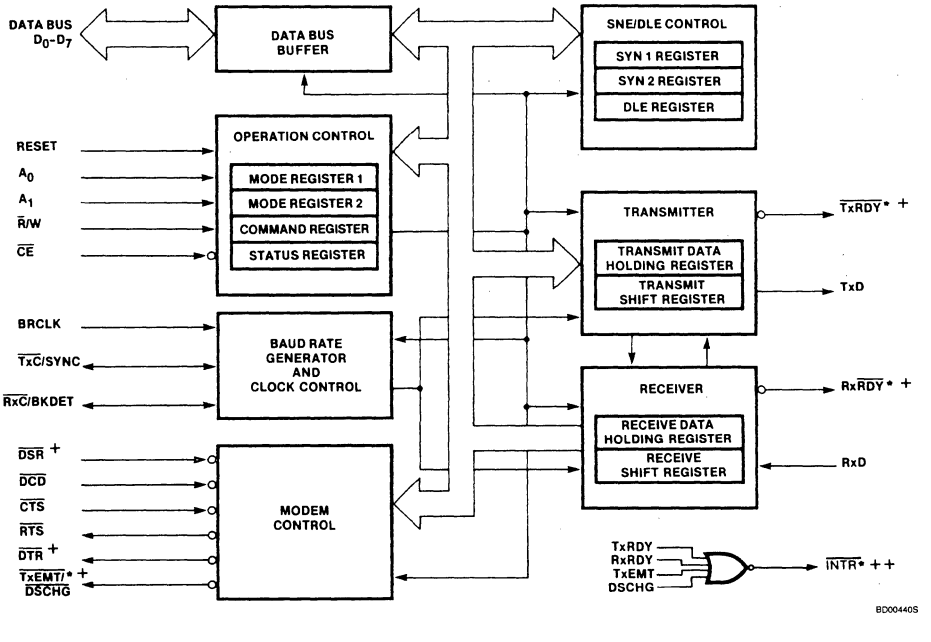
SCN2661/SCN68661

ORDERING CODE

PACKAGES	V_{CC} = 5V ±5%		
	Commercial	Automotive	Military
	0°C to +70°C	−40°C to +85°C	−55°C to +125°C
Ceramic DIP 28-Pin 0.6" Wide	SCN2661AC1I28 SCN2661BC1I28 SCN2661CC1I28	SCN2661AA1I28 SCN2661BA1I28 SCN2661CA1I28	SCN2661AM1I28 SCN2661BM1I28 SCN2661CM1I28
Plastic DIP 28-Pin 0.6" Wide	SCN2661AC1N28 SCN2661BC1N28 SCN2661CC1N28	Contact Factory	Not Available
Plastic LCC	SCN2661AC1A28 SCN2661BC1A28 SCN2661CC1A28	Contact Factory	Not Available

NOTES:
1. See table 1 for baud rates. Specify SCN2661A, B, or C depending on baud rate selected.
2. The SCN68661 is identical to the SCN2661. Order using part numbers above.

BLOCK DIAGRAM



NOTES:
* Open drain output pin.
\+ 28-pin only.
\+ + 24-pin only.

BLOCK DIAGRAM

The EPCI consists of six major sections. These are the transmitter, receiver, timing, operation control, modem control and SYN/DLE control. These sections communicate with each other via an internal data bus and an internal control bus. The internal data bus interfaces to the microprocessor data bus via a data bus buffer.

Operation Control

This functional block stores configuration and operation commands from the CPU and generates appropriate signals to various internal sections to control the overall device operation. It contains read and write circuits to permit communications with the microprocessor via the data bus and contains mode registers 1 and 2, the command register, and the status register. Details of register addressing and protocol are presented in the EPCI programming section of this data sheet.

Timing

The EPCI contains a baud rate generator (BRG) which is programmable to accept external transmit or receive clocks or to divide an external clock to perform data communications. The unit can generate 16 commonly used baud rates, any one of which can be selected for full-duplex operation. See table 1.

Receiver

The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU.

Transmitter

The transmitter accepts parallel data from the CPU, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin.

Modem Control

The modem control section provides interfacing for three input signals and three output signals used for "handshaking" and status indication between the CPU and a modem.

SYN/DLE Control

This section contains control circuitry and three 8-bit registers storing the SYN1, SYN2, and DLE characters provided by the CPU. These registers are used in the synchronous mode of operation to provide the characters required for synchronization, idle fill and data transparency.

Table 1. BAUD RATE GENERATOR CHARACTERISTICS

SCN2661A (BRCLK = 4.9152MHz)

MR23 – 20	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	50	0.8kHz	—	6144
0001	75	1.2	—	4096
0010	110	1.7598	-0.01	2793
0011	134.5	2.152	—	2284
0100	150	2.4	—	2048
0101	200	3.2	—	1536
0110	300	4.8	—	1024
0111	600	9.6	—	512
1000	1050	16.8329	0.196	292
1001	1200	19.2	—	256
1010	1800	28.7438	-0.19	171
1011	2000	31.9168	-0.26	154
1100	2400	38.4	—	128
1101	4800	76.8	—	64
1110	9600	153.6	—	32
1111	19200	307.2	—	16

SCN2661B (BRCLK = 4.9152MHz)

MR23 – 20	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	45.5	0.7279kHz	0.005	6752
0001	50	0.8	—	6144
0010	75	1.2	—	4096
0011	110	1.7598	-0.01	2793
0100	134.5	2.152	—	2284
0101	150	2.4	—	2048
0110	300	4.8	—	1024
0111	600	9.6	—	512
1000	1200	19.2	—	256
1001	1800	28.7438	-0.19	171
1010	2000	31.9168	-0.26	154
1011	2400	38.4	—	128
1100	4800	76.8	—	64
1101	9600	153.6	—	32
1110	19200	307.2	—	16
1111	38400	614.4	—	8

SCN2661C (BRCLK = 5.0688MHz)

MR23 – 20	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	50	0.8kHz	—	6336
0001	75	1.2	—	4224
0010	110	1.76	—	2880
0011	134.5	2.1523	0.016	2355
0100	150	2.4	—	2112
0101	300	4.8	—	1056
0110	600	9.6	—	528
0111	1200	19.2	—	264
1000	1800	28.8	—	176
1001	2000	32.081	0.253	158
1010	2400	38.4	—	132
1011	3600	57.6	—	88
1100	4800	76.8	—	66
1101	7200	115.2	—	44
1110	9600	153.6	—	33
1111	19200	316.8	3.125	16

NOTE:
16X clock is used in asynchronous mode. In synchronous mode, clock multiplier is 1X and BRG can be used only for TxC.

OPERATION

The functional operation of the SCN2661 is programmed by a set of control words supplied by the CPU. These control words specify items such as synchronous or asynchronous mode, baud rate, number of bits per character, etc. The programming procedure is described in the EPCI programming section of the data sheet.

After programming, the EPCI is ready to perform the desired communications functions. The receiver performs serial to parallel conversion of data received from a modem or equivalent device. The transmitter converts parallel data received from the CPU to a serial bit stream. These actions are accomplished within the framework specified by the control words.

Receiver

The SCN2661 is conditioned to receive data when the $\overline{DCD}$ input is low and the RxEN bit in the command register is true. In the asynchronous mode, the receiver looks for a high-to-low (mark to space) transition of the start bit on the RxD input line. If a transition is detected, the state of the RxD line is sampled again after a delay of one-half of a bit time. If RxD is now high, the search for a valid start bit is begun again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input line at one bit time intervals until the proper number of data bits, the parity bit, and one stop bit have been assembled. The data are then transferred to the receive data holding register, the RxRDY bit in the status register is set, and the $\overline{RxRDY}$ output is asserted. If the character length is less than 8 bits, the high order unused bits in the holding register are set to zero. The parity error, framing error, and overrun error status bits are strobed into the status register on the positive going edge of $\overline{RxC}$ corresponding to the received character boundary. If the stop bit is present, the receiver will immediately begin its search for the next start bit. If the stop bit is absent (framing error), the receiver will interpret a space as a start bit if it persists into the next bit time interval. If a break condition is detected (RxD is low for the entire character as well as the stop bit), only one character consisting of all zeros (with the FE status bit SR5 set) will be transferred to the holding register. The RxD input must return to a high condition before a search for the next start bit begins.

Pin 25 can be programmed to be a break detect output by appropriate setting of MR27-MR24. If so, a detected break will cause that pin to go high. When RxD returns to mark for one RxC time, pin 25 will go low. Refer to the break detection timing diagram.

Table 2. CPU-RELATED SIGNALS

PIN NAME	NO.	INPUT/ OUTPUT	FUNCTION
RESET	21	I	A high on this input performs a master reset on the 2661. This signal asynchronously terminates any device activity and clears the mode, command and status registers. The device assumes the idle state and remains there until initialized with the appropriate control words.
A0 – A1	12, 10	I	Address lines used to select internal EPCI registers.
$\overline{R}$/W	13	I	Read command when low, write command when high.
$\overline{CE}$	11	I	Chip enable command. When low, indicates that control and data lines to the EPCI are valid and that the operation specified by the $\overline{R}$/W, A_1 and A_0 inputs should be performed. When high, places the $D_0 - D_7$ lines in the three-state condition.
D7 – D0	27, 28 1, 2, 5 – 8	I/O	8-bit, three-state data bus used to transfer commands, data and status between EPCI and the CPU. D_0 is the least significant bit; D_7 the most significant bit.
$\overline{TxRDY}$	15	O	This output is the complement of status register bit SR0. When low, it indicates that the transmit data holding register (THR) is ready to accept a data character from the CPU. It goes high when the data character is loaded. This output is valid only when the transmitter is enabled. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{RxRDY}$	14	O	This output is the complement of status register bit SR1. When low, it indicates that the receive data holding register (RHR) has a character ready for input to the CPU. It goes high when the RHR is read by the CPU, and also when the receiver is disabled. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{TxEMT}$/ $\overline{DSCHG}$	18	O	This output is the complement of status register bit SR2. When low, it indicates that the transmitter has completed serialization of the last character loaded by the CPU, or that a change of state of the $\overline{DSR}$ or $\overline{DCD}$ inputs has occurred. This output goes high when the status register is read by the CPU, if the TxEMT condition does not exist. Otherwise, the THR must be loaded by the CPU for this line to go high. It is an open drain output which can be used as an interrupt to the CPU. See Status Register (SR2) for details.

When the EPCI is initialized into the synchronous mode, the receiver first enters the hunt mode on a 0 to 1 transition of RxEN(CR2). In this mode, as data are shifted into the receiver shift register a bit at a time, the contents of the register are compared to the contents of the SYN1 register. If the two are not equal, the next bit is shifted in and the comparison is repeated. When the two registers match, the hunt mode is terminated and character assembly mode begins. If single SYN operation is programmed, the SYN DETECT status bit is set. If double SYN operation is programmed, the first character assembled after SYN1 must be SYN2 in order for the SYN DETECT bit to be set. Otherwise, the EPCI returns to the hunt mode. (Note that the sequence SYN1 – SYN1 – SYN2 will not achieve syn-

chronization.) When synchronization has been achieved, the EPCI continues to assemble characters and transfer them to the holding register, setting the RxRDY status bit and asserting the $\overline{\text{RxRDY}}$ output each time a character is transferred. The PE and OE status bits are set as appropriate. Further receipt of the appropriate SYN sequence sets the SYN DETECT status bit. If the SYN stripping mode is commanded, SYN characters are not transferred to the holding register. Note that the SYN characters used to establish initial synchronization are not transferred to the holding register in any case.

External jam synchronization can be achieved via pin 9 by appropriate setting of MR27 – MR24. When pin 9 is an XSYNC input, the internal SYN1, SYN1 – SYN2, and DLE – SYN1 detection is disabled. Each positive going signal on XSYNC will cause the receiver to establish synchronization on the rising edge of the next RxC pulse. Character assembly will start with the RxD input at this edge. XSYNC may be lowered on the next rising edge of RxC. This external synchronization will cause the SYN DETECT status bit to be set until the status register is read. Refer to XSYNC timing diagram.

Transmitter

The EPCI is conditioned to transmit data when the $\overline{\text{CTS}}$ input is low and the TxEN command register bit is set. The SCN2661 indicates to the CPU that it can accept a character for transmission by setting the TxRDY status bit and asserting the $\overline{\text{TxRDY}}$ output. When the CPU writes a character into the transmit data holding register, these conditions are negated. Data are transferred from the holding register to the transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again. Thus, one full character time of buffering is provided.

In the asynchronous mode, the transmitter automatically sends a start bit followed by the programmed number of data bits, the least significant bit being sent first. It then appends an optional odd or even parity bit and the programmed number of stop bits. If, following transmission of the data bits, a new character is not available in the transmit holding register, the TxD output remains in the marking (high) condition and the $\overline{\text{TxEMT/DSCHG}}$ output and its corresponding status bit are asserted. Transmission resumes when the CPU loads a new character into the holding register. The transmitter can be forced to output a continuous low (BREAK) condition by setting the send break command bit (CR3) high.

In the synchronous mode, when the SCN2661 is initially conditioned to transmit, the TxD output remains high and the TxRDY condition is asserted until the first character to be transmitted (usually a SYN character) is loaded by the CPU. Subsequent to this, a continuous stream of characters is transmitted. No extra bits (other than parity, if commanded) are generated by the EPCI unless the CPU fails to send a new character to the EPCI by the time the transmitter has completed sending the previous character. Since synchronous communication does not allow gaps between characters, the EPCI asserts TxEMT and automatically "fills" the gap by transmitting SYN1s, SYN1 – SYN2 doublets, or DLE – SYN1 doublets, depending on the state of MR16 and MR17. Normal

Table 3. DEVICE RELATED SIGNALS

PIN NAME	NO.	INPUT/ OUTPUT	FUNCTION
BRCLK	20	I	Clock input to the internal baud rate generator (see table 1). Not required if external receiver and transmitter clocks are used.
$\overline{\text{RxC}}$/BKDET	25	I/O	Receiver clock. If external receiver clock is programmed, this input controls the rate at which the character is to be received. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. Data are sampled on the rising edge of the clock. If internal receiver clock is programmed, this pin can be a 1X/16X clock or a break detect output pin.
$\overline{\text{TxC}}$/XSYNC	9	I/O	Transmitter clock. If external transmitter clock is programmed, this input controls the rate at which the character is transmitted. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. The transmitted data changes on the falling edge of the clock. If internal transmitter clock is programmed, this pin can be a 1X/16X clock output or an external jam synchronization input.
RxD	3	I	Serial data input to the receiver. "Mark" is high, "space" is low.
TxD	19	O	Serial data output from the transmitter. "Mark" is high, "space" is low. Held in mark condition when the transmitter is disabled.
$\overline{\text{DSR}}$	22	I	General purpose input which can be used for data set ready or ring indicator condition. Its complement appears as status register bit SR7. Causes a low output on $\overline{\text{TxEMT/DSCHG}}$ when its state changes if CR2 or CR0 = 1.
$\overline{\text{DCD}}$	16	I	Data carrier detect input. Must be low in order for the receiver to operate. Its complement appears as status register bit SR6. Causes a low output on $\overline{\text{TxEMT/DSCHG}}$ when its state changes if CR2 or CR0 = 1. If $\overline{\text{DCD}}$ goes high while receiving, the RxC is internally inhibited.
$\overline{\text{CTS}}$	17	I	Clear to send input. Must be low in order for the transmitter to operate. If it goes high during transmission, the character in the transmit shift register will be transmitted before termination.
$\overline{\text{DTR}}$	24	O	General purpose output which is the complement of command register bit CR1. Normally used to indicate data terminal ready.
$\overline{\text{RTS}}$	23	O	General purpose output which is the complement of command register bit CR5. Normally used to indicate request to send. See Command Register (CR5) for details.

Table 4. SCN2661 REGISTER ADDRESSING

$\overline{CE}$	A_1	A_0	$\overline{R}/W$	FUNCTION
1	X	X	X	Three-state data bus
0	0	0	0	Read receive holding register
0	0	0	1	Write transmit holding register
0	0	1	0	Read status register
0	0	1	1	Write SYN1/SYN2/DLE registers
0	1	0	0	Read mode register ½
0	1	0	1	Write mode register ½
0	1	1	0	Read command register
0	1	1	1	Write command register

NOTE:
See AC characteristics section for timing requirements.

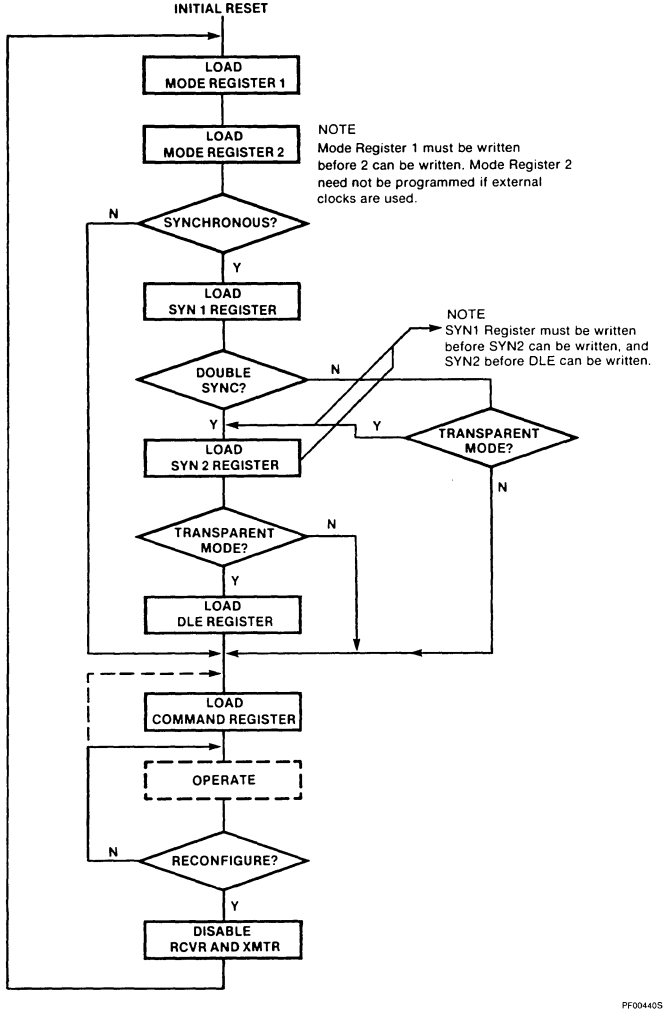


Figure 1. SCN2661 Initialization Flowchart

transmission of the message resumes when a new character is available in the transmit data holding register. If the send DLE bit in the command register is true, the DLE character is automatically transmitted prior to transmission of the message character in the THR.

2

EPCI PROGRAMMING

Prior to initiating data communications, the SCN2661 operational mode must be programmed by performing write operations to the mode and command registers. In addition, if synchronous operation is programmed, the appropriate SYN/DLE registers must be loaded. The EPCI can be reconfigured at any time during program execution. A flowchart of the initialization process appears in figure 1.

The internal registers of the EPCI are accessed by applying specific signals to the $\overline{CE}$, $\overline{R}/W$, A_1 and A_0 inputs. The conditions necessary to address each register are shown in table 4.

The SYN1, SYN2, and DLE registers are accessed by performing write operations with the conditions $A_1 = 0$, $A_0 = 1$, and $\overline{R}/W = 1$. The first operation loads the SYN1 register. The next loads the SYN2 register, and the third loads the DLE register. Reading or loading the mode registers is done in a similar manner. The first write (or read) operation addresses mode register 1, and a subsequent operation addresses mode register 2. If more than the required number of accesses are made, the internal sequencer recycles to point at the first register. The pointers are reset to SYN1 register and mode register 1 by a RESET input or by performing a read command register operation, but are unaffected by any other read or write operation.

The SCN2661 register formats are summarized in tables 5, 6, 7 and 8. Mode registers 1 and 2 define the general operational characteristics of the EPCI, while the command register controls the operation within this basic framework. The EPCI indicates its status in the status register. These registers are cleared when a RESET input is applied.

Mode Register 1 (MR1)

Table 5 illustrates mode register 1. Bits MR11 and MR10 select the communication format and baud rate multiplier. 00 specifies synchronous mode and 1X multiplier. 1X, 16X, and 64X multipliers are programmable for asynchronous format. However, the multiplier in asynchronous format applies only if the external clock input option is selected by MR24 or MR25.

MR13 and MR12 select a character length of 5, 6, 7 or 8 bits. The character length does not include the parity bit, if programmed, and

Table 5. MODE REGISTER 1 (MR1)

MR17	MR16	MR15	MR14	MR13	MR12	MR11	MR10
Sync/Async		Parity Type	Parity Control	Character Length		Mode and Baud Rate Factor	
Async: Stop bit length 00 = Invalid 01 = 1 stop bit 10 = 1½ stop bits 11 = 2 stop bits		0 = Odd 1 = Even	0 = Disabled 1 = Enabled	00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits		00 = Synchronous 1X rate 01 = Asynchronous 1X rate 10 = Asynchronous 16X rate 11 = Asynchronous 64X rate	
Sync: Number of SYN char 0 = Double SYN 1 = Single SYN	**Sync: Transparency control** 0 = Normal 1 = Transparent						

NOTE:
Baud rate factor in asynchronous applies only if external clock is selected. Factor is 16X if internal clock is selected. Mode must be selected (MR11, MR10) in any case.

Table 6. MODE REGISTER 2 (MR2)

MR27 – MR24											MR23 – MR20
	TxC	RxC	Pin 9	Pin 25		TxC	RxC	Pin 9	Pin 25	Mode	Baud Rate Selection
0000	E	E	TxC	RxC	1000	E	E	XSYNC[1]	RxC/TxC	sync	See baud rates in table 1
0001	E	I	TxC	1X	1001	E	I	TxC	BKDET	async	
0010	I	E	1X	RxC	1010	I	E	XSYNC[1]	RxC	sync	
0011	I	I	1X	1X	1011	I	I	1X	BKDET	async	
0100	E	E	TxC	RxC	1100	E	E	XSYNC[1]	RxC/TxC	sync	
0101	E	I	TxC	16X	1101	E	I	TxC	BKDET	async	
0110	I	E	16X	RxC	1110	I	E	XSYNC[1]	RxC	sync	
0111	I	I	16X	16X	1111	I	I	16X	BKDET	async	

NOTES:
1. When pin 9 is programmed as XSYNC input, SYN1, SYN1-SYN2, and DLE-SYN1 detection is disabled.
E = External clock
I = Internal clock (BRG)
1X and 16X are clock outputs.

does not include the start and stop bits in asynchronous mode.

MR14 controls parity generation. If enabled, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR15 selects odd or even parity when parity is enabled by MR14.

In asynchronous mode, MR17 and MR16 select character framing of 1, 1.5, or 2 stop bits. (If 1X baud rate is programmed, 1.5 stop bits defaults to 1 stop bits on transmit.) In synchronous mode, MR17 controls the number of SYN characters used to establish synchronization and for character fill when the transmitter is idle. SYN1 alone is used if MR17 = 1, and SYN1 – SYN2 is used when MR17 = 0. If the transparent mode is specified by MR16, DLE – SYN1 is used for character fill and SYN detect, but the normal synchronization sequence is used to establish character sync. When transmitting, a DLE character in the transmit holding register will cause a second DLE character to be transmitted. This DLE stuffing eliminates the software DLE compare and stuff on each transparent mode data character. If the send DLE command (CR3) is active when a DLE is loaded into THR, only one additional DLE will be transmitted. Also, DLE stripping and DLE detect (with MR14 = 0) are enabled.

The bits in the mode register affecting character assembly and disassembly (MR12 – MR16) can be changed dynamically (during active receive/transmit operation). The character mode register affects both the transmitter and receiver; therefore in synchronous mode, changes should be made only in half-duplex mode (RxEN = 1 or TxEN = 1, but not both simultaneously = 1). In asynchronous mode, character changes should be made when RxEN and TxEN = 0 or when TxEN = 1 and the transmitter is marking in half-duplex mode (RxEN = 0).

To effect assembly/disassembly of the next received/transmitted character, MR12 – 15 must be changed within n bit times of the active going state of $\overline{\text{RxRDY}}/\overline{\text{TxRDY}}$. Transparent and non-transparent mode changes (MR16) must occur within n – 1 bit times of the character to be affected when the receiver or transmitter is active. (n = smaller of the new and old character lengths.)

Table 7. COMMAND REGISTER (CR)

CR7 CR6	CR5	CR4	CR3	CR2	CR1	CR0
Operating Mode	**Request To Send**	**Reset Error**	**Sync/ Async**	**Receive Control (RxEN)**	**Data Terminal Ready**	**Transmit Control (TxEN)**
00 = Normal operation 01 = Async: Automatic echo mode Sync: SYN and/or DLE stripping mode 10 = Local loopback 11 = Remote loopback	0 = Force $\overline{RTS}$ output high one clock time after TxSR serialization 1 = Force $\overline{RTS}$ output low	0 = Normal 1 = Reset error flags in status register (FE, OE, PE/DLE detect.)	**Async: Force break** 0 = Normal 1 = Force break **Sync Send DLE** 0 = Normal 1 = SendDLE	0 = Disable 1 = Enable	0 = Force $\overline{DTR}$ output high 1 = Force $\overline{DTR}$ output low	0 = Disable 1 = Enable

Table 8. STATUS REGISTER (SR)

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
Data Set Ready	**Data Carrier Detect**	**FE/SYN Detect**	**Overrun**	**PE/DLE Detect**	**TxEMT DSCHG**	**RxRDY**	**TxRDY**
0 = $\overline{DSR}$ input is high 1 = $\overline{DSR}$ input is low	0 = $\overline{DCD}$ input is high 1 = $\overline{DCD}$ input is low	**Async:** 0 = Normal 1 = Framing error **Sync:** 0 = Normal 1 = SYN detected	0 = Normal 1 = Overrun error **Sync:**	**Async:** 0 = Normal 1 = Parity error 0 = Normal 1 = Parity error or DLE received	0 = Normal 1 = Change in $\overline{DSR}$ or $\overline{DCD}$, or transmit shift register is empty	0 = Receive holding register empty 1 = Receive holding register has data	0 = Transmit holding register busy 1 = Transmit holding register empty

Mode Register 2 (MR2)

Table 6 illustrates mode register 2. MR23, MR22, MR21 and MR20 control the frequency of the internal baud rate generator (BRG). Sixteen rates are selectable for each EPCI version (–1, –2, –3). Versions 1 and 2 specify a 4.9152MHz TTL input at BRCLK (pin 20); version 3 specifies a 5.0688MHz input which is identical to the Signetics 2651. MR23 – 20 are don't cares if external clocks are selected (MR25 – MR24 = 0). The individual rates are given in table 1.

MR24 – MR27 select the receive and transmit clock source (either the BRG or an external input) and the function at pins 9 and 25. Refer to table 6.

Command Register (CR)

Table 7 illustrates the command register. Bits CR0 (TxEN) and CR2 (RxEN) enable or disable the transmitter and receiver respectively. A 0 to 1 transition of CR2 forces start bit search (async mode) or hunt mode (sync mode) on the second $\overline{RxC}$ rising edge. Disabling the receiver causes $\overline{RxRDY}$ to go high (inactive). If the transmitter is disabled, it will complete the transmission of the character in the transmit shift register (if any) prior to terminating operation. The TxD output will then remain in the marking state (high) while $\overline{TxRDY}$ and $\overline{TxEMT}$ will go high (inactive). If the receiver is disabled, it will terminate operation immediately. Any character being assembled will be neglected. A 0 to 1 transition of CR2 will initiate start bit search (async) or hunt mode (sync).

Bits CR1 (DTR) and CR5 (RTS) control the $\overline{DTR}$ and $\overline{RTS}$ outputs. Data at the outputs are the logical complement of the register data.

In asynchronous mode, setting CR3 will force and hold the TxD output low (spacing condition) at the end of the current transmitted character. Normal operation resumes when CR3 is cleared. The TxD line will go high for at least one bit time before beginning transmission of the next character in the transmit data holding register. In synchronous mode, setting CR3 causes the transmission of the DLE register contents prior to sending the character in the transmit data holding register. Since this is a one time command, CR3 does not have to be reset by software. CR3 should be set when entering and exiting transparent mode and for all DLE – non-DLE character sequences.

Setting CR4 causes the error flags in the status register (SR3, SR4, and SR5) to be cleared. This is a one time command. There is no internal latch for this bit.

When CR5 (RTS) is set, the $\overline{RTS}$ pin is forced low. A 1 to 0 transition of CR5 will cause $\overline{RTS}$ to go high (inactive) one TxC time after the last serial bit has been transmitted. If a 1-to-0 transition of CR5 occurs while data is being transmitted, $\overline{RTS}$ will remain low (active) until both the THR and the transmit shift register are empty and then go high (inactive) one TxC time later.

The EPCI can operate in one of four submodes within each major mode (synchronous or asynchronous). The operational sub-mode is determined by CR7 and CR6. CR7 – CR6 = 00 is the normal mode, with the transmitter and receiver operating independently in accordance with the mode and status register instructions.

In asynchronous mode, CR7 – CR6 = 01 places the EPCI in the automatic echo mode. Clocked, regenerated received data are automatically directed to the TxD line while normal receiver operation continues. The receiver must be enabled (CR2 = 1), but the transmitter need not be enabled. CPU to receiver communications continues normally, but the CPU to transmitter link is disabled. Only the first character of a break condition is echoed.

Table 9. SCN2661 EPCI vs SCN2651 PCI

FEATURE	EPCI	PCI
1. MR2 Bit 6, 7	Control pin 9, 25	Not used
2. DLE detect – SR3	SR3 = 0 for DLE – DLE, DLE – SYN1	SR3 = 1 for DLE – DLE, DLE – SYN1
3. Reset of SR3, DLE detect	Second character after DLE, or receiver disable, or CR4 = 1	Receiver disable, or CR4 = 1
4. Send DLE – CR3	One time command	Reset via CR3 on next $\overline{\text{TxRDY}}$
5. DLE stuffing in transparent mode	Automatic DLE stuffing when DLE is loaded except if CR3 = 1	None
6. SYN1 stripping in double sync non-transparent mode	All SYN1	First SYN1 of pair
7. Baud rate versions	Three	One
8. Terminate ASYNC transmission (drop RTS)	Reset CR5 in response to $\overline{\text{TxEMT}}$ changing from 1 to 0	Reset CR0 when $\overline{\text{TxEMT}}$ goes from 1 to 0. Then reset CR5 when $\overline{\text{TxEMT}}$ goes from 1 to 0
9. Break detect	Pin 25[1]	FE and null character
10. Stop bit searched	One	Two
11. External jam sync	Pin 9[2]	No
12. Data bus timing	Improved over 2651	—
13. Data bus drivers	Sink 2.2mA Source 400μA	Sink 1.6mA Source 100μA

NOTES:
1. Internal BRG used for RxC.
2. Internal BRG used for TxC.

The TxD output will go high until the next valid start is detected. The following conditions are true while in automatic echo mode:

1. Data assembled by the receiver are automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. $\overline{\text{TxRDY}}$ output = 1.
4. The $\overline{\text{TxEMT/DSCHG}}$ pin will reflect only the data set change condition.
5. The TxEN command (CR0) is ignored.

In synchronous mode, CR7 – CR6 = 01 places the EPCI in the automatic SYN/DLE stripping mode. The exact action taken depends on the setting of bits MR17 and MR16:

1. In the non-transparent, single SYN mode (MR17 – MR16 = 10), characters in the data stream matching SYN1 are not transferred to the receive data holding register (RHR).
2. In the non-transparent, double SYN mode (MR17 – MR16 = 00), characters in the data stream matching SYN1, or SYN2 if immediately preceded by SYN1, are not transferred to the RHR.
3. In transparent mode (MR16 = 1), characters in the data stream matching DLE, or SYN1 if immediately preceded by DLE, are not transferred to the RHR. However, only the first DLE of a DLE-DLE pair is stripped.

Note that automatic stripping mode does not affect the setting of the DLE detect and SYN detect status bits (SR3 and SR5).

Two diagnostic sub-modes can also be configured. In local loopback mode (CR7 – CR6 = 10), the following loops are connected internally:

1. The transmitter output is connected to the receiver input.
2. $\overline{\text{DTR}}$ is connected to $\overline{\text{DCD}}$ and $\overline{\text{RTS}}$ is connected to $\overline{\text{CTS}}$.
3. The receiver is clocked by the transmit clock.
4. The $\overline{\text{DTR}}$, $\overline{\text{RTS}}$ and $\overline{\text{TxD}}$ outputs are held high.
5. The $\overline{\text{CTS}}$, $\overline{\text{DCD}}$, $\overline{\text{DSR}}$ and RxD inputs are ignored.

Additional requirements to operate in the local loopback mode are that CR0 (TxEN), CR1 (DTR), and CR5 (RTS) must be set to 1. CR2 (RxEN) is ignored by the EPCI.

The second diagnostic mode is the remote loopback mode (CR7 – CR6 = 11). In this mode:

1. Data assembled by the receiver are automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. No data are sent to the local CPU, but the error status conditions (PE, FE) are set.
4. The $\overline{\text{RxRDY}}$, $\overline{\text{TxRDY}}$, and $\overline{\text{TxEMT/DSCHG}}$ outputs are held high.
5. CR0 (TxEN) is ignored.
6. All other signals operate normally.

Status Register

The data contained in the status register (as shown in table 8) indicate receiver and transmitter conditions and modem/data set status.

SR0 is the transmitter ready (TxRDY) status bit. It, and its corresponding output, are valid only when the transmitter is enabled. If equal to 0, it indicates that the transmit data holding register has been loaded by the CPU and the data has not been transferred to the transmit shift register. If set equal to 1, it indicates that the holding register is ready to accept data from the CPU. This bit is initially set when the transmitter is enabled by CR0, unless a character has previously been loaded into the holding register. It is not set when the automatic echo or remote loopback modes are programmed. When this bit is set, the $\overline{\text{TxRDY}}$ output pin is low. In the automatic echo and remote loopback modes, the output is held high.

SR1, the receiver ready (RxRDY) status bit, indicates the condition of the receive data holding register. If set, it indicates that a character has been loaded into the holding register from the receive shift register and is ready to be read by the CPU. If equal to zero, there is no new character in the holding register. This bit is cleared when the CPU reads the receive data holding register or when the receiver is disabled by CR2. When set, the $\overline{\text{RxRDY}}$ output is low.

The TxEMT/DSCHG bit, SR2, when set, indicates either a change of state of the $\overline{\text{DSR}}$ or $\overline{\text{DCD}}$ inputs (when CR2 or CR0 = 1) or that the transmit shift register has completed transmission of a character and no new character has been loaded into the transmit data holding register. Note that in synchronous mode this bit will be set even though the appropriate "fill" character is transmitted. TxEMT will not go active until at least one character has been transmitted. It is cleared by loading the transmit data holding register. The DSCHG condition is enabled when TxEN = 1 or RxEN = 1. It is cleared when the status register is read by the CPU. If the status register is read twice and SR2 = 1 while SR6 and SR7 remain unchanged, then a TxEMT condition exists. When SR2 is set, the $\overline{\text{TxEMT/DSCHG}}$ output is low.

SR3, when set, indicates a received parity error when parity is enabled by MR14. In

synchronous transparent mode (MR16 = 1), with parity disabled, it indicates that a character matching DLE register was received and the present character is neither SYN1 nor DLE. This bit is cleared when the next character following the above sequence is loaded into RHR, when the receiver is disabled, or by a reset error command, CR4.

The overrun error status bit, SR4, indicates that the previous character loaded into the receive holding register was not read by the CPU at the time a new received character was transferred into it. This bit is cleared when the receiver is disabled or by the reset error command, CR4.

In asynchronous mode, bit SR5 signifies that the received character was not framed by a stop bit; i.e., only the first stop bit is checked. If RHR = 0 when SR5 = 1, a break condition is present. In synchronous nontransparent mode (MR16 = 0), it indicates receipt of the SYN1 character in single SYN mode or the SYN1 – SYN2 pair in double SYN mode. In synchronous transparent mode (MR16 = 1), this bit is set upon detection of the initial synchronizing characters (SYN1 or SYN1 – SYN2) and, after synchronization has been achieved, when a DLE – SYN1 pair is received. The bit is reset when the receiver is disabled, when the reset error command is given in asynchronous mode, or when the status register is read by the CPU in the synchronous mode.

SR6 and SR7 reflect the conditions of the $\overline{DCD}$ and $\overline{DSR}$ inputs respectively. A low input sets its corresponding status bit, and a high input clears it.

2

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	Note 4	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Input voltage						V
V_{IL}	Low				0.8	
V_{IH}	High		2.0			
Output voltage						V
V_{OL}	Low	I_{OL} = 2.2mA			0.4	
V_{OH}[7]	High	I_{OH} = −400μA	2.4			
I_{IL}	Input leakage current	V_{IN} = 0 to 5.5V			10	μA
3-state output leakage current						μA
I_{LH}	Data bus high	V_O = 4.0V			10	
I_{LL}	Data bus low	V_O = 0.45V			10	
I_{CC}	Power supply current				150	mA

CAPACITANCE T_A = 25°C, V_{CC} = 0V

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Capacitance						pF
C_{IN}	Input				20	
C_{OUT}	Output	fc = 1MHz			20	
$C_{I/O}$	Input/Output	Unmeasured pins tied to ground			20	

AC ELECTRICAL CHARACTERISTICS[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Pulse width						ns
t_{RES}	Reset		1000			
t_{CE}	Chip enable		250			
Set-up and hold time						ns
t_{AS}	Address set-up		10			
t_{AH}	Address hold		10			
t_{CS}	$\overline{R}$/W control set-up		10			
t_{CH}	$\overline{R}$/W control hold		10			
t_{DS}	Data set-up for write		150			
t_{DH}	Data hold for write		10			
t_{RXS}	RX data set-up		300			
t_{RXH}	RX data hold		350			
t_{DD}	Data delay time for read	C_L = 150pF			200	ns
t_{DF}	Data bus floating time for read	C_L = 150pF			100	
t_{CED}	CE to CE delay		600			
Input clock frequency						MHz
f_{BRG}	Baud rate generator (2661A,B)		1.0	4.9152	4.9202	
f_{BRG}	Baud rate generator (2661C)		1.0	5.0688	5.0738	
$f_{R/T}$[10]	$\overline{TxC}$ or $\overline{RxC}$		dc		1.0	
Clock width						ns
t_{BRH}[9]	Baud rate high (2661A,B)		75			
t_{BRH}[9]	Baud rate high (2661C)		70			
t_{BRL}[9]	Baud rate low (2661A,B)		75			
t_{BRL}[9]	Baud rate low (2661C)		70			
$t_{R/TH}$	$\overline{TxC}$ or $\overline{RxC}$ high		480			
$t_{R/TL}$[10]	$\overline{TxC}$ or $\overline{RxC}$ low		480			
t_{TXD}	TxD delay from falling edge of $\overline{TxC}$	C_L = 150pF			650	ns
t_{TCS}	Skew between TxD changing and falling edge of $\overline{TxC}$ output[8]	C_L = 150pF		0		

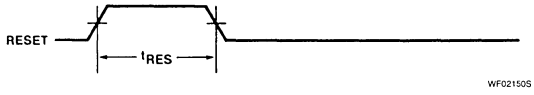
NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified. See ordering code table for applicable temperature range and operating supply range.
5. All voltage measurements are referenced to ground. All time measurements are at the 50% level for inputs (except t_{BRH} and t_{BRL}) and at 0.8V and 2.0V for outputs. Input levels swing between 0.4V and 2.4V, with a transition time of 20ns maximum.
6. Typical values are at +20°C, typical supply voltages and typical processing parameters.
7. $\overline{INTR}$, $\overline{TxRDY}$, $\overline{RxRDY}$ and $\overline{TxEMT/DSCHG}$ outputs are open drain.
8. Parameter applies when internal transmitter clock is used.
9. Under test conditions of 5.0688MHz f_{BRG} (2661C) and 4.9152MHz f_{BRG} (2661A,B), t_{BRH} and t_{BRL} measured at V_{IH} and V_{IL} respectively.
10. In asynchronous local loopback mode, using 1X clock, the following parameters apply: $f_{R/T}$ = 0.83MHz max and $t_{R/TL}$ = 700ns min.

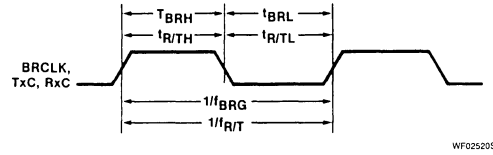
2

TIMING DIAGRAMS

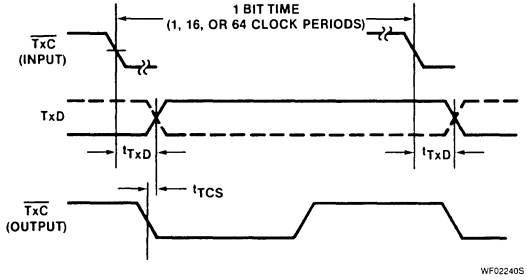
RESET



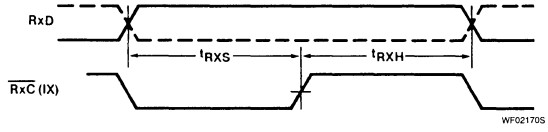
CLOCK



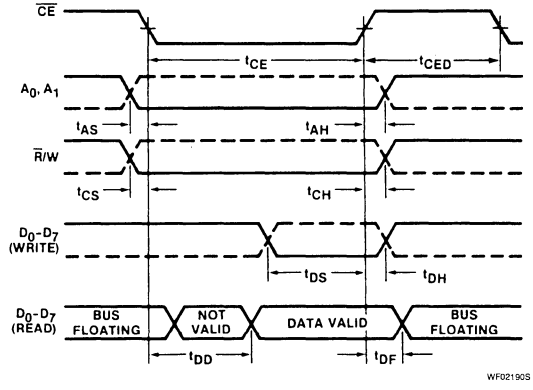
TRANSMIT



RECEIVE

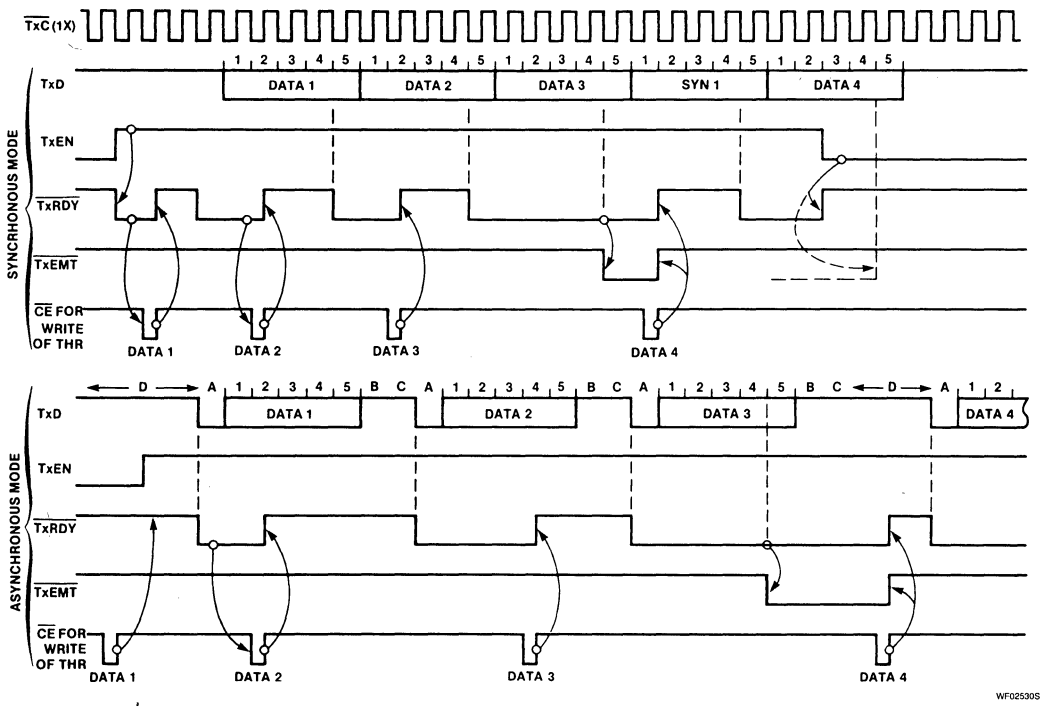


READ AND WRITE



TIMING DIAGRAMS (Continued)

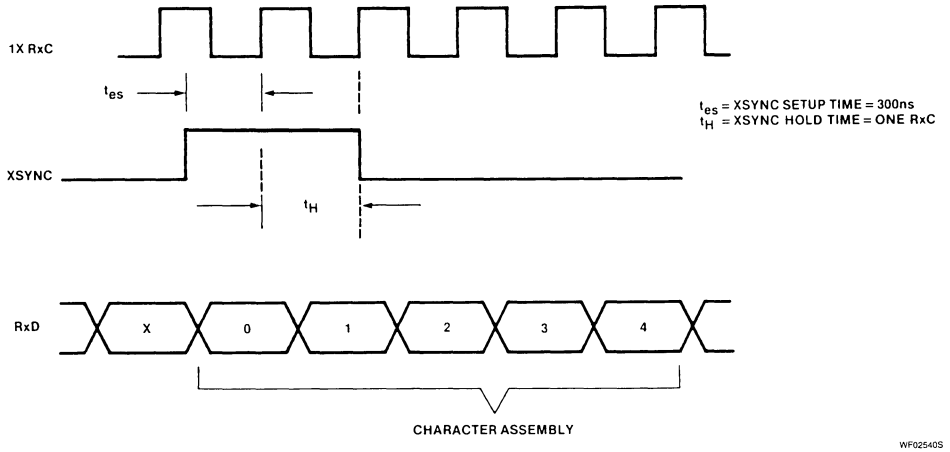
TxRDY, TxEMT (Shown for 5-bit characters, no parity, 2 stop bits [in asynchronous mode])



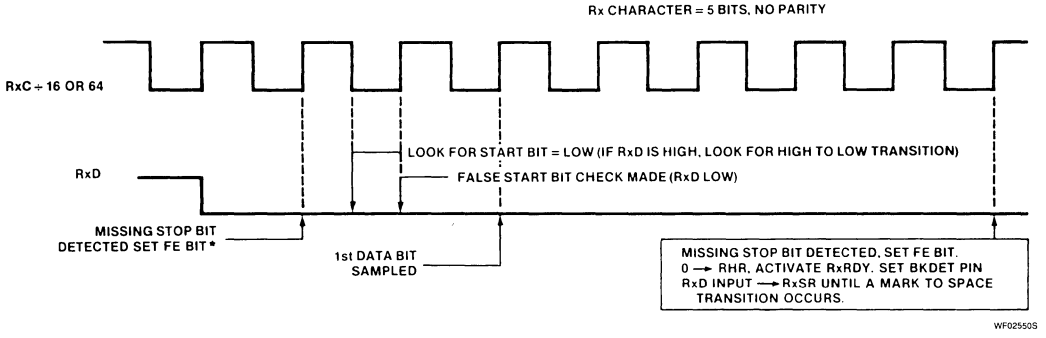
NOTES:
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
TxEMT goes low at the beginning of the last data bit, or, if parity is enabled, at the beginning of the parity bit.

TIMING DIAGRAMS (Continued)

EXTERNAL SYNCHRONIZATION WITH XSYNC



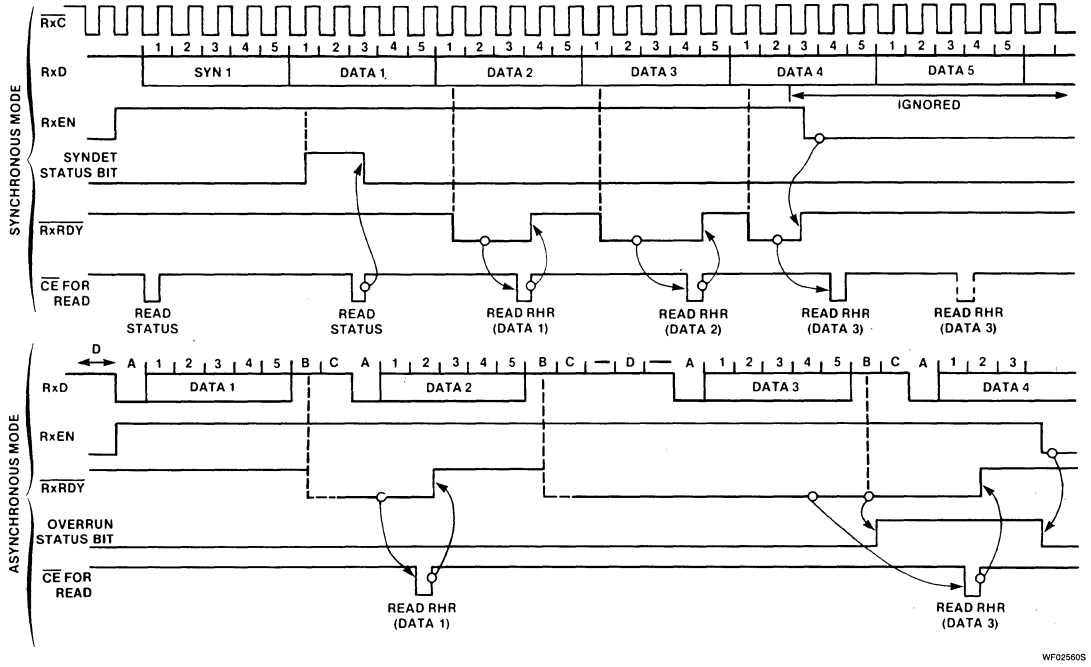
BREAK DETECTION TIMING



NOTE:
* If the stop bit is present, the start bit search will commence immediately.

TIMING DIAGRAMS (Continued)

RxRDY (Shown for 5-bit characters, no parity, 2 stop bits [in asynchronous mode])

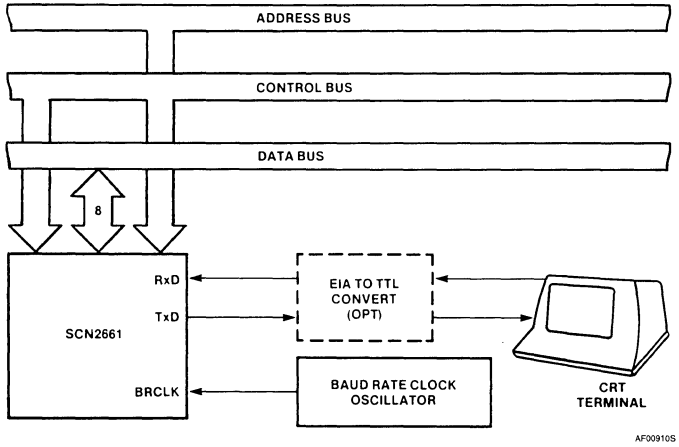


NOTES:
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
Only one stop bit is detected.

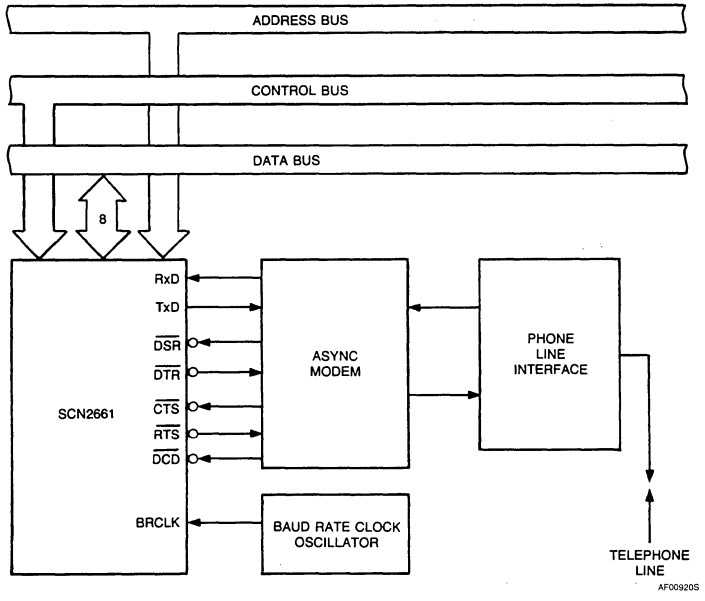
2

TYPICAL APPLICATIONS

ASYNCHRONOUS INTERFACE TO CRT TERMINAL

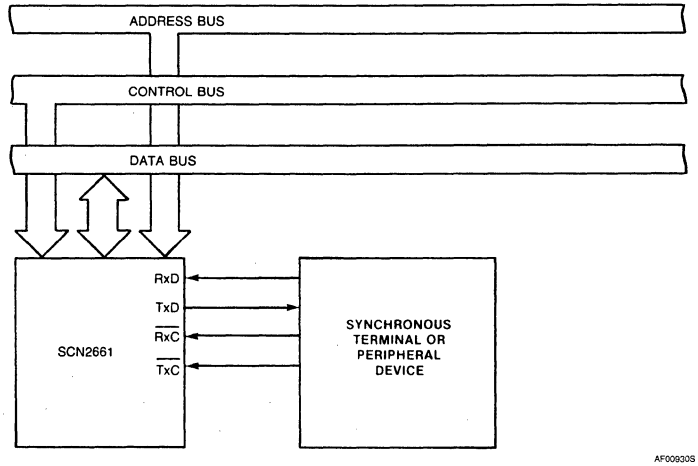


ASYNCHRONOUS INTERFACE TO TELEPHONE LINES

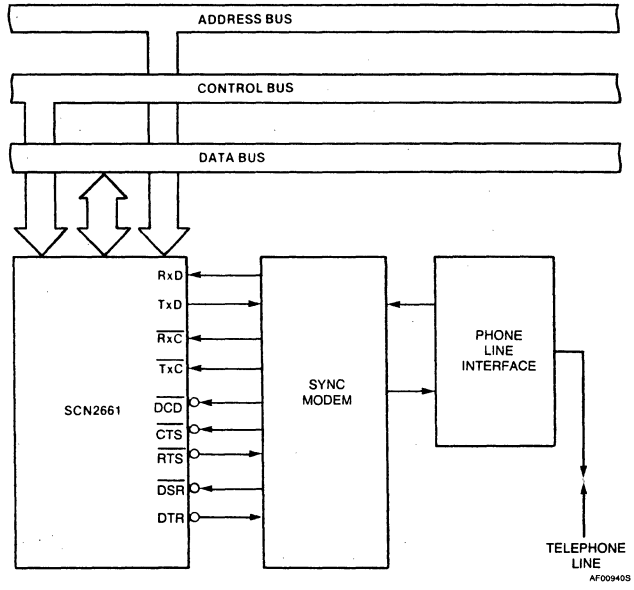


TYPICAL APPLICATIONS (Continued)

SYNCHRONOUS INTERFACE TO TERMINAL OR PERIPHERAL DEVICE



SYNCHRONOUS INTERFACE TO TELEPHONE LINES



SCN2672
Programmable Video Timing Controller (PVTC)

Product Specification

Microprocessor Products

DESCRIPTION

The Signetics SCN2672 Programmable Video Timing Controller (PVTC) is a programmable device designed for use in CRT terminals and display systems that employ raster scan techniques. The PVTC generates the vertical and horizontal timing signals necessary for the display of interlaced or non-interlaced data on a CRT monitor. It provides consecutive addressing to a user specified display buffer memory domain and controls the CPU-display buffer interface for various buffer configuration modes. A variety of operating modes, display formats, and timing profiles can be implemented by programming the control registers in the PVTC.

A minimum CRT terminal system configuration consists of a PVTC, an SCN2671 Keyboard and Communication Controller (PKCC), an SCN2670 Display Character and Graphics Generator (DCGG), an SCN2673/2677 Video and Attributes Controller (VAC), a single chip microcomputer such as the 8048, a display buffer RAM, and a small amount of TTL for miscellaneous address decoding, interface, and control. Typically, the package count for a minimum system is between 15 and 20 devices; system complexity can be enhanced by upgrading the microprocessor and expanding via the system address and data busses.

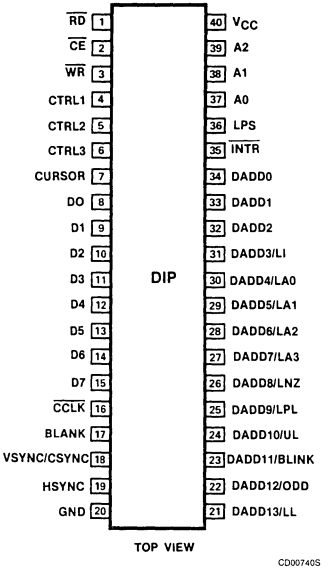
FEATURES

- **4MHz and 2.7MHz character rate versions**
- **Up to 256 characters per row**
- **1 to 16 raster lines per character row**
- **Up to 128 character rows per frame**
- **Programmable horizontal and vertical sync generators**
- **Interlaced or non-interlaced operation**
- **Up to 16K RAM addressing for multiple page operation**
- **Automatic wraparound of RAM**
- **Addressable incrementable and readable cursor**
- **Programmable cursor size, position, and blink**
- **Split screen and horizontal scroll capability**
- **Light pen register**
- **Selectable buffer interface modes**
- **Dynamic RAM refresh**
- **Completely TTL compatible**
- **Single +5 volt power supply**
- **Power on reset circuit**

APPLICATIONS

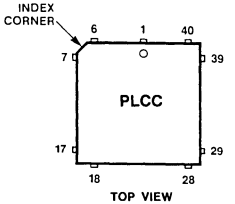
- **CRT terminals**
- **Word processing systems**
- **Small business computers**
- **Home Computers**

PIN CONFIGURATION



TOP VIEW

CD00740S



TOP VIEW

CD0044PS

Pin	Function	Pin	Function
1	NC	23	NC
2	$\overline{RD}$	24	DADD13/LL
3	$\overline{CE}$	25	DADD12/ODD
4	$\overline{WR}$	26	DADD11/BLINK
5	CTRL1	27	DADD10/UL
6	CTRL2	28	DADD9/LPL
7	CTRL3	29	DADD8/LNZ
8	CURSOR	30	DADD7/LA3
9	D0	31	DADD6/LA2
10	D1	32	DADD5/LA1
11	D2	33	DADD4/LA0
12	NC	34	NC
13	D3	35	DADD3/LI
14	D4	36	DADD2
15	D5	37	DADD1
16	D6	38	DADD0
17	D7	39	$\overline{INTR}$
18	$\overline{CCLK}$	40	LPS
19	BLANK	41	A0
20	VSYNC/CSYNC	42	A1
21	HSYNC	43	A2
22	GND	44	V_{CC}

Programmable Video Timing Controller (PVTC) SCN2672

ORDERING CODE

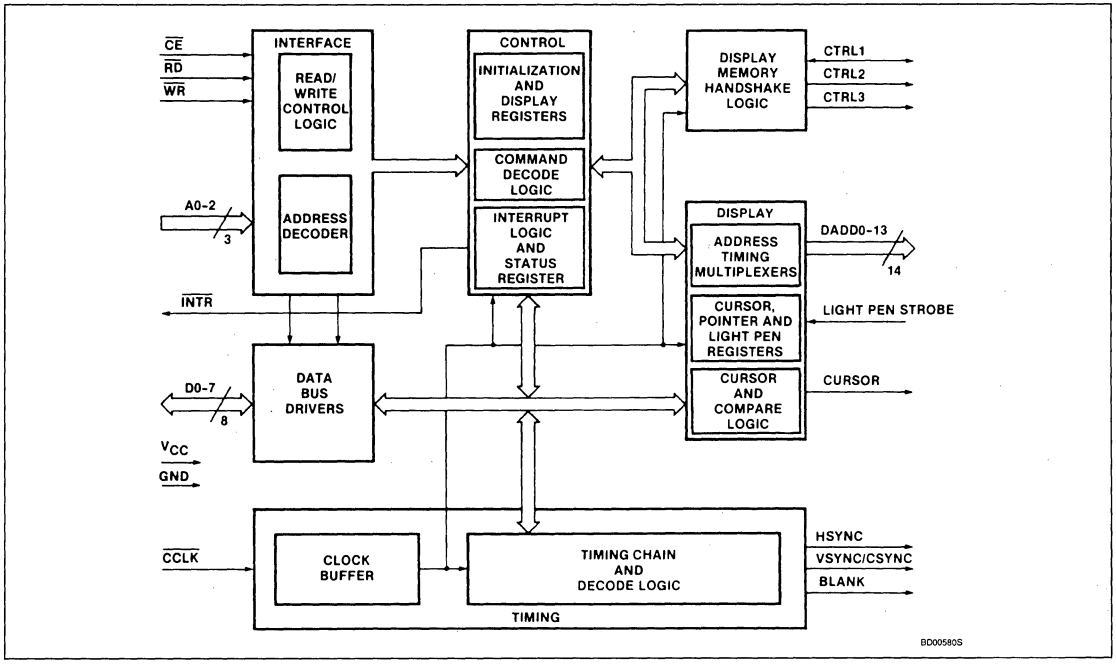
PACKAGES	V_{CC} = 5V ±5%, T_A = 0°C to 70°C	
	4MHz	2.7MHz
Ceramic DIP	SCN2672BC4I40	SCN2672BC3I40
Plastic DIP	SCN2672BC4N40	SCN2672BC3N40
Plastic LCC	SCN2672BC4A44	SCN2672BC3A44

FUNCTIONAL DESCRIPTION

As shown on the block diagram, the PVTC contains the following major blocks:

- Data bus buffer
- Interface Logic
- Operation Control
- Timing
- Display Control
- Buffer Control

BLOCK DIAGRAM



Data Bus Driver

The data bus buffer provides the interface between the external and internal data busses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the PVTC.

Interface Logic

The interface logic contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer. The functions performed by the CPU read and write operations are as shown in table 1.

Table 1. PVTC ADDRESSING

A2	A1	A0	READ ($\overline{RD}$ = 0)	WRITE ($\overline{WR}$ = 0)
0	0	0	Interrupt register	Initialization registers[1]
0	0	1	Status register	Command register
0	1	0	Screen start address lower register	Screen start address lower reg.
0	1	1	Screen start address upper register	Screen start address upper reg.
1	0	0	Cursor address lower register	Cursor address lower register
1	0	1	Cursor address upper register	Cursor address upper register
1	1	0	Light pen address lower register	Display pointer address lower reg.
1	1	1	Light pen address upper register	Display pointer address upper reg.

NOTE:

1. There are 11 initialization registers which are accessed sequentially via a single address. The PVTC maintains an internal pointer to these registers which is incremented after each write at this address until the last register (IR10, the split screen register) is accessed. The pointer then continues to point to the split screen register. Upon power-up or a master reset command, the internal pointer is reset to point to the first register (IR0) of the initialization register group. The internal pointer can also be preset to any register of the group via the 'load IR address pointer' command.

PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
A0 – A2	37 – 39	41 – 43	I	**Address Lines:** Used to select PVTC internal registers for read/write operations and for commands.
D0 – D7	8 – 15	9 – 11, 13 – 17	I/O	**8-Bit Bidirectional Three-State Data Bus.** Bit 0 is the LSB and bit 7 is the MSB. All data, command, and status transfers between the CPU and the PVTC take place over this bus. The direction of the transfer is controlled by the $\overline{RD}$ and $\overline{WR}$ inputs when the $\overline{CE}$ input is low. When the $\overline{CE}$ input is high, the data bus is in the three-state condition.
$\overline{RD}$	1	2	I	**Read Strobe:** Active low input. A low on this pin while $\overline{CE}$ is low causes the contents of the register selected by A0 – A2 to be placed on the data bus. The read cycle begins on the leading (falling) edge of $\overline{RD}$.
$\overline{WR}$	3	4	I	**Write Strobe:** Active low input. A low on this pin while $\overline{CE}$ is also low causes the contents of the data bus to be transferred to the register selected by A0 – A2. The transfer occurs on the trailing (rising) edge of $\overline{WR}$.
$\overline{CE}$	2	3	I	**Chip Enable:** Active low input. When low, data transfers between the CPU and the PVTC are enabled on D0 – D7 as controlled by the $\overline{WR}$, $\overline{RD}$, and A0 – A2 inputs. When $\overline{CE}$ is high, the PVTC is effectively isolated from the data bus and D0 – D7 are placed in the three-state condition.
$\overline{CCLK}$	16	18	I	**Character Clock:** Timing signal derived from the video dot clock which is used to synchronize the PVTC's timing functions.
HSYNC	19	21	O	**Horizontal Sync:** Active high output which provides video horizontal sync pulses. The timing parameters are programmable.
VSYNC/CSYNC	18	20	O	**Vertical Sync/Composite Sync:** A control bit selects either vertical or composite sync pulses on this active high output. When CSYNC is selected, equalization pulses are included. The timing parameters are programmable.
BLANK	17	19	O	**Blank:** This active high output defines the horizontal and vertical borders of the display. Display control signals which are output on DADD3 through DADD13 are valid on the trailing edge of BLANK.
CURSOR	7	8	O	**Cursor Gate:** This active high output becomes active for a specified number of scan lines when the address contained in the cursor registers match the address output on DADD0 through DADD13. The first and last lines of the cursor and a blink option are programmable.
$\overline{INTR}$	35	39	O	**Interrupt Request:** Open drain output which supplies an active low interrupt request from any of five maskable sources. This pin is inactive after power on reset or a master reset command.
LPS	36	40	I	**Light Pen Strobe:** Positive edge triggered input indicating a light pen hit. Causes the current value of the display address to be strobed into the light pen register.
CTRL1	4	5	I/O	**Handshake Control 1:** In independent mode, provides an active low write data buffer ($\overline{WDB}$) output which strobes data from the interface latch into the display memory. In transparent and shared modes, this is an active low processor bus request ($\overline{PBREQ}$) input which indicates that the CPU desires to access the display memory. This pin must be tied high when operating in row buffer mode.
CTRL2	5	6	O	**Handshake Control 2:** In independent mode, provides an active low read data buffer ($\overline{RDB}$) output which strobes data from the display memory into the interface latch. In transparent and shared modes, this is an active low bus external enable ($\overline{BEXT}$) output which indicates that the PVTC has relinquished control of the display memory (DADD0 – DADD13 are in the three-state condition) in response to a CPU bus request. $\overline{BEXT}$ also goes low in response to a 'display off and float DADD' command. In row buffer mode, it is an active low bus request ($\overline{BREQ}$) output which halts the CPU during a line DMA.
CTRL3	6	7	O	**Handshake Control 3:** In independent mode, provides the active low buffer chip enable ($\overline{BCE}$) signal to the display memory. In transparent and shared modes provides an active low bus acknowledge ($\overline{BACK}$) output which serves as a ready signal to the CPU in response to a processor bus request. In row buffer mode, this is an active high memory bus control

PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
				(MBC) output which configures the system for the DMA transfer of one row of character codes from system memory to the row display buffer.
DADD0 – DADD13	34 – 21	38 – 35 33 – 24	O	**Display Address:** Used by the PVTC to address up to 16K of display memory. These outputs are floated at various times depending on the buffer mode. Various control signals are multiplexed on DADD3 thru DADD13 and are valid at the trailing edge of BLANK. These control signals are: **DADD3 /LI** **Line Interlace:** Replaces DADD4/LA0 as the least significant line address for the interlaced sync and video applications. A low indicates an even row of an even field or an odd row of an odd field. **DADD4 – DADD7/LA0 – LA3** **Line Address:** Provides the number of the current scan line within each character row. **DADD8/LNZ** **Line Zero:** Asserted before the first scan line in each character row. **DADD9/LPL** **Light Pen Line:** Asserted before the scan line which matches the programmed light pen line position (line 3, 5, 7, or 9). **DADD10/UL** **Underline:** Asserted before the scan line which matches the programmed underline position (line 0 thru 15). **DADD11/BLINK** **Blink frequency:** Provides an output divided down from the vertical sync rate. **DADD12/ODD** **Odd Field:** Active high signal which is asserted before each scan line of the odd field when interlace is specified. **DADD13/LL** **Last Line:** Asserted before the last scan line of each character row.
V_{CC}	40	44	I	**Power Supply:** +5 volts ±5% power input.
GND	20	22	I	**Ground:** Signal and power ground input.

Operation Control

The operation control section decodes configuration and operation commands from the CPU and generates appropriate signals to other internal sections to control the overall device operation. It contains the timing and display registers which configure the display format and operating mode, the interrupt logic, and the status register which provides operational feedback to the CPU.

Timing

The timing section contains the counters and decoding logic necessary to generate the monitor timing outputs and to control the display format. These timing parameters are selected by programming of the initialization registers.

Display Control

The display control section generates linear addressing for up to 16K bytes of display memory. Internal comparators limit the portion of the memory which is displayed to programmed values. Additional functions performed in this section include cursor positioning, storage of light pen 'hit' location, and address comparisons required for generation of timing signals and the split screen interrupt.

Buffer Control

The buffer control section generates three signals which control the transfer of data between the CPU and the display buffer memory. Four system configurations requiring four different 'handshaking' schemes are supported. These are described below.

SYSTEM CONFIGURATIONS

Figure 1 illustrates the block diagram of a typical display terminal using the Signetics 2670, 2671, 2672, and 2673/2677 CRT terminal devices. In this system, the CPU examines inputs from the data communications line and the keyboard and places the data to be displayed in the display buffer memory. This buffer is typically a RAM which holds the data for a single or multiple screenload (page) or for a single character row.

The PVTC supports four common system configurations of display buffer memory: the independent, transparent, shared, and row buffer modes. The first three modes utilize a single or multiple page RAM and differ primarily in the means used to transfer display data between the RAM and the CPU. The row buffer mode makes use of a single row buffer (which can be a shift register or a small RAM) that is updated in real time to contain the appropriate display data.

The user programs bits 0 and 1 of IR0 to select the mode best suited for the system environment. The CNTRL1 – 3 outputs perform different functions for each mode and are named accordingly in the description of each mode.

Independent Mode

The CPU to RAM interface configuration for this mode is illustrated in figure 2. Transfer of data between the CPU and display memory is accomplished via a bidirectional latched port and is controlled by the signals read data buffer ($\overline{RDB}$), write data buffer ($\overline{WDB}$), and

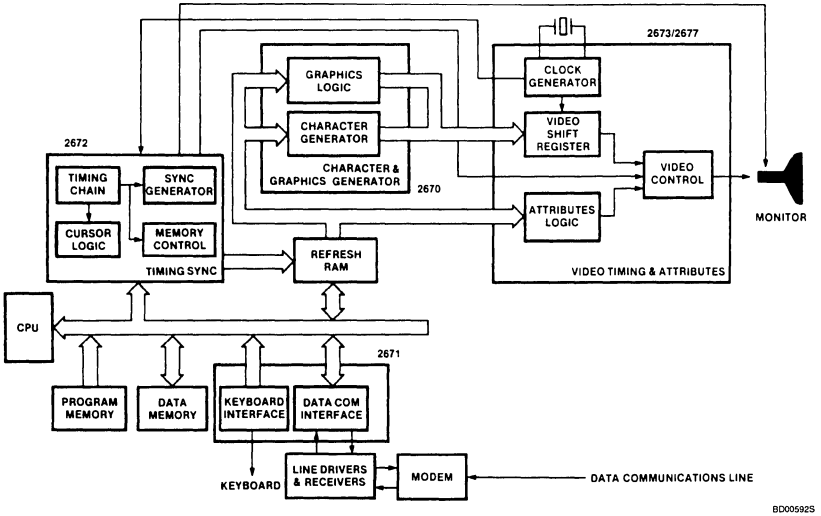


Figure 1. CRT Terminal Block Diagram

buffer chip enable ($\overline{BCE}$). This mode provides a non-contention type of operation that does not require address multiplexers. The CPU does not address the memory directly – the read or write operation is performed at the address contained in the cursor address register or the pointer address register as specified by the CPU. The PVTC enacts the data transfers during blanking intervals in order to prevent visual disturbances of the displayed data.

For a data buffer write command, the $\overline{WDB}$ signal will go active on the rising edge of character clock ($\overline{CCLK}$) and will remain so until the next $\overline{CCLK}$ rising edge. $\overline{BCE}$ is always in the active state except before and after a $\overline{WDB}$ command. When a write has been executed, $\overline{BCE}$ becomes inactive on the falling edge of $\overline{CCLK}$. When the write occurs ($\overline{WDB}$ active) on the next rising edge, $\overline{BCE}$ also becomes active. $\overline{BCE}$ and $\overline{WDB}$ both become inactive on the rising edge of $\overline{CCLK}$. $\overline{BCE}$ will then return to the active state on the next falling edge if there is no other write command to be executed.

The CPU manages the data transfers by supplying commands to the PVTC. The commands used are:

1. Read/Write at pointer address.
2. Read/Write at cursor address (with optional increment of address).
3. Write from cursor address to pointer address.

The operational sequence for a write operation is:

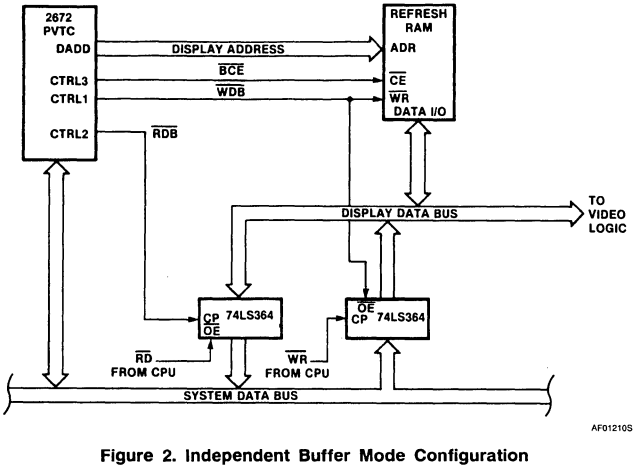
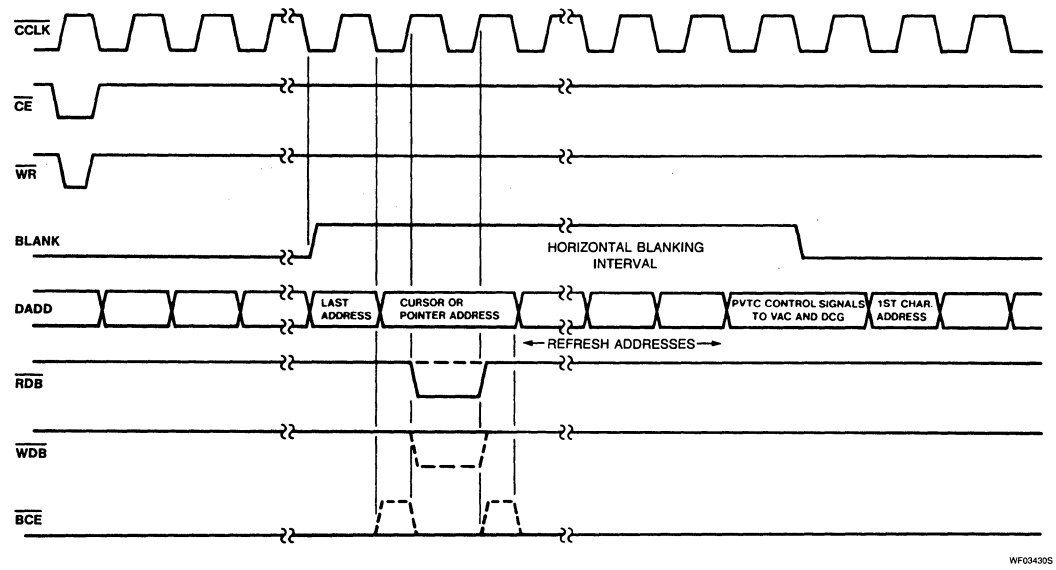


Figure 2. Independent Buffer Mode Configuration

1. CPU checks RDFLG status bit to assure that any previous operation has been completed.
2. CPU loads data to be written to display memory into the interface latch.
3. CPU writes address into cursor or pointer registers.
4. CPU issues 'write at cursor with/without increment' or 'write at pointer' command.
5. PVTC generates control signals and outputs specified address to perform requested operation. Data is copied from the interface latch into the memory.
6. PVTC sets RDFLG status to indicate that the write is completed.

Similarly, a read operation proceeds as follows:

1. Steps 1 and 3 as above.
2. CPU issues 'read at cursor with/without increment' or 'read at pointer' command.
3. PVTC generates control signals and outputs specified address to perform requested operation. Data is copied from memory to the interface latch and PVTC sets RDFLG status to indicate that the read is completed.



NOTE:
Write waveforms shown in dotted lines.

Figure 3. Read/Write at Cursor/Pointer Command Timing (Command Received During Active Display Window)

4. CPU checks RDFLG status to see if operation is completed.
5. CPU reads data from interface latch.

Loading the same data into a block of display memory is accomplished via the 'write from cursor to pointer' command:

1. CPU checks RDFLG status bit to assure that any previous operation has been completed.
2. CPU loads data to be written to display memory into the interface latch.
3. CPU writes beginning address of memory block into cursor address register and ending address of block into pointer address register.
4. CPU issues 'write from cursor to pointer' command.
5. PVTC generates control signals and outputs block addresses to copy data from the interface latch into the specified block of memory.
6. PVTC sets RDFLG status to indicate that the block write is completed.

Similar sequences can be implemented on an interrupt driven basis using the READY interrupt output to advise the CPU that a previously requested command has been completed.

Two timing sequences are possible for the 'read/write at cursor/pointer' commands. If the command is given during the active display window (defined as first scan line of the first character row to the last scan line of the last character row), the operation takes place during the next horizontal blanking interval, as illustrated in figure 3. If the command is given during the vertical blanking interval, or while the display has been commanded blanked, the operation takes place immediately. In the latter case, the execution time for the command is approximately one microsecond plus six (6) character clocks (see figure 4).

Timing for the 'write from cursor to pointer' operation is shown in figure 5. The BLANK output is asserted automatically and remains asserted until the horizontal retrace interval following completion of the command. The memory is filled at a rate of one location per two character times, plus a small amount of overhead.

Shared And Transparent Buffer Modes

In these modes the display buffer RAM is a part of the CPU memory domain and is addressed directly by the CPU. Both modes use the same hardware configuration with the CPU accessing the display buffer via three-state drivers (see figure 6). The processor bus request ($\overline{PBREQ}$) control signal informs the PVTC that the CPU is requesting access to the display buffer. In response to this request, the PVTC raises bus acknowledge ($\overline{BACK}$) until its bus external ($\overline{BEXT}$) output has freed the display address and data busses for CPU access. $\overline{BACK}$, which can be used as a 'hold' input to the CPU, is then lowered to indicate that the CPU can access the buffer.

In transparent mode, the PVTC delays the granting of the buffer to the CPU until a vertical or horizontal blanking interval, thereby causing minimum disturbance of the display. In shared mode, the PVTC will blank the display and grant immediate access to the CPU. Timing for these modes is illustrated in figures 7, 8, and 9.

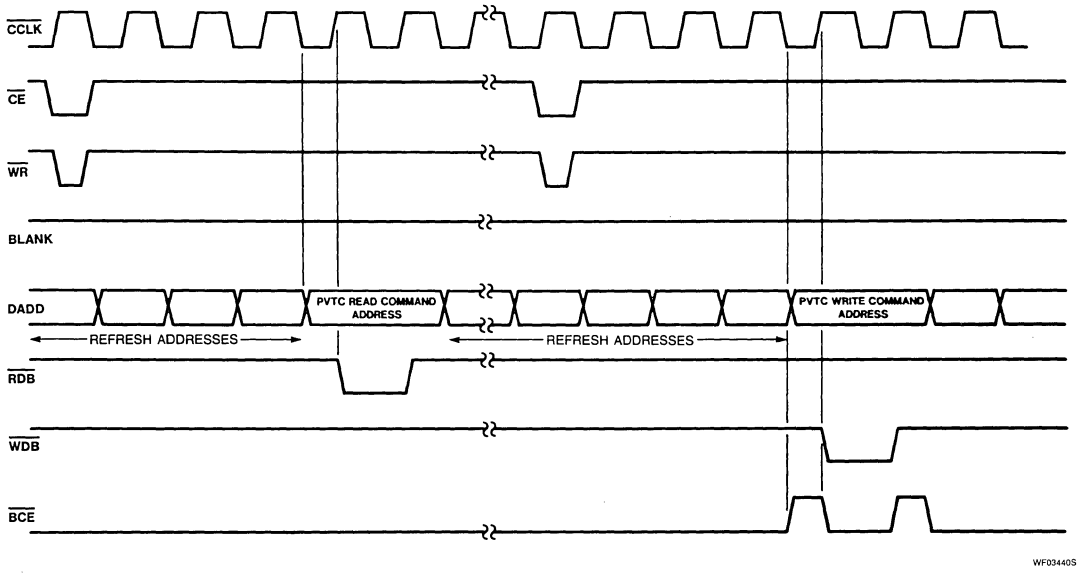


Figure 4. Read/Write at Cursor/Pointer Command Timing (Command Received While Display is Blanked)

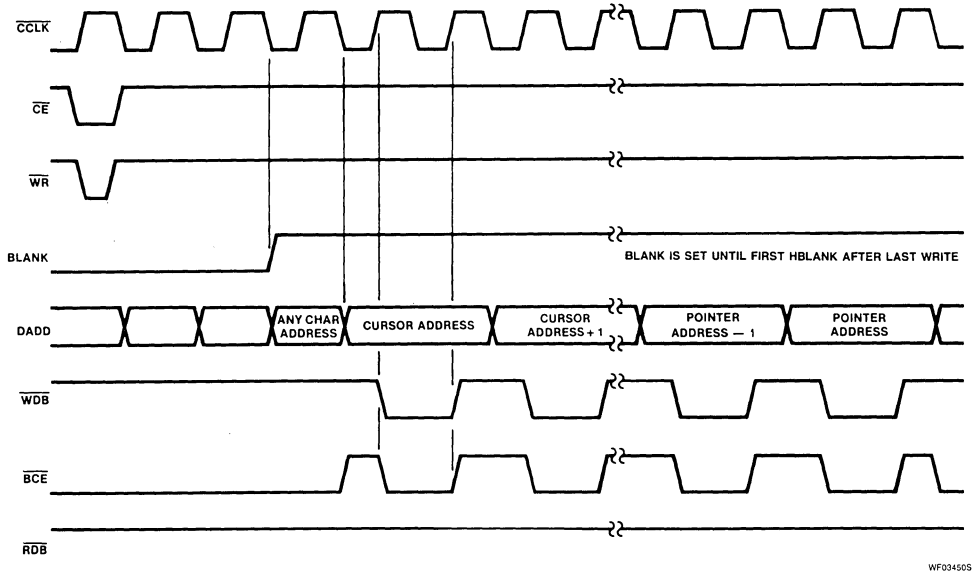


Figure 5. Write from Cursor to Pointer Command Timing

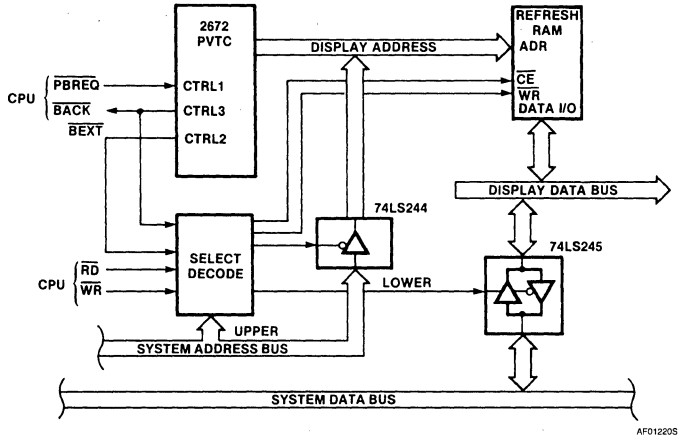
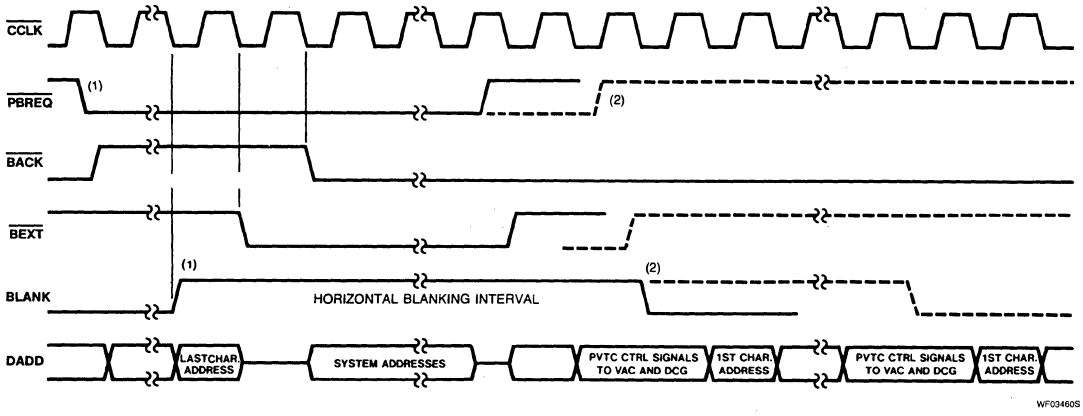
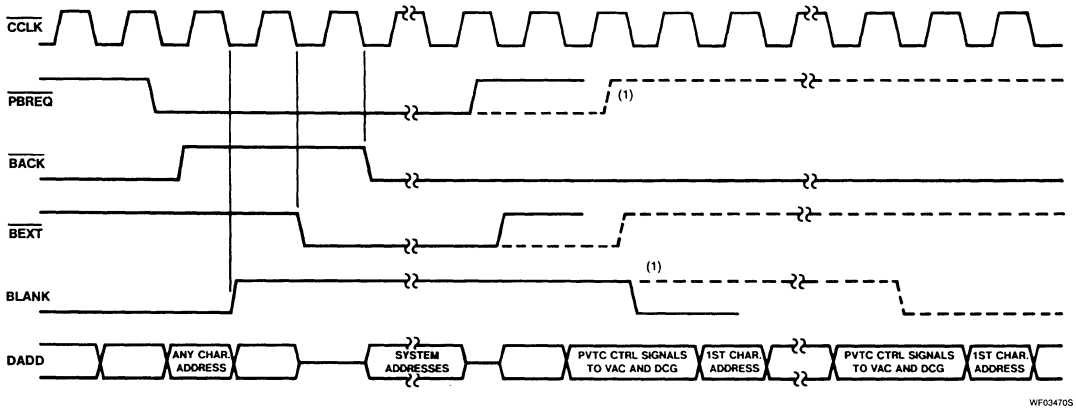


Figure 6. PVTC Shared or Transparent Buffer Modes



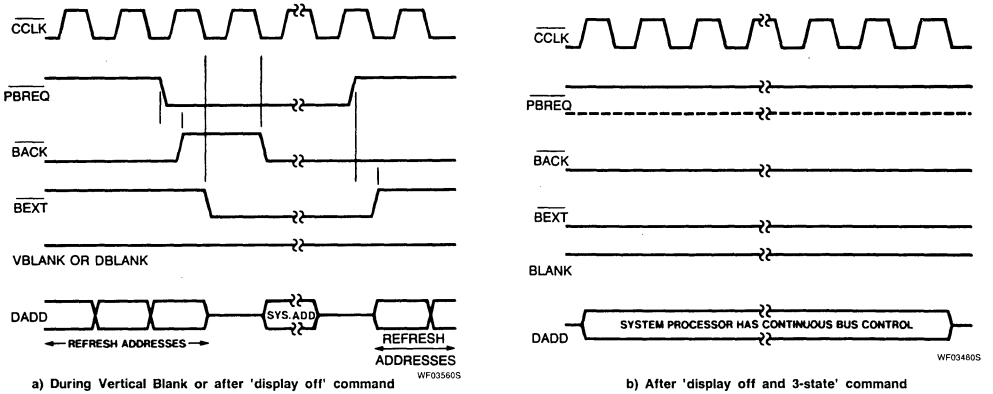
NOTES:
1. PBREQ must be asserted prior to the rising edge of BLANK in order for sequence to begin during that blanking period.
2. If PBREQ is negated after the next to last CCLK of the horizontal blanking interval, the next scan line will also be blanked.

Figure 7. Transparent Buffer Mode Timing



NOTE:
1. If $\overline{PBREQ}$ is negated after the next to last $\overline{CCLK}$ of the horizontal blanking interval, the next scan line will also be blanked.

Figure 8. Shared Buffer Mode Timing



a) During Vertical Blank or after 'display off' command

b) After 'display off and 3-state' command

Figure 9. Shared and Transparent Mode Timing

Row Buffer Mode

Figures 10 and 11 show the timing and a typical hardware implementation for the row buffer mode. During the first scan line (line 0) of each character row, the PVTC halts the CPU and DMA's the next row of character data from the system memory to the row buffer memory. The PVTC then releases the CPU and displays the row buffer data for the programmed number of scan lines. The bus request control ($\overline{BREQ}$) signal informs the CPU that character addresses and the memory bus control (MBC) signal will start at the next falling edge of BLANK. The CPU must release the address and data busses before this time to prevent bus contention. After the row of character data is transferred to the row buffer RAM, $\overline{BREQ}$ returns high to grant memory control back to the CPU.

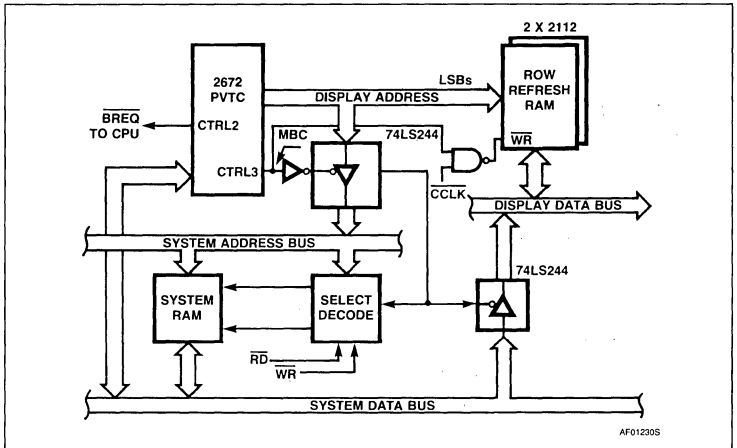


Figure 10. Row Buffer Mode Configuration

OPERATION

After power is applied, the PVTC will be in an inactive state. Two consecutive 'master reset' commands are necessary to release this circuitry and ready the PVTC for operation. Two register groups exist within the PVTC: the initialization registers and the display control registers. The initialization registers select the system configuration, monitor timing, cursor shape, display memory domain, and screen format. These are loaded first and normally require no modification except for certain special visual effects. The display control registers specify the memory address of the base character (upper left corner of screen), the cursor position, and the pointer address for independent memory access mode. These usually require modification during operation.

After initial loading of the two register groups, the PVTC is ready to control the monitor screen. Prior to executing the PVTC commands which turn on the display and cursor, the user should load the display memory with the first data to be displayed. During operation, the PVTC will sequentially address the display memory within the limits programmed into its registers. The memory outputs character codes to the system character and graphics generation logic, where they are converted to the serial video stream necessary to display the data on the CRT. The user effects changes to the display by modifying the contents of the display memory, the PVTC display control and command registers, and the initialization registers, if required. Interrupts and status conditions generated by the PVTC supply the 'handshaking' information necessary for the CPU to effect the display changes in the proper time frame.

Initialization Registers

There are 11 initialization registers (IR0 – IR10) which are accessed sequentially via a single address. The PVTC maintains an internal pointer to these registers which is incremented after each write at this address until the last register (IR10, the split screen register) is accessed. The pointer then continues to point to the split screen register. Upon power-up or a master reset command, the internal pointer is reset to point to the first register (IR0) of the initialization register group. The internal pointer can also be preset to any register of the group via the 'load IR address pointer' command. These registers are write only and are used to specify parameters such as the system configuration, display format, cursor shape, and monitor timing. Register formats are shown in table 2.

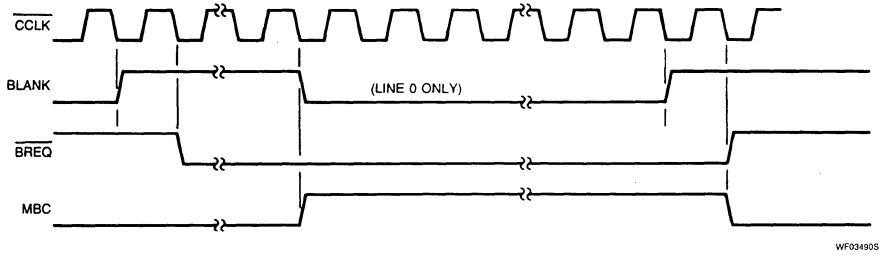


Figure 11. Row Buffer Mode Timing

2

Table 2. INITIALIZATION REGISTER BIT FORMATS

IR0

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	SCAN LINES PER CHARACTER ROW				SYNC SELECT	BUFFER MODE SELECT	
	NON-INTERLACED		INTERLACED				
NOT USED	0000 = 1 LINE 0001 = 2 LINES 0010 = 3 LINES • • 1110 = 15 LINES 1111 = 16 LINES		0000 = UNDEFINED 0001 = 5 LINES 0010 = 7 LINES • • 1110 = 31 LINES 1111 = UNDEFINED		0 = VSYNC 1 = CSYNC	00 = INDEPENDENT 01 = TRANSPARENT 10 = SHARED 11 = ROW	

IR1

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
INTERLACE ENABLE	EQUALIZING CONSTANT						
0 = NON-INT 1 = INTER.	00000000 = 1 CCLK 00000001 = 2 CCLK • • 11111110 = 127 CCLK 11111111 = 128 CCLK		CALCULATED FROM: $EC = 0.5(H_{ACT} + H_{FP} + H_{SYNC} + H_{BP}) - 2(H_{SYNC})$				

IR2

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	HORIZONTAL SYNC WIDTH				HORIZONTAL BACK PORCH		
NOT USED	0000 = 2 CCLK 0001 = 4 CCLK • • 1110 = 30 CCLK 1111 = 32 CCLK				000 = 1 CCLK 001 = 5 CCLK • • 110 = 25 CCLK 111 = 29 CCLK		

IR3

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
VERTICAL FRONT PORCH			VERTICAL BACK PORCH				
000 = 4 SCAN LINES 001 = 8 SCAN LINES • • 110 = 28 SCAN LINES 111 = 32 SCAN LINES			00000 = 4 SCAN LINES 00001 = 6 SCAN LINES • • 11110 = 64 SCAN LINES 11111 = 66 SCAN LINES				

IR4

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CHARACTER BLINK RATE	ACTIVE CHARACTER ROWS PER SCREEN (NOTE 1)						
0 = 1/16 VSYNC 1 = 1/32 VSYNC	00000000 = 1 ROW 00000001 = 2 ROWS • • 11111110 = 127 ROWS 11111111 = 128 ROWS						

NOTE:
1. In interlace mode with odd total character rows per screen the last character row will be the programmed scan lines per character row minus one.

Table 2. INITIALIZATION REGISTER BIT FORMATS (Continued)

IR5

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ACTIVE CHARACTERS PER ROW							
00000010 = 3 CHARACTERS 00000011 = 4 CHARACTERS • • 11111110 = 255 CHARACTERS 11111111 = 256 CHARACTERS							

IR6

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
FIRST LINE OF CURSOR				LAST LINE OF CURSOR			
0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15				0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15			

IR7

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LIGHT PEN LINE		CURSOR BLINK	DOUBLE HEIGHT CHAR.	UNDERLINE POSITION			
00 = SCAN LINE 3 01 = SCAN LINE 5 10 = SCAN LINE 7 11 = SCAN LINE 9		0= NO 1= YES	0= NO 1= YES	0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15			

IR8

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DISPLAY BUFFER FIRST ADDRESS LSB'S							
H'000' = 0 H'001' = 1 • • H'FFE' = 4,094 H'FFF' = 4,095				NOTE: MSB'S ARE IN IR9[3:0]			

IR9

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DISPLAY BUFFER LAST ADDRESS				DISPLAY BUFFER FIRST ADDRESS MSB'S			
0000 = 1,023 0001 = 2,047 • • 1110 = 15,359 1111 = 16,383				SEE IR8			

IR10

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CURSOR BLINK RATE	SPLIT SCREEN INTERRUPT ROW						
0 = 1/16 VSYNC 1 = 1/32 VSYNC	00000000 = ROW 0 00000001 = ROW 1 • • 11111110 = ROW 126 11111111 = ROW 127						

IR0[6:3] – Scan Lines Per Character Row
Both interlaced and non-interlaced scanning are supported by the PVTC. For interlaced mode, two different formats can be implemented, depending on the interconnection between the PVTC and the character generator (see IR1[7]). This field defines the number of scan lines used to compose a character row for each technique. As scanning occurs, the scan line count is output on the LA0 – LA3 and LI pins.

IR0[2] – VS/CS Enable
This bit selects either vertical sync pulses or composite sync pulses on the VSYNC/CSYNC output (pin 18). The composite sync waveform conforms to EIA RS170 standards, with the vertical interval composed of six equalizing pulses, six vertical sync pulses, and six more equalizing pulses.

IR0[1:0] – Buffer Mode Select
Four buffer memory modes may be selectively enabled to accommodate the desired system configuration. See System Configurations.

IR1[7] – Interface Enable
Specifies interlaced or noninterlaced timing operation. Two modes of interlaced operation are available, depending on whether LA0 – LA3 or LI, LA0 – LA2 are used as the line address for the character generator. The resulting displays are shown in figure 12.

For 'interlaced sync' operation, the same information is displayed in both odd and even fields, resulting in enhanced readability. The PVTC outputs successive line numbers in ascending order on the LA0 – LA3 lines, one per scan line for each field.

The 'interlaced sync and video' format doubles the character density on the screen. The PVTC outputs successive line numbers in ascending order on the LI, LA0 – LA2 lines, one per scan line for each field, but alternates beginning the count with even and odd line numbers. In the interlaced sync and video mode, the number of scan lines per character row is always odd. Assume that the first character row is row 0 (even). When in the odd field, the scan line numbers being displayed are even for even character rows and odd for odd character rows. When in the even field, the scan line numbers being displayed are odd for even character rows and even for odd character rows (see figure 12c). This provides balanced beam currents in the odd and even fields, thus minimizing character variations due to different loading of the CRT anode supply between fields.

IR1[6:0] – Equalizing Constant
This field indirectly defines the horizontal front porch and is used internally to generate the equalizing pulses for the RS170 compatible CSYNC. The value for this field is the total number of character clocks ($\overline{CCLK}$) during a horizontal line period divided by two, minus two times the number of character clocks in the horizontal sync pulse:

$$EC = \frac{H_{ACT} + H_{FP} + H_{SYNC} + H_{BP}}{2} - 2(H_{SYNC})$$

The definition of the individual parameters is illustrated in figure 13. The minimum value of H_{FP} is two character clocks.

Note that when using the 2673/2677 VAC, the blank pulse is delayed three CCLKs relative to the HSYNC pulse. Because of this delay, the actual HFP and HBP values will be different from the values programmed into the PVTC. The actual HFP will be decreased by 3 character clocks. The actual HBP will be increased by 3 character clocks.

IR2[6:3] – Horizontal Sync Pulse Width
This field specifies the width of the HSYNC pulse in $\overline{CCLK}$ periods.

IR2[2:0] – Horizontal Back Porch
This field defines the number of $\overline{CCLK}$s between the trailing edge of HSYNC and the trailing edge of BLANK.

IR3[7:5] – Vertical Front Porch
Programs the number of scan line periods between the rising edges of BLANK and VSYNC during a vertical retrace interval. The width of the VSYNC pulse is fixed at three scan lines.

IR3[4:0] – Vertical Back Porch
This field determines the number of scan line periods between the falling edges of the VSYNC and BLANK outputs.

IR4[7] – Character Blink Rate
Specifies the frequency for the character blink attribute timing. The blink rate can be specified as 1/16 or 1/32 of the vertical field rate. The timing signal has a duty cycle of 75% and is multiplexed onto the DADD11/BLINK output at the falling edge of each BLANK.

IR4[6:0] – Character Rows Per Screen
This field defines the number of character rows to be displayed. This value multiplied by the scan lines per character row, plus the vertical front and back porch values, and the vertical sync pulse width (three scan lines) is the vertical scan period in scan lines.

IR5[7:0] – Active Characters Per Row
This field determines the number of characters to be displayed on each row of the CRT screen. The sum of this value, the horizontal front porch, the horizontal sync width, and the horizontal back porch is the horizontal scan period in $\overline{CCLK}$s.

IR6[7:4], IR6[3:0] – First and Last Scan Line of Cursor
These two fields specify the height and position of the cursor on the character block. The 'first' line is the topmost line when scanning from the top to the bottom of the screen. The value of the first line of cursor must be less than the last line of cursor value.

IR7[7:6] – Light Pen Line Position
This field defines which of four scan lines of the character row will be used for the light pen strike – through attribute by the 2673/2677 VAC. The timing signal is multiplexed onto the DADD9/LPL output during the falling edge of BLANK.

IR7[5] – Cursor Blink Enable
This bit controls whether or not the cursor output pin will be blinked at the selected rate (IR10[7]). The blink duty cycle for the cursor is 50%.

IR7[4] – Double Height Character Row Enable
If enabled, the scan line count will be repeated twice in succession, causing the height of the character row to double. This bit can be changed at any time but will only become effective at the beginning of the character row following the time it is changed. This allows selected character rows to be of double height. The split screen interrupt can be used to notify the CPU when to effectuate changes to this bit. For each double height row which replaces a normal row, one row count should be subtracted from the 'character rows per screen' field (IR4) to maintain the same total number of scan lines per field.

IR7[3:0] – Underline Position
This field defines which scan line of the character row will be used for the underline attribute by the 2673/2677 VAC. The timing signal is multiplexed onto the DADD10/UL output during the falling edge of BLANK.

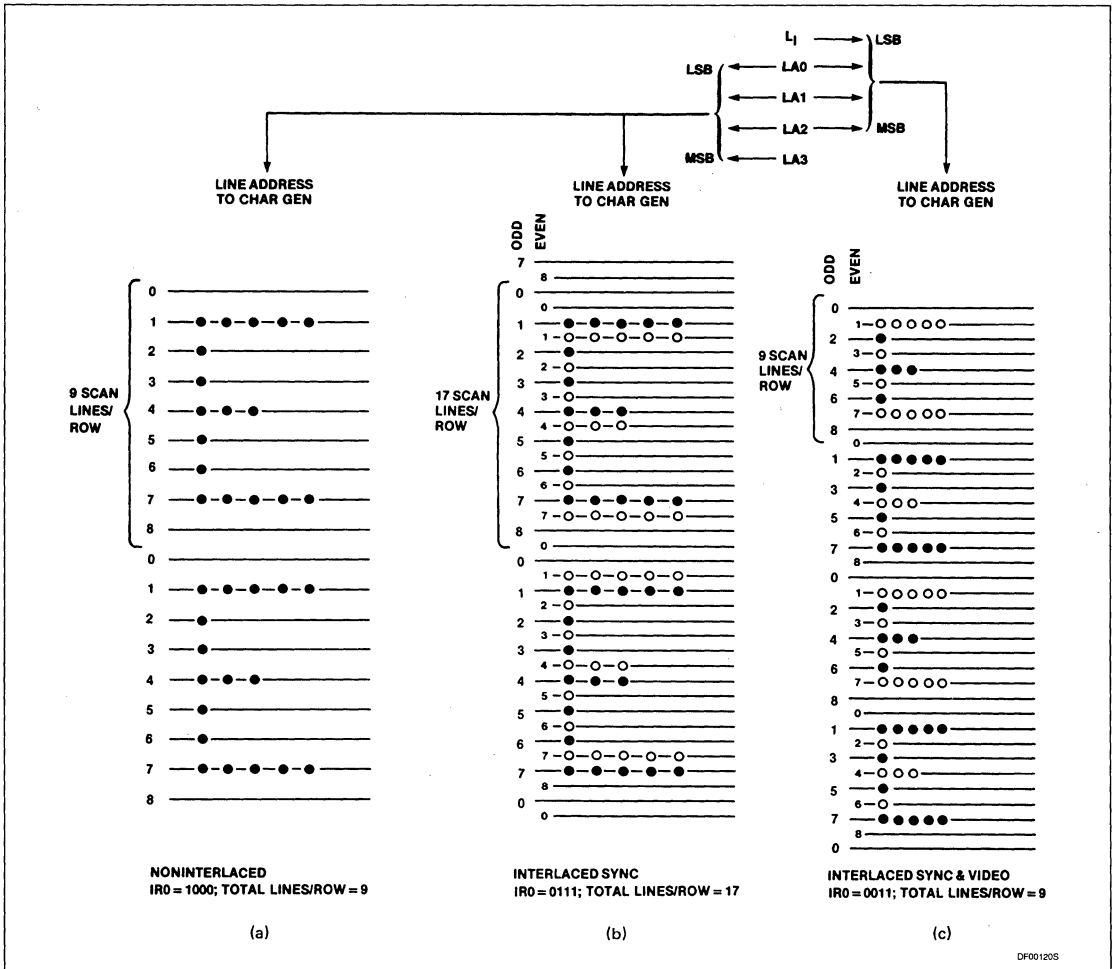


Figure 12. Interlaced Display Modes

IR9[3:0], IR8[7:0] – Display Buffer First Address

IR9[7:4] – Display Buffer Last Address
These two fields define the area within the buffer memory where the display data will reside. When the data at the 'display buffer last address' is displayed, the PVTC will wraparound and obtain the data to be displayed at the next screen position from the 'display buffer first address'. If 'last address' is the end of a character row and a new screen start address has been loaded into the screen start register, or if 'last address' is the last character position of the screen, the next data is obtained from the address contained in the screen start register.

Note that there is no restriction in displaying data from other areas of the addressable memory. Normally, the area between these two bounds is used for data which can be overwritten (e.g., as a result of scrolling), while data that is not to be overwritten would be contained outside these bounds and accessed by means of the split screen interrupt feature of the PVTC.

IR10[7] – Cursor Blink Rate
The cursor blink rate can be specified at 1/16 or 1/32 of the vertical scan frequency. Blink is effective only if blink is enabled by IR7[5].

IR10[6:0] – Split Screen Interrupt
The split screen interrupt can be used to provide special screen effects such as a row of double height characters or to change the normal addressing sequence of the display memory. The contents of this field is compared, in real time, to the current character row number. Upon a match, the PVTC sets the split screen status bit, and issues an interrupt request if so programmed. The status change/interrupt request is made at the beginning of scan line zero of the split screen character row.

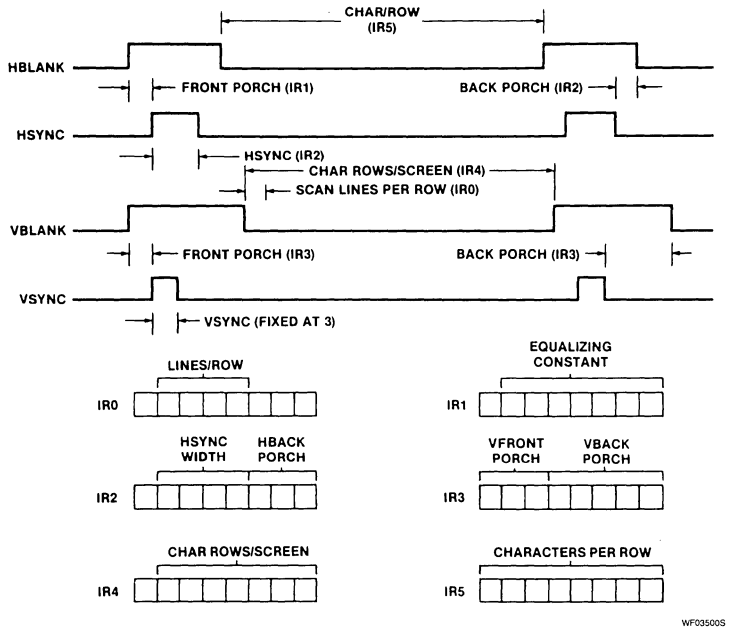


Figure 13. Horizontal and Vertical Timing

Timing Considerations

Normally, the contents of the initialization registers are not changed during operation. However, this may be necessary to implement special display features such as multiple cursors, smooth scrolling, horizontal scrolling, and double height character rows. Table 3 describes timing details for these registers which should be considered when implementing these features.

Table 3. TIMING CONSIDERATIONS

PARAMETER	TIMING CONSIDERATIONS
First line of cursor Last line of cursor Light pen line Underline	These parameters must be established at a minimum of two character times prior to their occurrence.
Double height characters	Set/reset during the character row prior to the affected row.
Cursor blink Cursor blink rate Character blank rate	New values become effective within one field after values are changed
Split screen interrupt row	Change anytime prior to line zero of desired row
Character rows per screen	Change only during vertical blanking period
Vertical front porch	Change prior to first line of V_{FP}
Vertical back porch	Change prior to fourth line after V_{SYNC}
Screen start register	Change prior to the horizontal blanking interval of the last line of character row prior to the affected row.

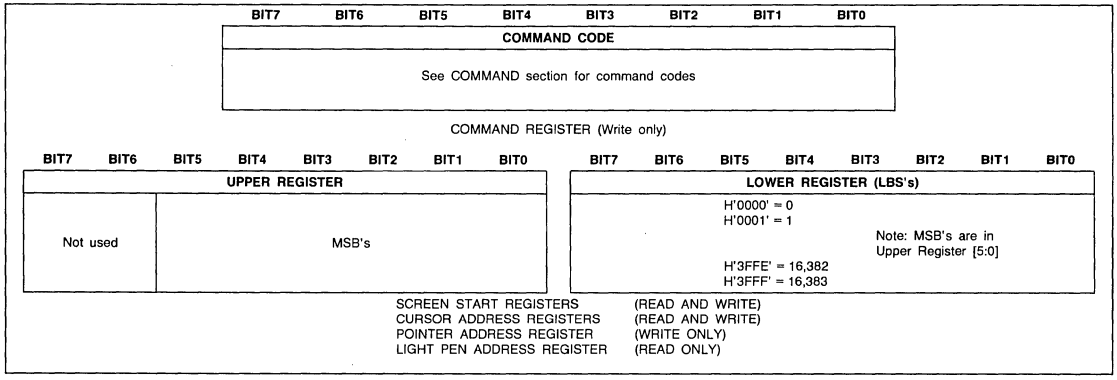
Display Control Registers

There are nine registers in this group, each with an individual address. Their formats are illustrated in table 4. The command register is used to invoke one of 16 possible PVTC commands as described in the COMMANDS section of this data sheet. The remaining registers in the group store address values which specify the cursor and buffer pointer locations, the location of the first character to be displayed on the screen, and the location of a light pen 'hit'. With the exception of the light pen register, the user initializes these registers after powering on the system and changes their values to control the data which is displayed.

Screen Start Registers

The screen start registers contain the address of the first character of the first row (upper left corner of the active display). At the beginning of the first scan line of the first row, this address is transferred to the row start register (RSR) and into the memory address counter (MAC). The counter is then advanced sequentially at the character rate the number of times programmed into the active characters per row register (IR5), thus reaching the address of the last character of the row plus one. At the beginning of each subsequent scan line of the first row, the MAC is reloaded from the RSR and the above sequence is repeated. At the end of the last scan line of the first row, the contents of the MAC are loaded into the RSR to serve as the starting memory address for the second character

Table 4. DISPLAY CONTROL REGISTER FORMATS



row. This process is repeated for the programmed number of rows per screen. Thus, the data in the display memory is displayed sequentially starting from the address contained in the screen start register. After the ensuing vertical retrace interval the contents of the screen registers are reloaded into the RSR and MAC and the process is repeated.

The sequential operation described above will be modified upon the occurrence of either of two events. First, if during the incrementing of the memory address counter the 'display buffer last address' (IR9[7:4]) is reached, the MAC will be loaded from the 'display buffer first address' register (IR9[3:0], IR8[7:0]), at the next character clock. Sequential operation will then resume starting from this address. This wraparound operation allows portions of the display buffer to be used for purposes other than storage of displayable data and is completely automatic without any CPU intervention (see figure 14a).

Second, the sequential row to row addressing can also be modified under CPU control. If the contents of the screen start register (upper, lower, or both) are changed during any character row (say row 'n'), the starting address of the next character row (row 'n + 1') will be the new value of the screen start register and addressing will continue sequentially from there. This allows features such as split screen operation, partial scroll, or status line display to be implemented. The split screen interrupt feature of the PVTC is useful in controlling this type of operation. Note that in order to obtain the correct screen display, the screen start register must be reloaded with the original value prior to the end of the vertical retrace. See figure 14b.

Refresh Addressing

During vertical blanking the address counter operation is modified by stopping the automatic load of the contents of the RSR into the counter, thereby allowing the address outputs to free-run. This allows dynamic memory refresh to occur during the vertical retrace interval. The refresh addressing starts at the last address displayed on the screen and increments by one for each character clock during the retrace interval. If the display buffer last address is encountered, wraparound will occur and refreshing will continue from the display buffer first address.

Cursor Address Registers

The contents of these registers define the buffer memory address of the cursor. If enabled, the cursor output will be asserted when the memory address counter (MAC) matches the value of the cursor address registers. The cursor address registers may be read or written by the CPU or incremented via the 'increment cursor address' command. In independent buffer mode, these registers define a buffer memory address for PVTC controlled access in response to 'read/write at cursor with/without increment' commands, or the first address to be used in executing the 'write from cursor to pointer' command.

Display Pointer Address Registers

These registers define a buffer memory address for PVTC controlled accesses in response to 'read/write at pointer' commands. They also define the last buffer memory address to be written for the 'write from cursor to pointer' command.

Light Pen Address Registers

If the light pen input is enabled, these registers are used to store the current character address upon receipt of a light pen strobe input. Several sources of delay between the display of a character upon the screen and the receipt of a light pen hit can be expected to exist in a system environment. These delays include address pipelining in the character generation circuits, delays in the video generation circuits, and delays in the light detection circuitry itself. These delays cause the value stored in the light pen register to differ from the actual address of the character at which the light pen hit actually was detected. Software must be used to correct this condition.

Interrupt/Status Registers

The interrupt and status registers provide information to the CPU to allow it to interact with the PVTC to effect desired changes to implement various display operations. The interrupt register provides information on five possible interrupting conditions, as shown in table 5. These conditions may be selectively enabled or disabled (masked) from causing interrupts by certain PVTC commands. An interrupt condition which is enabled (mask bit equal to one) will cause the $\overline{INTR}$ output to be asserted and will cause the corresponding bit in the interrupt register to be set-upon occurrence of the interrupting condition. An interrupt condition which is disabled (mask bit equal to zero) has no effect on either the $\overline{INTR}$ output or the interrupt register.

The status register provides six bits of status information: the five possible interrupting conditions plus the RDFLG bit. For this register, however, the contents are not affected by the state of the mask bits.

Descriptions of each interrupt/status register bit follow. Unless otherwise indicated, a bit, once set, will remain set until reset by the CPU by issuing a 'reset interrupt/status bits' command. The bits are also reset by a 'master reset' command and upon power-up. This bit is set to a one upon a master reset.

SR[5] – RDFLG

This bit is present in the status register only. A zero indicates that the PVTC is currently executing the previously issued command. A one indicates that the PVTC is ready to accept a new command.

Table 5. INTERRUPT AND STATUS REGISTER FORMAT

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
		RDFLG	VBLANK	LINE ZERO	SPLIT SCREEN	READY	LIGHT PEN
Not used always read as 0		0 = Busy 1 = Ready *	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = Busy 1 = Ready	0 = No 1 = Yes

NOTE:
*Status register only. Always 0 when reading interrupt register.

I/SR[4] – VBLANK
Indicates the beginning of a vertical blanking interval. Is set to a one at the beginning of the first scan line of the vertical front porch.

I/SR[3] – Line Zero
Is set to a one at the beginning of the first scan line (line 0) of each active character row.

I/SR[2] – Split Screen
This bit is set when a match occurs between the current character row number and the value contained in the split screen interrupts register, IR10[6:0]. The equality condition is only checked at the beginning of line zero of each character row. This bit is reset when either of the screen start registers is loaded by the CPU.

I/SR[1] – Ready
Certain PVTC commands affect the display and may require the PVTC to wait for a blanking interval before enacting the command. This bit is set to one when execution of the command has been completed. No command should be invoked until the prior command is completed. This bit is set to a zero upon a master reset.

I/SR[0] – Light Pen
A one indicates that a light pen hit has occurred and that the contents of the light pen register have been updated. This bit will be reset when either of the light pen registers is read.

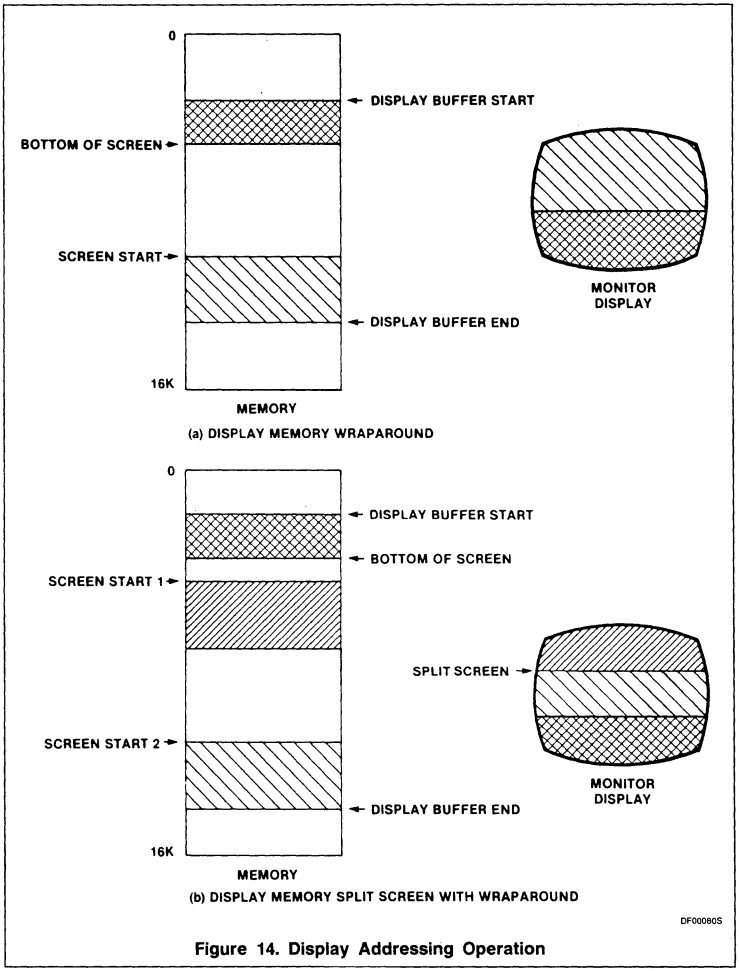


Figure 14. Display Addressing Operation

COMMANDS

The PVTC commands are divided into two classes: the instantaneous commands, which are executed immediately after they are invoked, and the delayed commands which may need to wait for a blanking interval prior to their execution. Command formats are shown in table 6. The commands are asserted by performing a write operation to the command register with the appropriate bit pattern as the data byte.

Instantaneous Commands

The instantaneous commands are executed immediately after the trailing edge of the $\overline{WR}$ pulse during which the command is issued. These commands do not affect the state of the RDFLG or READY interrupt/status bits.

However, a command should not be invoked if the RDFLG bit is low.

Master Reset
This command initializes the PVTC and may be invoked at any time to return the PVTC to its initial state. Upon power-up, two successive master reset commands must be applied to release the PVTC's internal power on circuits. In transparent and shared buffer modes, the CNTRL1 input must be high when the command is issued. The command causes the following:

1. VSYNC and HSYNC are driven low for the duration of RESET and BLANK goes high. BLANK remains high until a 'display on' command is received.
2. The interrupt and status bits and masks are set to zero, except for the RDFLG flag which is set to a one.
3. The transparent mode, cursor off, display off, and light pen disable states are set.
4. The initialization register pointer is set to address IR0.

Load IR Address
This command is used to preset the initialization register pointer with the value 'V' defined by D3 – D0. Allowable values are 0 to 10.

Enable Light Pen
After invoking this command, receipt of a light pen strobe input will cause the light pen register to be loaded with the current buffer memory address and the corresponding interrupt and status flag to be set. Once loaded, further loads are inhibited until either one of the light pen registers are read or a reset function is performed.

Disable Light Pen
Light pen hits will not be recognized.

Display Off
Asserts the BLANK output. The DADD0 through DADD13 display address bus outputs may be optionally placed in the three-state condition by setting bit 2 to a '1' when invoking the command.

Display On
Restores normal blanking operation either at the beginning of the next field (bit 2 = 1) or at the beginning of the next scan line (bit 2 = 0). Also returns the DADD0-DADD13 drivers to their active state.

Cursor Off
Disables cursor operation. Cursor output is placed in the low state.

Cursor On
Enables normal cursor operation.

Reset Interrupt/Status Bits
This command resets the designated bits in the interrupt and status registers. The bit positions correspond to the bit positions in the registers:

Bit 0 – Light pen

Bit 1 – Ready

Bit 2 – Split screen

Bit 3 – Line zero

Bit 4 – Vertical blank

Disable Interrupts
Sets the interrupt mask to zeros for the designated conditions, thus disabling these conditions from asserting the $\overline{INTR}$ output. Bit position correspondence is as above.

Enable Interrupts
Resets the selected interrupt and status register bits and writes the associated interrupt mask bits to a one. This enables the corresponding conditions to assert the $\overline{INTR}$ output. Bit position correspondence is as above.

Delayed Commands

This group of commands is utilized for the independent buffer mode of operation, although the 'increment cursor' command can also be used in other modes. With the exception of the 'write from cursor to pointer' and 'increment cursor' commands, all the commands of this type will be executed immediately or will be delayed depending on when the command is invoked. If invoked during the active screen time, the command is executed at the next horizontal blanking interval. If invoked during a vertical retrace interval or a 'display off' state, the command is executed immediately.

The 'increment cursor' and 'write from cursor to pointer' commands are executed immediately after they are issued. 'Increment cursor' requires approximately three $\overline{CCLK}$ periods for completion. 'Write from cursor to pointer' asserts the BLANK output during its execution. BLANK will not be released until the beginning of the horizontal blanking interval following the last write operation. This will allow more than one 'write from cursor to pointer' command to be executed during one frame and will blank the screen for the time required to execute the command.

In all cases, the PVTC will assert the READY/RDFLG status to signify completion of the command. No other commands should be given until the current command is completed. Therefore, the READY interrupt or RDYFLG status flag should be used for handshaking control between the PVTC and CPU when using these commands.

Read/Write at Pointer
Transfers data between the display buffer the bus interface latch using the address contained in the pointer register.

Table 6. PVTC COMMAND FORMATS

D7	D6	D5	D4	D3	D2	D1	D0	COMMAND
Instantaneous Commands:								
0	0	0	0	0	0	0	0	Master reset
0	0	0	1	V	V	V	V	Load IR pointer with value V (V = 0 to 10)
0	0	1	d	d	d	1	0[1]	Disable light pen
0	0	1	d	d	d	1	1[2]	Enable light pen
0	0	1	d	1	N	d	0[1]	Display off. Float DADD bus if N = 1
0	0	1	d	1	N	d	1[2]	Display on: Next field (N = 1) or scan line (N = 0)
0	0	1	1	d	d	d	0[1]	Cursor off
0	0	1	1	d	d	d	1[2]	Cursor on
0	1	0	N	N	N	N	N	Reset interrupt/status: Bit reset where N = 1
1	0	0	N	N	N	N	N	Disable interrupt: Disable where N = 1
0	1	1	N	N	N	N	N	Enable interrupt: Enables interrupts and resets the corresponding interrupt/status bits where N = 1
			V B	L Z	S S	R D	L P	
Delayed Commands:								**Hex**
1	0	1	0	0	1	0	0	A4 — Read at pointer address
1	0	1	0	0	0	1	0	A2 — Write at pointer address
1	0	1	0	1	0	0	1	A9 — Increment cursor address
1	0	1	0	1	1	0	0	AC — Read at cursor address
1	0	1	0	1	0	1	0	AA — Write at cursor address
1	0	1	0	1	1	0	1	AD — Read at cursor address and increment address
1	0	1	0	1	0	1	1	AB — Write at cursor address and increment address
1	0	1	1	1	0	1	1	BB — Write from cursor address to pointer address

NOTES:
1. Any combination of these three commands is valid.
2. Any combination of these three commands is valid.
3. d = Don't care.

Read/Write at Cursor
Transfers data between the display buffer and the bus interface latch using the address contained in the cursor register.

Increment Cursor
Adds one (modulo 16K) to the cursor address register.

Read/Write at Cursor and Increment
Transfers data between the display buffer and the bus interface latch using the address contained in the cursor register and then adds one (modulo 16K) to the cursor address register.

Write from Cursor to Pointer
Writes the data contained in the bus interface latch into the block of display memory designated by the cursor address and pointer address registers, inclusive. After completion of the command, the pointer address will be unchanged, but the cursor register contents will be equal to the pointer address.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		0.2			V
V_{OL}	Output low voltage	$I_{OL} = 2.4mA$			0.4	V
V_{OH}	Output high voltage (except $\overline{INTR}$ output)	$I_{OH} = -200\mu A$	2.4			V
I_{IL}	Input leakage current	$V_{IN} = 0$ to V_{CC}	−10		10	μA
I_{LL}	Data bus three-state leakage current	$V_O = 0$ to V_{CC}	−10		10	μA
I_{OD}	$\overline{INTR}$ open drain output leakage current	$V_O = 0$ to V_{CC}			10	μA
I_{CC}	Power supply current				160	mA

NOTES:
1. Stresses above those listed under Absolute Maximum Rating may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying voltages greater than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND).
6. Typical values are at +25°C, typical processing parameters.
7. For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
8. Test condition for outputs: $C_L = 150pF$.
9. Timing is illustrated and specified to $\overline{WR}$ and $\overline{RD}$ inputs. Device may also be operated with $\overline{CE}$ as the 'strobing' input. In this case, all timing specifications apply referenced to falling and rising edges of $\overline{CE}$.
10. This specification requires that the $\overline{CE}$ input be negated (high) between read and/or write cycles.
11. $\overline{BCE}$, $\overline{WDB}$,and $\overline{RDB}$ delays track each other within 10nsec. Also, these output delays will tend to follow direction (min/max) of DADD0 − 13 delays.
12. These values were measured with a capacitance load of 150pF. To adjust the output delay,use the following correction factor: $50pF \leqslant C_L < 150pF$: −0.15ns/pF

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6,7,8]

PARAMETER		TEST CONDITIONS	LIMITS				UNIT
			2.7MHz		4.0MHz		
			Min	Max	Min	Max	
Bus timing (figure 15)[9]							
t_{AS}	A0-A2 set-up time to $\overline{WR}$,$\overline{RD}$ low		30		30		ns
t_{AH}	A0-A2 hold time from $\overline{WR}$,$\overline{RD}$ high		0		0		ns
t_{CS}	$\overline{CE}$ set-up time to $\overline{WR}$,$\overline{RD}$ low		0		0		ns
t_{CH}	$\overline{CE}$ hold time from $\overline{WR}$,$\overline{RD}$ high		0		0		ns
t_{RW}	$\overline{WR}$,$\overline{RD}$ pulse width		250		250		ns
t_{DD}	Data valid after $\overline{RD}$ low			200		200	ns
t_{DF}	Data bus floating after $\overline{RD}$ high			100		100	ns
t_{DS}	Data set-up time to $\overline{WR}$ high		150		150		ns
t_{DH}	Data hold time from $\overline{WR}$ high		10		5		ns
t_{CC}	High time from $\overline{CE}$ to $\overline{CE}$[10]						
	Consecutive commands		600		600		ns
	Other accesses		300		300		ns
$\overline{CCLK}$ timing (figures 16 and 17)							
t_{CCP}	$\overline{CCLK}$ period		370		250		ns
t_{CCH}	$\overline{CCLK}$ high time		125		100		ns
t_{CCL}	$\overline{CCLK}$ low time		125		100		ns
	Output delay from $\overline{CCLK}$ edge[12]						ns
t_{CCD1}	DADD0-13,MBC		40	175	40	150	ns
t_{CCD2}	BLANK,HSYNC,VSYNC/CSYNC, CURSOR, $\overline{BEXT}$,$\overline{BREQ}$,$\overline{BACK}$, $\overline{BCE}$,$\overline{WDB}$, $\overline{RDB}$[11]		40	225	40	200	ns
Other timings (figures 17 and 18)							
t_{RDL}	READY/RDFLG low from $\overline{WR}$ high[9]			t_{CCP} + 30		t_{CCP} + 30	ns
t_{BAK}	$\overline{BACK}$ high from $\overline{PBREQ}$ low			225		200	ns
t_{BEXT}	$\overline{BEXT}$ high from $\overline{PBREQ}$ high			225		200	ns
t_{LPS}	Light pen strobe set-up time to $\overline{CCLK}$ low		120		120		ns
t_{LPH}	Light pen strobe hold time from $\overline{CCLK}$ low		−10		−10		ns
t_{IRL}	$\overline{INTR}$ low from $\overline{CCLK}$ low			225		200	ns
t_{IRH}	$\overline{INTR}$ high from $\overline{WR}$,$\overline{RD}$ high[9]			600		600	ns

2

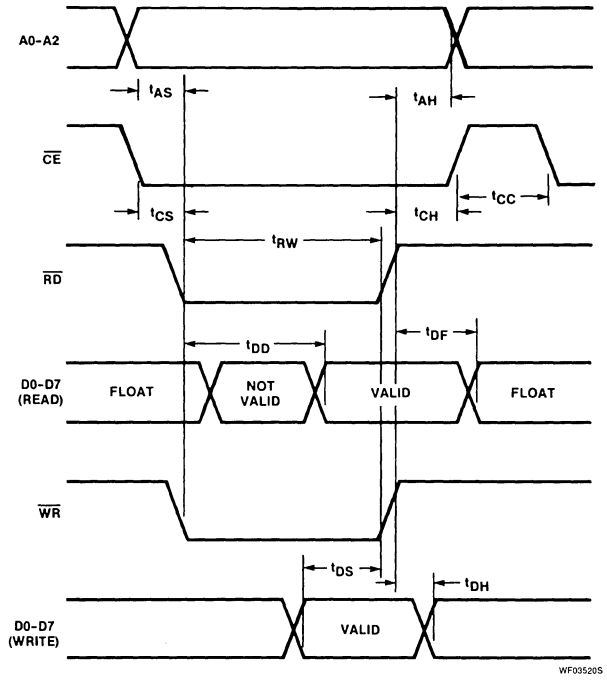
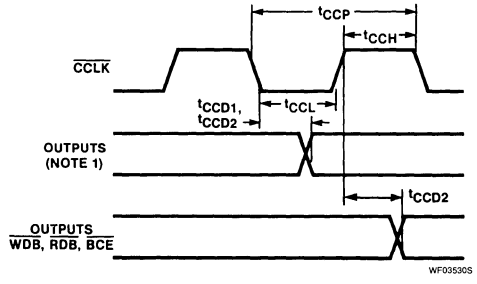


Figure 15. Bus Timing



NOTES:
1. DADD0-DADD13, BLANK, HSYNC, CSYNC/VSYNC, CURSOR, BEXT, BREQ, BCE, MBC, BACK.
2. BCE changes state on both CCLK edges — (see figures 3 and 4).

Figure 16. CCLK Timing

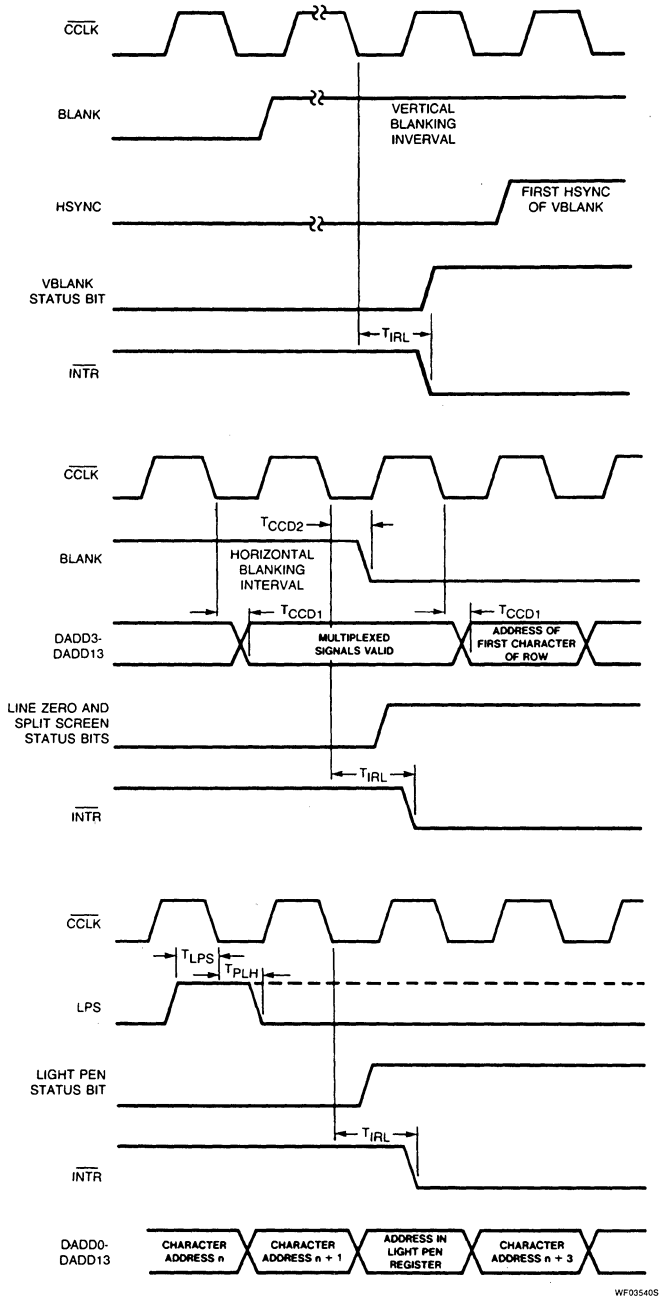


Figure 17. Other Timings

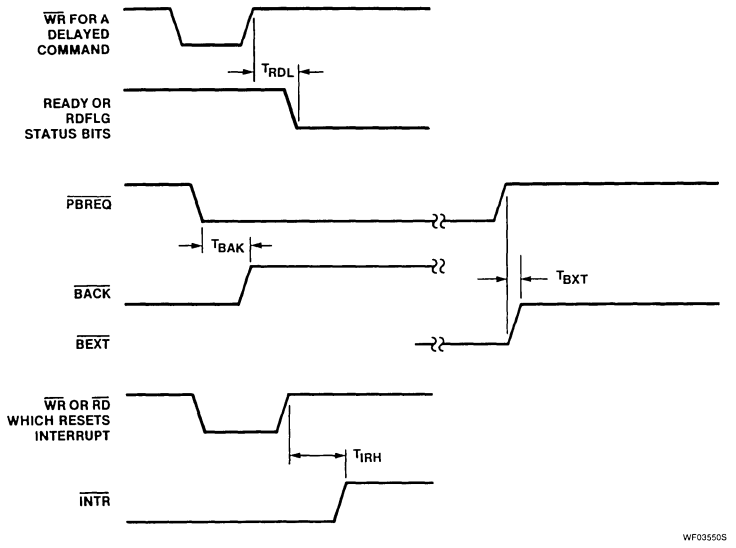
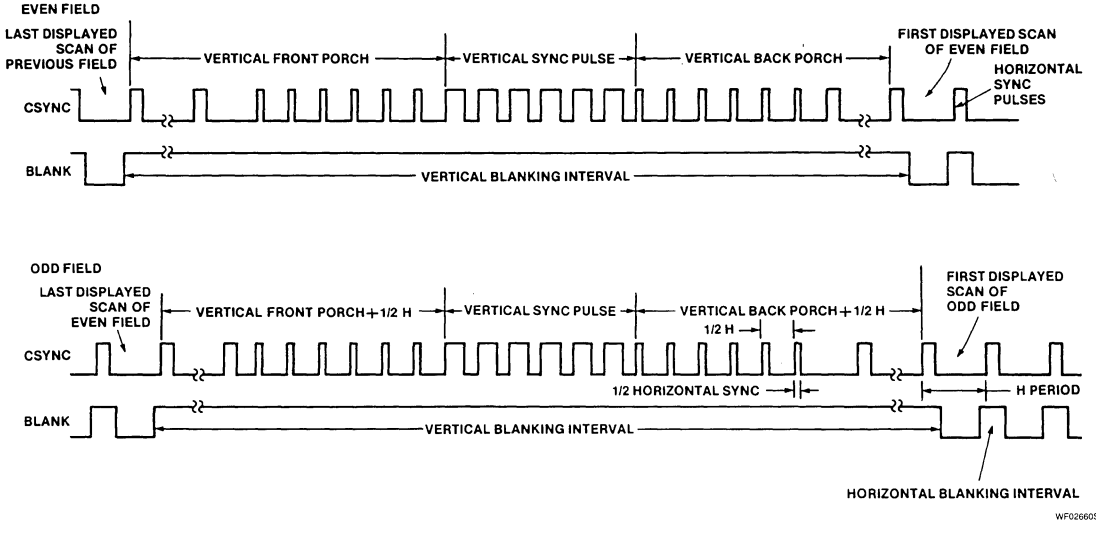


Figure 18. Other Timings



NOTES:
1. In non-interlaced operation the even field is repeated continuously, and the odd field is not.
2. In interlaced operation the even field alternates with the odd field.

Figure 19. Composite Sync Timing

Signetics

SCB2673
Video Attributes Controller (VAC)

Product Specification

Microprocessor Products

DESCRIPTION

The Signetics 2673A and 2673B Video Attributes Controllers (VAC) are bipolar LSI devices designed for CRT terminals and display systems that employ raster scan techniques. Each contains a high speed video shift register, field and character attributes logic, attribute latch, cursor format logic and half dot shift control.

The VAC provides control of visual attributes on a field or character by character. Internal logic preserves field attribute data from character row to character row so that an attribute byte is not required at the beginning of each row. The 2673B provides for reverse video, blank (non-display), blink, underline and highlight attributes and a graphics mode attribute to work in conjunction with the Signetics 2670 Display Character and Graphics Generator (DCGG). The 2673A substitutes a light pen (strike-thru) attribute for the graphics attribute.

The horizontal dot frequency is the basic timing input to the VAC. Internally, this clock is divided down to provide a character clock output for system synchronization. Up to ten bits of video dot data are parallel loaded into the video shift register on each character boundary. The video data is shifted out on three outputs at the dot frequency. On the VIDEO output, the data is presented as a three level signal representing low, medium and high intensities. The three intensities are also encoded on two TTL compatible video outputs. Light or dark screen background can be selected.

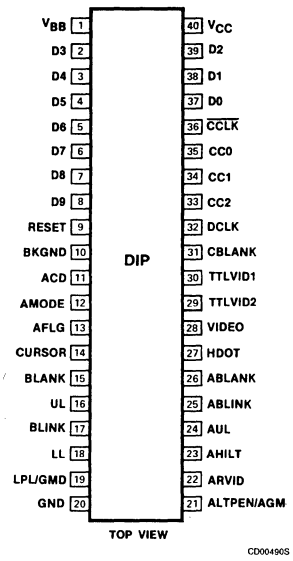
FEATURES

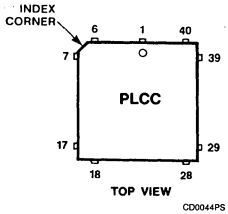
- **18 and 25MHz video dot rate**
- **Three level current driven video output**
- **Three level encoded TTL video outputs**
- **Character/field attribute logic:**
  - **Reverse video**
  - **Character blank**
  - **Character blink**
  - **Underline**
  - **Highlight**
  - **Light pen strike-thru or graphics control**
- **Field attributes extend from row to row**
- **Light or dark field**
- **Cursor reverse video logic**
- **Up to 10 dots per character**
- **Composite blanking for light field retrace**
- **Optional field graphics control output**
- **High speed bipolar design**
- **TTL compatible**
- **Compatible with Signetics 2672 PVTC and 2670 DCGG**

APPLICATIONS

- **CRT terminals**
- **Word processing systems**
- **Small business computers**

PIN CONFIGURATION





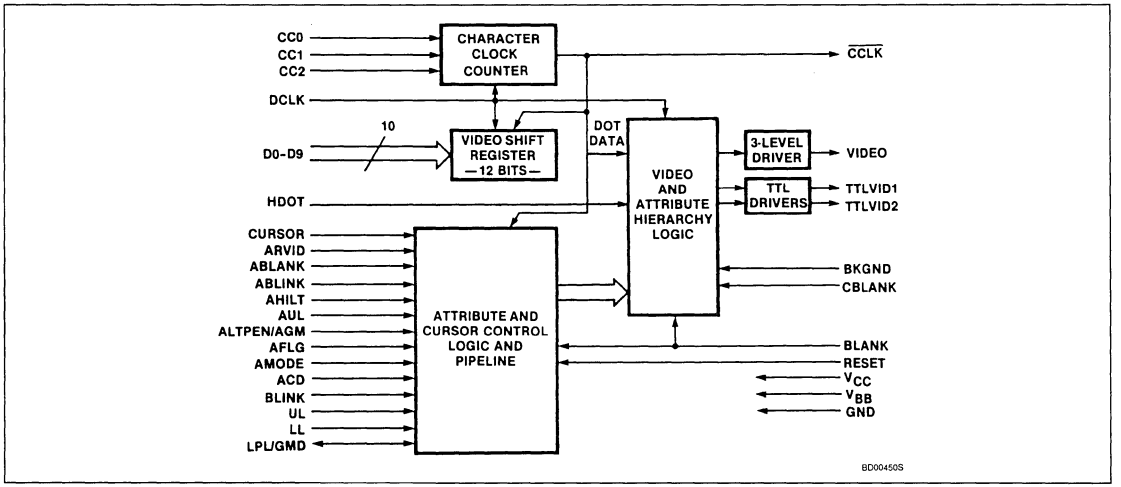
Pin	Function	Pin	Function
1	NC	23	NC
2	V_{BB}	24	ALTPEN/AGM
3	D3	25	ARVID
4	D4	26	AHILT
5	D5	27	AUL
6	D6	28	ABLINK
7	D7	29	ABLANK
8	D8	30	HDOT
9	D9	31	VIDEO
10	RESET	32	TTLVID2
11	BKGND	33	TTLVID1
12	NC	34	NC
13	ACD	35	CBLANK
14	AMODE	36	DCLK
15	AFLG	37	CC2
16	CURSOR	38	CC1
17	BLANK	39	CC0
18	UL	40	$\overline{CCLK}$
19	BLINK	41	D0
20	LL	42	D1
21	LPL/GMD	43	D2
22	GND	44	V_{CC}

Video Attributes Controller (VAC) SCB2673

ORDERING CODE

PACKAGES	V_{CC} = 5V ±5%, T_A = 0°C to 70°C			
	Graphics Attribute		Light Pen Attribute	
	25MHz	18MHz	25MHz	18MHz
Ceramic DIP	SCB2673BC5I40	SCB2673BC8I40	SCB2673AC5I40	SCB2673AC8I40
Plastic DIP	SCB2673BC5N40	SCB2673BC8N40	SCB2673AC5N40	SCB2673AC8N40
Plastic LCC	SCB2673BC5A44	SCB2673BC8A44	SCB2673AC5A44	SCB2673AC8A44

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
DCLK	32	36	I	**Dot Clock:** Dot frequency input. Video shift output rate.
$\overline{CCLK}$	36	40	O	**Character Clock:** A submultiple of DCLK. The frequency ranges from one sixth to one twelfth of DCLK, as determined by the state of the CC0 – CC2 inputs.
CC2 – CC0	33 – 35	37 – 39	I	**Character Clock Control:** The logic state on these three static inputs determine the internal divide factor for the $\overline{CCLK}$ output rate. Character clock rates of 6 through 12 dots per character may be specified.
D0 – D9	37 – 39, 2 – 8	41 – 43, 3 – 9	I	**Dot Data Input:** These are parallel inputs corresponding to the character graphic symbol dot data for a given scan line. These inputs are strobed into the video shift register on the falling edge of each character clock.
HDOT	27	30	I	**Half Dot Shift:** When this input is high, the serial video output is delayed by one half dot time. This input is latched on the falling edge of each character clock.
CURSOR	14	16	I	**Cursor Timing:** This input provides the timing for the cursor video. When high, effectively reverses the intensities of the video and attributes. Cursor position, shape, and blink rate are controlled by this input.
BKGND	10	11	I	**Background Intensity:** Specifies light or dark video during BLANK and character fields. Affects the intensities of all attributes.
BLANK	15	17	I	**Screen Blank:** When high, this input forces the video ouputs to the level specified by the BKGND input (either high or low intensity). Not effective when CBLANK is high.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
CBLANK	31	35	I	**Composite Blank:** Used with the TTL video outputs only. When high, this input forces the video outputs to a low intensity state for retrace blanking. When BKGND input is low, or when using video outputs, this input may be tied low.
ARVID	22	25	I	**Reverse Video Attribute:** The intensity of the associated character or field video is reversed. All other attributes are effectively reversed.
AHILT	23	26	I	**Highlight Attribute:** All dot video (including underline) of the associated character or field is highlighted with respect to the BKGND input and the reverse video attribute.
ABLANK	26	29	I	**Blank Attribute:** Generates a blank space in the associated character or field. The blank space intensity is determined by the BKGND input, the reverse video attribute, and the CURSOR input.
ABLINK	25	28	I	**Blink Attribute:** The associated character or field video is driven to the intensity determined by BKGND and the reverse video attribute when the BLINK input is high.
AUL	24	27	I	**Underline Attribute:** Specifies a line to be displayed on the character or field. The line is specified by the UL input. All other attributes apply to the underline video.
ALTPEN/ AGM	21	24	I	**Light Pen Attribute (2673A):** Specifies a highlighted line to be displayed on the character or field. The line is specified by the LPL input.
			I	**Attribute Graphics Mode (2673B):** This input is latched and synchronized to provide a field GMD output for the 2670 DCGG.
AMODE	12	14	I	**Attribute Mode:** Specifies character (AMODE = 0) or field (AMODE = 1) attributes mode.
AFLG	13	15	I	**Attributes Flag:** The VAC samples and latches the attributes inputs when this input is high. If field attributes are specified (AMODE = 1), the attributes are double buffered on a row basis. Thus, each scan line of every character row will start with the attributes that were valid at the end of the previous row.
ACD	11	13	I	**Attributes Control Display:** In field attributes mode (AMODE = 1), if ACD = 0, the first character in each new attribute field (the attribute control character) will be suppressed and only the attributes will be displayed. If ACD = 1, the first character and the attributes are displayed. This input has no effect in character mode (AMODE = 0).
BLINK	17	19	I	**Blink:** This input is sampled on the falling edge of BLANK to provide the blink rate for the character blink attribute. It should be a submultiple of the frame rate.
UL	16	18	I	**Underline:** Indicates the scan line(s) for the underline attribute. Latched on the falling edge of BLANK.
LPL/GMD	19	21	I	**Light Pen Line (2673A):** Indicates the scan line(s) for the light pen strike-thru attribute. Latched on the falling edge of BLANK.
			O	**Graphics Mode (2673B):** This output provides a synchronized, latched, field graphics mode corresponding to the AGM input. This output can be used to control the GM input on the 2670 DCGG.
LL	18	20	I	**Last Line:** Indicates the last scan line of each character row. Used internally to extend field attributes across row boundaries. Latched on the falling edge of BLANK. This input has no effect in character mode (AMODE = 0).
VIDEO	28	31	O	**Video:** A three level serial video output which corresponds to the composite dot pattern of characters, attributes and cursor.
TTLVID1	30	33	O	**TTL Video 1:** This output corresponds to the serial, non-highlighted video dot pattern.
TTLVID2	29	32	O	**TTL Video 2:** This output corresponds to the highlighted serial video dot pattern. Should be used with TTLVID1 to decode a composite video of three intensities.
RESET	9	10	I	**Manual Reset:** This active high input initializes the internal logic and resets the attribute latches.
V_{CC}	40	44	I	**Power Supply:** +5 Volts ±5%
V_{BB}	1	2	I	**Bias Supply:** See figure 13.
GND	20	22	I	**Ground:** 0V reference

FUNCTIONAL DESCRIPTION

The VAC consists of four major sections (see block diagram). The high speed dot clock input is divided internally to provide a character clock for system timing. The parallel dot data is loaded into the video shift register on each character boundary and shifted into the video logic block at the dot rate. The six attribute inputs are latched internally and combined with the serial dot data to provide a three level video source for the monitor.

A separate BLANK input defines the active screen area. When BLANK = 0, the video levels are derived internally by the combinations of dot data, attributes, cursor, and the state of the BKGND input. Either black or white background can be selected. Symbols (dot data) are normally gray and can be highlighted to white or black as shown in figure 1.

During the inactive screen area (BLANK = 1), the video level produced by the TTL outputs is either white (BKGND = 1) or black (BKGND = 0). A separate composite blank (CBLANK) input is provided to suppress raster retrace video when white background is specified. During the inactive screen area (BLANK = 1), the video level produced by the VIDEO output is either black (BKGND = 1) or white (BKGND = 0).

For the latter case, raster retrace video suppression is accomplished by raising the BKGND input during horizontal and vertical retrace intervals. For black background, tie BKGND high. Tie CBLANK input low for both cases.

Since BLANK is delayed by 3 $\overline{CCLK}$'s internally, CBLANK must be sync'ed with the internal BLANK signal to avoid active scan data from being suppressed. A CBLANK transition must occur at least 15ns before the rising edge of DCLK in order to be latched into the 2673 (see figure 8).

Table 1. CLOCK CONTROL INPUTS

			$\overline{CCLK}$	
CC2	CC1	CC0	Dots/Character	Duty Cycle*
0	0	0	6	3/3
0	0	1	6	3/3
0	1	0	7	4/3
0	1	1	8	4/4
1	0	0	9	5/4
1	0	1	10	5/5
1	1	0	11	6/5
1	1	1	12	6/6

* High/low

Character Clock Counter

The character clock counter divides the frequency on the DCLK input to generate the character clock ($\overline{CCLK}$). The divide factor is specified by the clock control inputs (CC0 – CC2) as shown in table 1.

Video Shift Register

On each character boundary, the parallel data (D0 – D9) is loaded into the video shift register. The data is shifted out least significant bit first (D0) by the DCLK. If 11 or 12 dots/character are specified (CC2 – CC0 = 110 or 111), a 0 (blank dot) is always shifted out before D0. For 12 dots/character, a 0 is also shifted out after D9. The serial dot data is shifted into the video logic where it is combined with the cursor and attributes to encode three levels of video.

Attribute and Cursor Control

The VAC visual attributes capabilities include: reverse video, character blank, blink, underline, highlight, and light pen strike-thru. The six attributes and the three attribute control inputs (AMODE, AFLG, and ACD) are clocked into the VAC on the falling edge of $\overline{CCLK}$. If AFLG is high, the attributes are latched internally and are effective for either one character time (AMODE = 0) or until another set of attributes is latched (AMODE = 1). The attributes set is double buffered on a row by row basis internally. Using this technique, field attributes can extend across character row boundaries thereby eliminating the necessity of starting each row with an attribute set.

When field attribute mode is selected, (AMODE = 1), the VAC will accommodate two attribute storage configurations. In one configuration, the attribute control data is stored in the refresh RAM, taking the place of the first character code in the field to be affected. For this mode, the ACD input is tied low and blank characters will be displayed in the screen positions occupied by the attribute data (see figure 11). In the second configuration, (ACD = 1), the character codes and attribute data are presented to the VAC in parallel. In this mode, dot data is displayed at each character position (see figure 12).

The CURSOR and the attribute input signals are pipelined internally to allow for system propagations (one $\overline{CCLK}$ for refresh RAM, one $\overline{CCLK}$ for dot generator). The attribute timing signals BLINK, UL, LPL and LL are clocked into the VAC at the beginning of each scan line by the falling edge of the BLANK input. Thus, these signals must be in their proper state at the falling edge of BLANK preceding the scan line at which they are to be active (see figure 4).

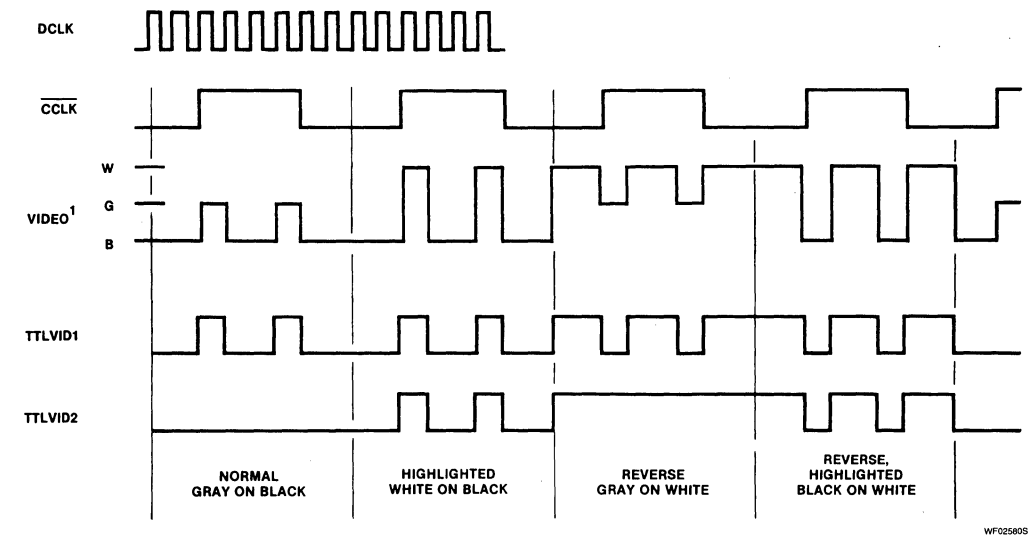


Figure 1. Encoded Video Outputs

Video Logic

The serial dot data and the pipelined cursor and attributes are combined to generate the three level current source on the VIDEO output. The three levels (white, gray, and black) are also encoded on the two TTL compatible outputs TTLVID1 and TTLVID2. The three levels are encoded as shown in table 2.

The video is normally shifted out on the leading edge of the DCLK. When the HDOT input is asserted, the corresponding dot data is delayed by one-half DCLK. This half dot shifting, when used on selected lines of character video, can be used to effect character rounding as shown in figure 2.

Attribute Hierarchy

The video of each character block consists of four components as shown in figure 3.

Table 2. VIDEO OUTPUT

TTLVID2	TTLVID1	INTENSITY
0	0	Black (or CBLANK)
0	1	Gray (on black surround)
1	0	Gray (on white surround)
1	1	White

NOTE:
The TTLVID1 output can be used independently to generate a two level non-highlighted video.

Symbol video is generated from the dot data inputs D0 – D9.

Underline video is enabled by the AUL attribute and is generated when the UL timing input is active. Underline and symbol video are always the same intensity.

Strike-thru video is enabled by the ALTPEN attribute and is generated when the LPL timing input is active. This video is always highlighted and takes precedence over the symbol and underline video. This feature applies to the 2673A only.

Surround video is the absence of symbol, underline and strike-thru video or the presence of the non-display attributes (ABLANK or ABLINK • BLINK).

The relative intensities of the four video components are determined by the remaining attributes (AHILT, ABLANK, ABLINK, ARVID) and the BKGND and CURSOR inputs as illustrated in table 3.

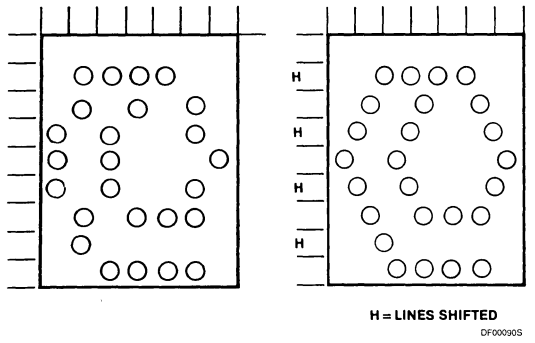


Figure 2. "AT" Symbol with and without Half Dot Shifting

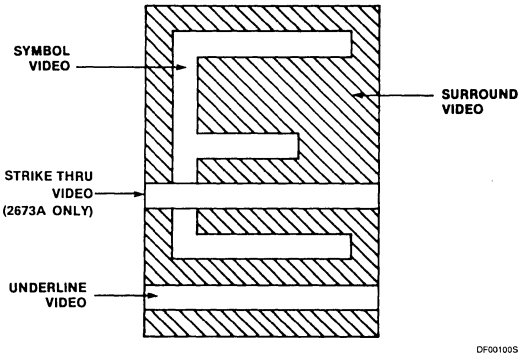


Figure 3. Video Components of Character Block

Table 3. ATTRIBUTES HIERARCHY

ATTRIBUTES AND CONTROL INPUTS d = don't care				RELATIVE VIDEO INTENSITIES W = White, B = Black, G = Gray		
BKGND[5]	Reverse[1]	Non-Display[2]	AHILT	Strike Thru Video[3]	Symbol Or Underline Video[3,4]	Surround Video[3]
0	0	0	0	W	G	B
0	0	0	1	W	W	B
0	0	1	d	B	B	B
0	1	0	0	B	G	W
0	1	0	1	B	B	W
0	1	1	d	W	W	W
1	0	0	0	B	G	W
1	0	0	1	B	B	W
1	0	1	d	W	W	W
1	1	0	0	W	G	B
1	1	0	1	W	W	B
1	1	1	d	B	B	B

NOTES:
1. Reverse = ARVID • $\overline{\text{CURSOR}}$ + $\overline{\text{ARVID}}$ • CURSOR
2. Non-display = ABLANK + ABLINK • BLINK
3. See figure 3.
4. Symbol and underline video are always the same intensity.
5. Reverse sense for VIDEO output.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground	−0.5 to +6.0	V

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = See figure 14[3,4,5]

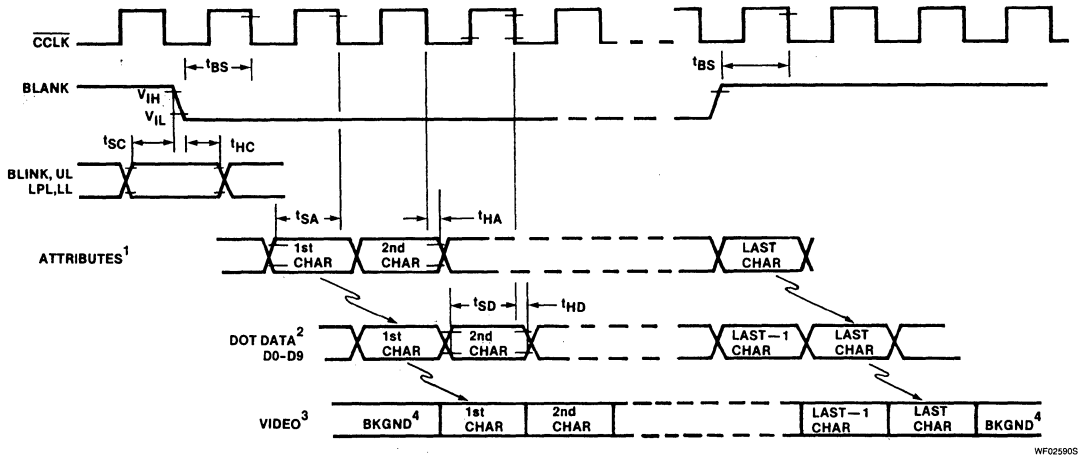
PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		2.0			V
V_{OL}	Output low voltage (except VIDEO)	$I_{OL} = 4mA$			0.4	V
V_{OH}	Output high voltage (except VIDEO)	$I_{OH} = -400\mu A$	2.4			V
V_B	VIDEO black level	$R_L = 150$ ohms to GND		0		V
V_G	VIDEO gray level	$R_L = 150$ ohms to GND		0.45		V
V_W	VIDEO white level	$R_L = 150$ ohms to GND		0.90		V
I_{IL}	Input low current	$V_{IN} = 0.4V$			−400/ −800[6]	μA
I_{IH}	Input high current	$V_{IN} = 2.4V$			20/40[6]	μA
I_{CC}	V_{CC} supply current	$V_{IN} = 0V$, V_{CC} = max			80	mA
I_{BB}	V_{BB} supply current	V_{BB} = max			120	mA

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = See figure 14[3,4,5]

PARAMETER		TEST CONDITIONS	LIMITS				UNIT
			25MHz Version		18MHz Version		
			Min	Max	Min	Max	
Dot clock (see figure 10)							
f_d	Frequency (HDOT = 0)			25		18	MHz
	(HDOT = 1)			18		18	MHz
t_{DH}	High		15		22		ns
t_{DL}	Low		15		22		ns
Set-up times to $\overline{CCLK}$							
(See figures 4, 5, 6 and 10)							
t_{BS}	BLANK		50		50		ns
t_{SC}	BLINK, UL, LPL, LL (ref to BLANK)		20		20		ns
t_{SA}	Attributes		45		55		ns
t_{SD}	Dot data D0 – D9		70		70		ns
t_{SK}	CURSOR		50		50		ns
t_{FS}	AFLG, AMODE		50		65		ns
t_{SH}	HDOT		45		55		ns
Hold times from $\overline{CCLK}$							
(See figures 4, 5, 6 and 10)							
t_{HC}	BLINK, UL, LPL, LL (ref to BLANK)		20		20		ns
t_{HA}	Attributes		20		20		ns
t_{HD}	Dot data D0 – D9		30		30		ns
t_{HK}	CURSOR		20		20		ns
t_{FH}	AFLG, AMODE		30		30		ns
t_{HH}	HDOT		20		20		ns
Set-up times to DCLK (See figure 9)							
t_{SG}	BKGND		15		15		ns
t_{SB}	CBLANK		15		15		ns
t_{CS}	CC0 – CC2		30		35		ns
Hold times from DCLK (See figure 9)							
t_{HG}	BKGND		15		15		ns
t_{HB}	CBLANK		15		15		ns
t_{CH}	CC0 – CC2		20		20		ns
Delay times (See figures 6 and 7)		$C_L = 150pF$					
t_{DGM}	GMD from DCLK			65		65	ns
t_{DC}	$\overline{CCLK}$ from DCLK			65		65	ns
t_{DV}[7]	TTLVID1 and TTLVID2 from DCLK			75		80	ns
t_{DV}	VIDEO from DCLK			240		240	ns

NOTES:

3. Parameters are valid over operating temperature range unless otherwise specified.
4. All voltage measurements are referenced to ground (V_{SS}). All input signals swing between 0.4V and 2.4V. All time measurements are referenced at input voltages of 0.8V, 2.0V and at output voltages of 0.8V, 2.0V as appropriate.
5. Typical values are at +25°C, typical supply voltages and typical processing parameters.
6. For DCLK input.
7. C_L less than 150pF minimum could be faster.



NOTES:
1. Attributes include: ABLINK, ABLANK, ARVID, AUL, AHILT, and ALTPEN.
2. One $\overline{CCLK}$ delay for dot data (obtained from delay through character generator).
3. See figure 7 for detail timing of VIDEO, TTLVID1, TTLVID2.
4. Non-active scan time. VIDEO reverts to polarity selected by the BKGND input.

Figure 4. VAC Pipeline Timing

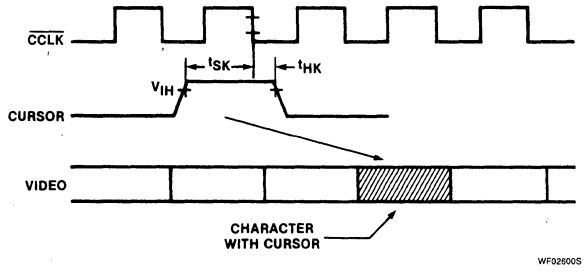
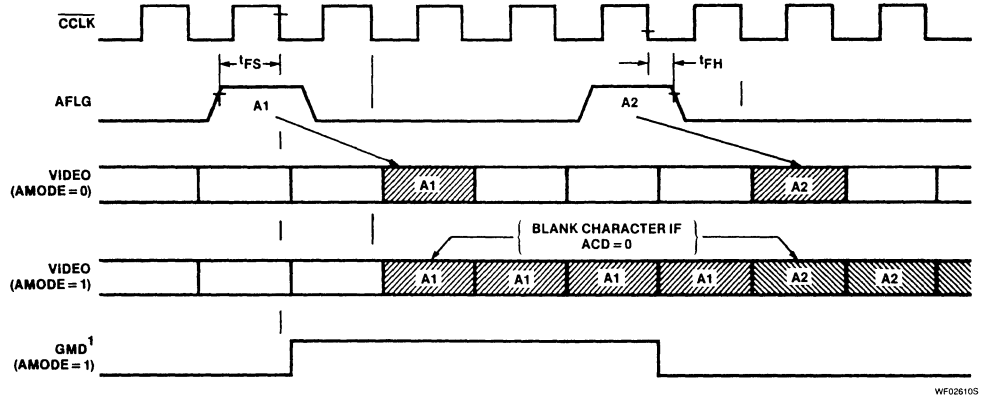
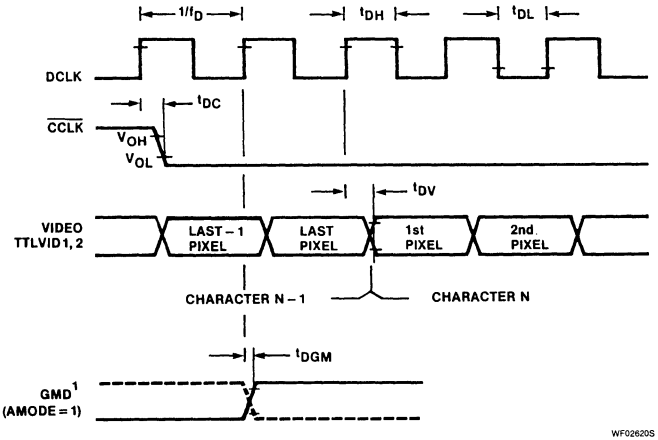


Figure 5. Cursor Pipeline Timing



NOTE:
1. GMD output in 2673B version only. See figure 7 for detail timing.

Figure 6. Character (AMODE = 0), Field (AMODE = 1), and GMD Attribute Timing



NOTE:
1. GMD output in 2673B version only.

Figure 7. Video and GMD Pipeline Timing

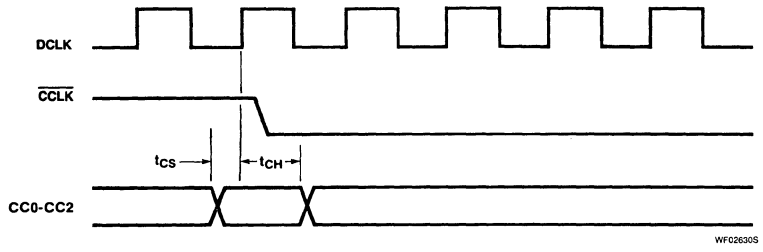


Figure 8. Character Clock Counter Timing

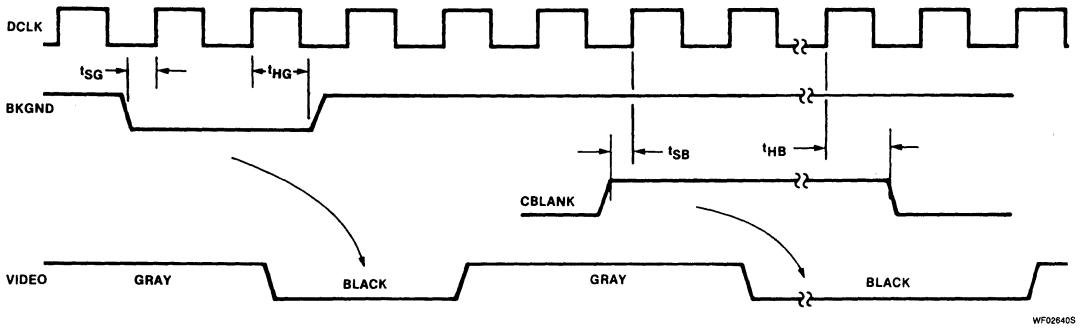
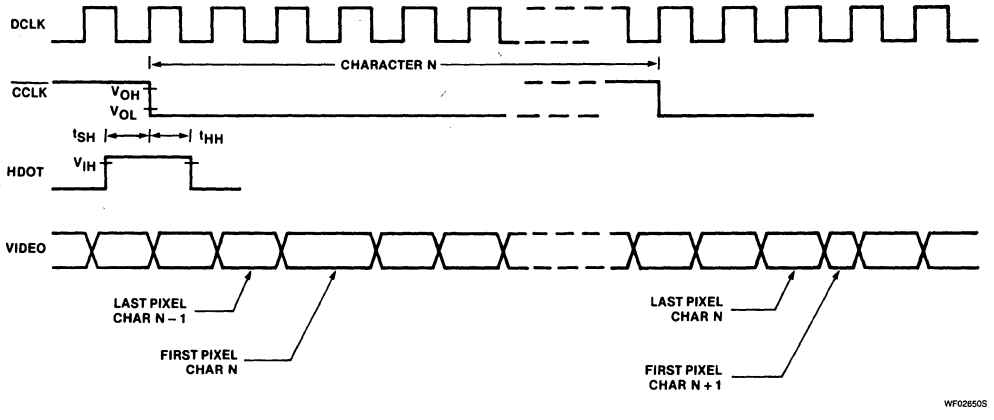


Figure 9. BKGND and RBLANK Timing During Inactive Scan Time (Blank = 1)



NOTE:
1. Half dot shift feature 18MHz maximum.

Figure 10. Half Dot Shift Timing

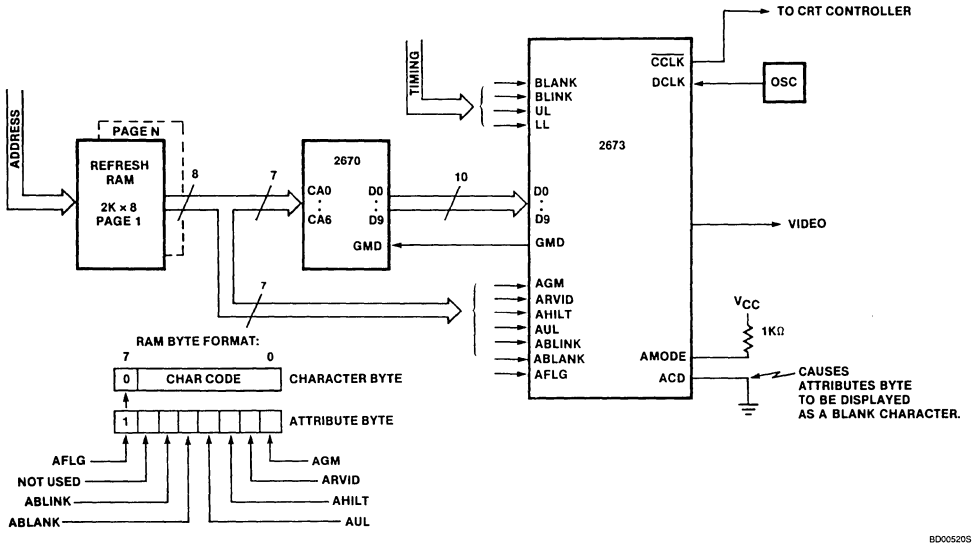
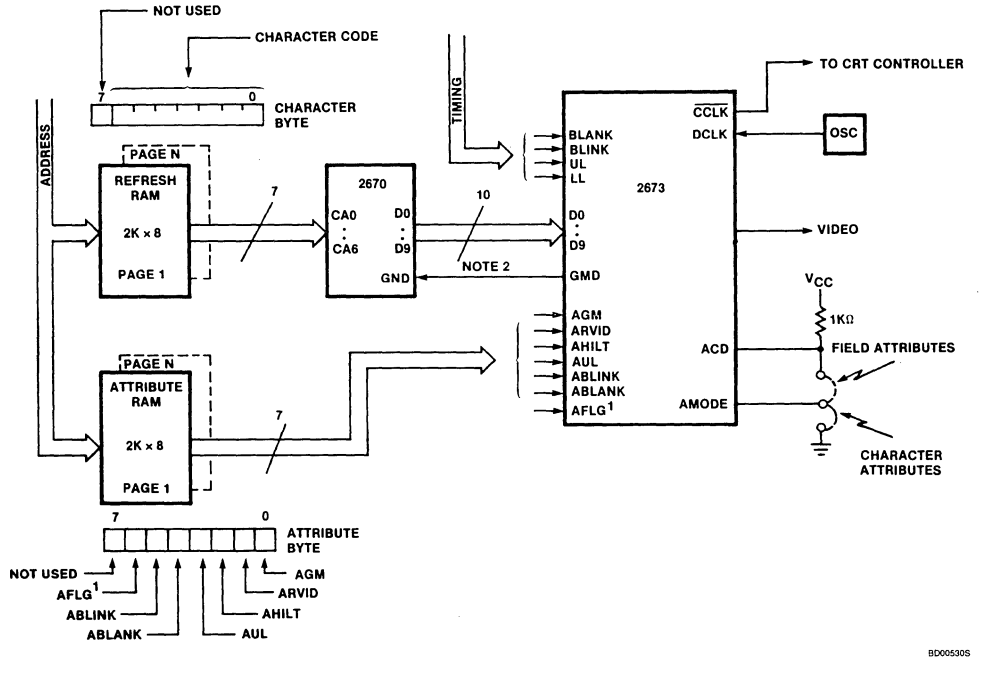


Figure 11. System Block Diagram of the 2673 in Field Attribute Mode Using the Narrow RAM (8 Wide) Configuration



NOTES:
1. For operation in character attribute mode, tie AFLG high.
2. In character attribute mode, AGM output from RAM should be connected directly to the 2670 GMD input.

Figure 12. System Block Diagram of the 2673 in Field or Character Attribute Mode Using the Wide RAM Configuration

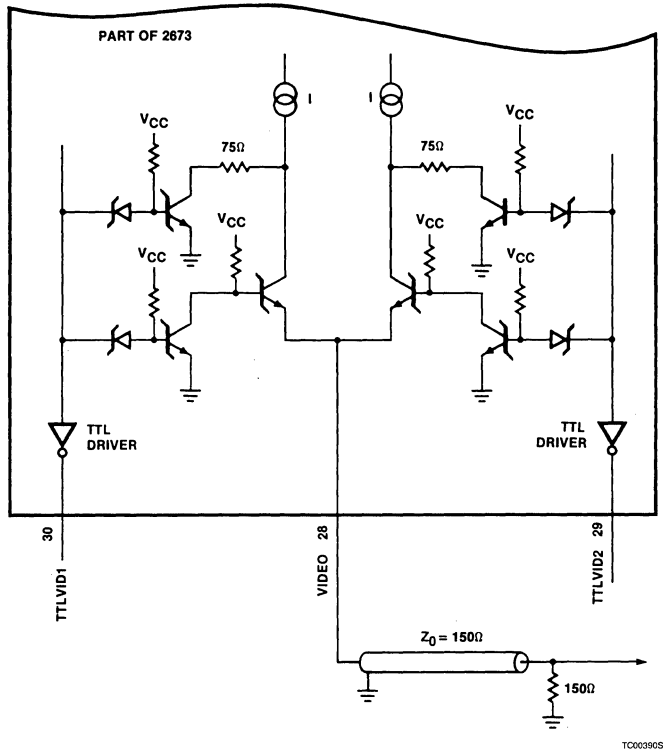


Figure 13. Video Output Stages of the 2673

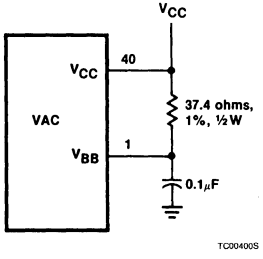


Figure 14. Test Diagram

SCN2674
Advanced Video Display Controller (AVDC)

Product Specification

Microprocessor Products

DESCRIPTION

The Signetics SCN2674 Advanced Video Display Controller (AVDC) is a programmable device designed for use in CRT terminals and display systems that employ raster scan techniques. The AVDC generates the vertical and horizontal timing signals necessary for the display of interlaced or non-interlaced data on a CRT monitor. It provides consecutive addressing to a user specified display buffer memory domain and controls the CPU display buffer interface for various buffer configuration modes. A variety of operating modes, display formats, and timing profiles can be implemented by programming the control registers in the AVDC.

A minimum CRT terminal system configuration consists of an AVDC, an SCN2671 Keyboard and Communication Controller (PKCC), an SCN2670 Display Character and Graphics Generator (DCGG), an SCB2675 Color/Monochrome Attributes Controller (CMAC), a single chip microcomputer such as the 8048, a display buffer RAM, and a small amount of TTL for miscellaneous address decoding, interface, and control. Typically, the package count for a minimum system is between 15 and 20 devices; system complexity can be enhanced by upgrading the microprocessor and expanding via the system address and data busses.

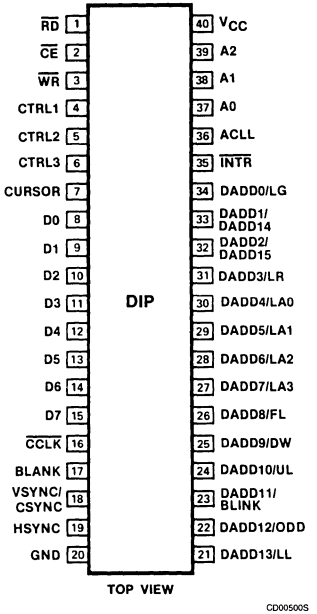
FEATURES

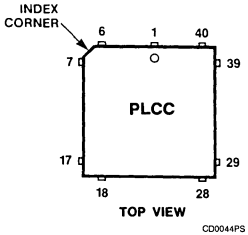
- **2.7MHz and 4MHz character rate**
- **1 to 256 characters per row**
- **1 to 16 raster lines per character row**
- **1 to 128 character rows per frame**
- **Bit mapped graphics mode**
- **Programmable horizontal and vertical sync generators**
  - **RS170 compatible sync**
- **Interlaced or non-interlaced operation**
- **Up to 64K RAM addressing for multiple page operation**
- **Readable, writable and incrementable cursor**
  - **Programmable cursor size and blink**
- **AC line lock**
- **Automatic wraparound of RAM**
- **Automatic split screen**
- **Automatic bidirectional soft scrolling**
  - **Programmable scan line increment**
- **Row table addressing mode**
- **Double height tops and bottoms**
- **Double width control output**
- **Selectable buffer interface modes**
- **Dynamic RAM refresh**
- **Completely TTL compatible**
- **Single +5 volt power supply**
- **Power on reset circuit**

APPLICATIONS

- **CRT terminals**
- **Word processing systems**
- **Small business computers**
- **Home computers**

PIN CONFIGURATION





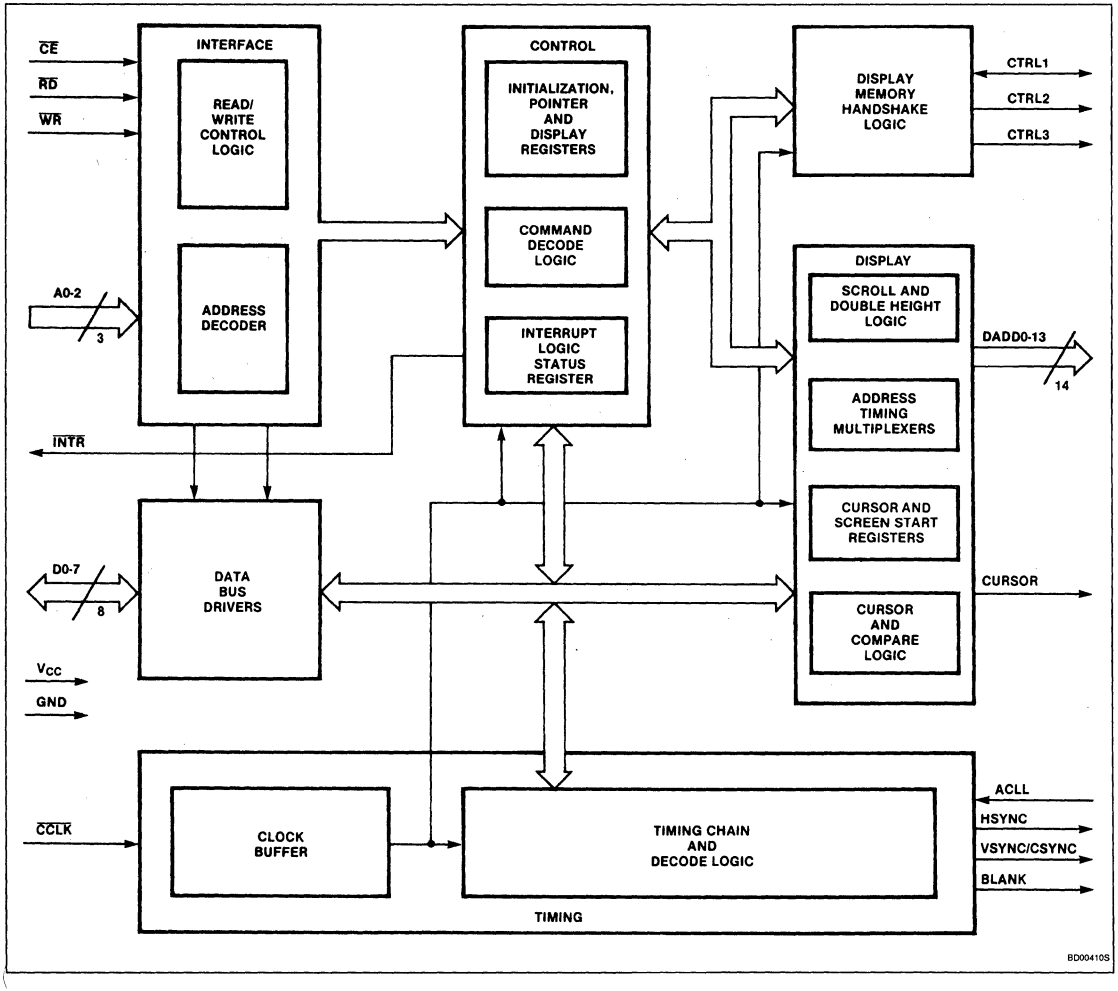
Pin	Function	Pin	Function	Pin	Function
1	NC	17	D7	31	DADD6/LA2
2	$\overline{RD}$	18	$\overline{CCLK}$	32	DADD5/LA1
3	$\overline{CE}$	19	BLANK	33	DADD4/LA0
4	$\overline{WR}$	20	VSYNC	34	NC
5	CTRL1		CSYNC/	35	DADD3/LR
6	CTRL2	21	HSYNC	36	DADD2/
7	CTRL3	22	GND		DADD15
8	CURSOR	23	NC	37	DADD1/
9	D0	24	DADD13/LL		DADD14
10	D1	25	DADD12/	38	DADD0/LG
11	D2		ODD	39	$\overline{INTR}$
12	NC	26	DADD11/	40	ACLL
13	D3		BLINK	41	A0
14	D4	27	DADD10/UL	42	A1
15	D5	28	DADD9/DW	43	A2
16	D6	29	DADD8/FL	44	V_{CC}
		30	DADD7/LA3		

Advanced Video Display Controller (AVDC) SCN2674

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C	
	4MHz	2.7MHz
Ceramic DIP	SCN2674BC4I40	SCN2674BC3I40
Plastic DIP	SCN2674BC4N40	SCN2674BC3N40
Plastic LCC	SCN2674BC4A44	SCN2674BC3A44

BLOCK DIAGRAM



2

PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
A0 – A2	37 – 39	41 – 43	I	**Address Lines:** Used to select AVDC internal registers for read/write operations and for commands.
D0 – D7	8 – 15	9 – 11 13 – 17	I/O	**8-Bit Bidirectional Three-State Data Bus:** Bit 0 is the LSB and bit 7 is the MSB. All data, command, and status transfers between the CPU and the AVDC take place over this bus. The direction of the transfer is controlled by the $\overline{RD}$ and $\overline{WR}$ inputs when the $\overline{CE}$ input is low. When the $\overline{CE}$ input is high, the data bus is in the three-state condition.
$\overline{RD}$	1	2	I	**Read Strobe:** Active low input. A low on this pin while $\overline{CE}$ is low causes the contents of the register selected by A0 – A2 to be placed on the data bus. The read cycle begins on the leading (falling) edge of $\overline{RD}$.
$\overline{WR}$	3	4	I	**Write Strobe:** Active low input. A low on this pin while $\overline{CE}$ is also low causes the contents of the data bus to be transferred to the register selected by A0 – A2. The transfer occurs on the trailing (rising) edge of $\overline{WR}$.
$\overline{CE}$	2	3	I	**Chip Enable:** Active low input. When low, data transfers between the CPU and the AVDC are enabled on D0 – D7 as controlled by the $\overline{WR}$, $\overline{RD}$, and A0 – A2 inputs. When $\overline{CE}$ is high, effectively, the AVDC is isolated from the data bus and D0 – D7 are placed in the three-state condition.
$\overline{CCLK}$	16	18	I	**Character Clock:** Timing signal derived from the video dot clock which is used to synchronize the AVDC's timing functions.
HSYNC	19	21	O	**Horizontal Sync:** Active high output which provides video horizontal sync pulses. The timing parameters are programmable.
VSYNC/ CSYNC	18	20	O	**Vertical Sync/Composite Sync:** A control selects either vertical or composite sync pulses on this active high output. When CSYNC is selected, equalization pulses are included. The timing parameters are programmable.
BLANK	17	19	O	**Blank:** This active high output defines the horizontal and vertical borders of the display. Display control signals which are output on DADD0 through DADD13 are valid on the trailing edge of BLANK.
CURSOR	7	8	O	**Cursor Gate:** This output becomes active for a specified number of scan lines when the address contained in the cursor register matches the address output on DADD0 through DADD13 for displayable character addresses. The first and last lines of the cursor and a blink option are programmable. When the row table addressing mode is enabled, this output is active for a portion of the blanking interval prior to the first scan line of a character row, while the AVDC is fetching the starting address for that row.
$\overline{INTR}$	35	39	O	**Interrupt Request:** Open drain output which supplies an active low interrupt request from any of five maskable sources. This pin is inactive after a power on reset or a master reset command.
ACLL	36	40	I	**AC Line Lock:** If this input is low after the programmed vertical front porch interval, the vertical front porch will be lengthened by increments of horizontal scan line times until this input goes high. This input should be pulled high if not being used.
CTRL1	4	5	I/O	**Handshake Control 1:** In independent mode, provides an active low write data buffer ($\overline{WDB}$) output which strobes data from the interface latch into the display memory. In transparent and shared modes, this is an active low processor bus request ($\overline{PBREQ}$) input which indicates that the CPU desires to access the display memory.
CTRL2	5	6	O	**Handshake Control 2:** In independent mode, provides an active low read data buffer ($\overline{RDB}$) output which strobes data from the display memory into the interface latch. In transparent and shared modes, this is an active low bus external enable ($\overline{BEXT}$) output which indicates that the AVDC has relinquished control of the display memory (DADD0 – DADD13 are in the three-state condition) in response to a CPU bus request. $\overline{BEXT}$ also goes low in response to a 'display off and float DADD' command. In row buffer mode, it is an active low bus request ($\overline{BREQ}$) output which halts the CPU during a line DMA.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
CTRL3	6	7	O	**Handshake Control 3:** In independent mode, provides the active low buffer chip enable ($\overline{BCE}$) signal to the display memory. In transparent and shared modes, provides an active low bus acknowledge ($\overline{BACK}$) output which serves as a ready signal to the CPU in response to a processor bus request. In row buffer mode, this is an active high memory bus control (MBC) output which configures the system for the DMA transfer of one row of character codes from system memory to the row display buffer.
DADD0–DADD13	34-21	38 – 35 33 – 24	O	**Display Address:** Used by the AVDC to address up to 16K of display memory directly, or up to 64K of memory by de-multiplexing DADD14 and DADD15. These outputs are floated at various times depending on the buffer mode. Various control signals are multiplexed on DADD0 through DADD13 and are valid at the trailing edge of BLANK. These control signals are: DADD0/LG **Line Graphics:** Output which denotes bit mapped graphic mode. DADD1/DADD14 **Display Address 14:** Multiplexed address bit used to extend addressing to 64K. DADD2/DADD15 **Display Address 15:** Multiplexed address bit used to extend addressing to 64K. DADD3/LR **Last Row:** Output which indicates the last active character row of each field. DADD4-DADD7/LA0-LA3 **Line Address:** Provides the number of the current scan line count for each character row. DADD8/FL **First Line:** Asserted during the blanking interval just prior to the first scan line of each character row. DADD9/DW **Double Width:** Output which denotes a double width character row. DADD10/UL **Underline:** Asserted during the blanking interval just prior to the scan line which matches the programmed underline position (line 0 through 15). DADD11/BLINK **Blink Frequency:** Provides an output divided down from the vertical sync rate. DADD12/ODD **Odd Field:** Active high signal which is asserted before each scan line of the odd field when interlace is specified. Replaces DADD4/LA0 as the least significant line address for interlaced sync and video applications. DADD13/LL **Last Line:** Asserted during the blanking interval just prior to the last scan line of each character row.
V_{CC}	40	44	I	**Power Supply:** +5 volts power input.
GND	20	22	I	**Ground:** Signal and power ground input.

FUNCTIONAL DESCRIPTION

As shown in the block diagram, the AVDC contains the following major blocks:

- Data bus buffer
- Interface Logic
- Operation Control
- Timing
- Display Control
- Buffer Control

Data Bus Buffer

The data bus buffer provides the interface between the external and internal data busses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the AVDC.

Interface Logic

The interface logic contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer. The functions performed by the CPU read and write operations are shown in table 1.

Operation Control

The operation control section decodes configuration and operation commands from the CPU and generates appropriate signals to other internal sections to control the overall device operation. It contains the timing and display registers which configure the display format and operating mode, the interrupt logic, and the status register which provides operational feedback to the CPU.

Timing

The timing section contains the counters and decoding logic necessary to generate the monitor timing outputs and to control the display format. These timing parameters are

selected by programming of the initialization registers.

Display Control

The display control section generates linear addressing for up to 16K bytes of display memory. Internal comparators limit the portion of the memory which is displayed to programmed values. Additional functions performed in this section include cursor positioning and address comparisons required for generation of timing signals, double height tops and bottoms, smooth scrolling, and the split screen interrupts.

Buffer Control

The buffer control section generates three signals which control the transfer of data between the CPU and the display buffer memory. Four system configurations requiring four different 'handshaking' schemes are supported. These are described below.

SYSTEM CONFIGURATIONS

Figure 1 illustrates the block diagram of a typical display terminal using the Signetics SCN2670, SCN2671, SCN2674, and SCB2675 CRT terminal devices. In this system, the CPU examines inputs from the data communications line and the keyboard and places the data to be displayed in the display buffer memory. This buffer is typically a RAM which holds the data for a single or multiple screenload (page) or for a single character row.

The AVDC supports four common system configurations of display buffer memory: the independent, transparent, shared, and row buffer modes. The first three modes utilize a single or multiple page RAM and differ primarily in the means used to transfer display data between the RAM and the CPU. The row buffer mode makes use of a single row buffer (which can be a shift register or a small RAM) that is updated in real time to contain the appropriate display data.

The user programs bits 0 and 1 of IR0 to select the mode best suited for the system environment. The CNTRL1 – 3 outputs perform different functions for each mode and are named accordingly in the description of each mode.

Independent Mode

The CPU to RAM interface configuration for this mode is illustrated in figure 2. Transfer of data between the CPU and display memory is accomplished via a bidirectional latched port and is controlled by read data buffer ($\overline{RDB}$), write data buffer ($\overline{WDB}$), and buffer chip enable ($\overline{BCE}$). This mode provides a noncontention type of operation that does not require address multiplexers. The CPU does not address the memory directly – the read or write operation is performed at the address contained in the cursor address register or the pointer address register as specified by the CPU. The AVDC enacts the data transfers during blanking intervals in order to prevent visual disturbances of the displayed data.

Table 1. AVDC ADDRESSING

A2	A1	A0	READ ($\overline{RD}$ = 0)	WRITE ($\overline{WR}$ = 0)
0	0	0	Interrupt register	Initialization registers[1]
0	0	1	Status register	Command register
0	1	0	Screen start 1 lower register	Screen start 1 lower register
0	1	1	Screen start 1 upper register	Screen start 1 upper register
1	0	0	Cursor address lower register	Cursor address lower register
1	0	1	Cursor address upper register	Cursor address upper register
1	1	0	Screen start 2 lower register	Screen start 2 lower register
1	1	1	Screen start 2 upper register	Screen start 2 upper register

[1]There are 15 initialization registers which are accessed sequentially via a single address. The AVDC maintains an internal pointer to these registers which is incremented after each write at this address until the last register (IR14) is accessed. The pointer then continues to point to IR14 for additional accesses. Upon a power-on or a master reset command, the internal pointer is reset to point to the first register (IR0) of the initialization register group. The internal pointer can also be preset to any register of the group via the 'load IR address pointer' command.

The CPU manages the data transfers by supplying commands to the AVDC. The commands used are:

1. Read/write at pointer address.
2. Read/write at cursor address (with optional increment of address).
3. Read/write from cursor address to pointer address.

The operational sequence for a write operation is:

1. CPU checks RDFLG status bit to assure that any delayed commands have been completed.
2. CPU loads data to be written to display memory into the interface latch.
3. CPU writes address into cursor or pointer registers.
4. CPU issues 'write at cursor with/without increment' or 'write at pointer' command.
5. AVDC generates control signals and outputs specified address to perform requested operation. Data is copied from the interface latch into the memory.
6. AVDC sets RDFLG status to indicate that the write is completed.

Similarly, a read operation proceeds as follows:

1. Steps 1 and 3 as above.
2. CPU issues 'read at cursor with/without increment' or 'read at pointer' command.
3. AVDC generates control signals and outputs specified address to perform requested operation. Data is copied from memory to the interface latch and AVDC sets RDFLG status to indicate that the read is completed.
4. CPU checks RDFLG status to see if operation is completed.
5. CPU reads data from interface latch.

Loading the same data into a block of display memory is accomplished via the 'write from cursor to pointer' command:

1. CPU checks RDFLG status bit to assure that any delayed commands have been completed.
2. CPU loads data to be written to display memory into the interface latch.
3. CPU writes beginning address of memory block into cursor address register and ending address of block into pointer address register.
4. CPU issues 'write from cursor to pointer' command.
5. AVDC generates control signals and outputs block addresses to copy data from the interface latch into the specified block of memory.
6. AVDC sets RDFLG status to indicate that the block write is completed.

Similar sequences can be implemented on an interrupt driven basis using the READY interrupt output to advise the CPU that a previously asserted delayed command has been completed.

Two timing sequences are possible for the 'read/write at cursor/pointer' commands. If the command is given during the active display window (defined as first scan line of the first character row to the last scan line of the last character row), the operation takes place during the next horizontal blanking interval, as illustrated in figure 3. If the command is given during the vertical blanking interval, or while the display has been commanded blanked, the operation takes place immediately. In the latter case, the execution time for the command is approximately five character clocks (see figure 4).

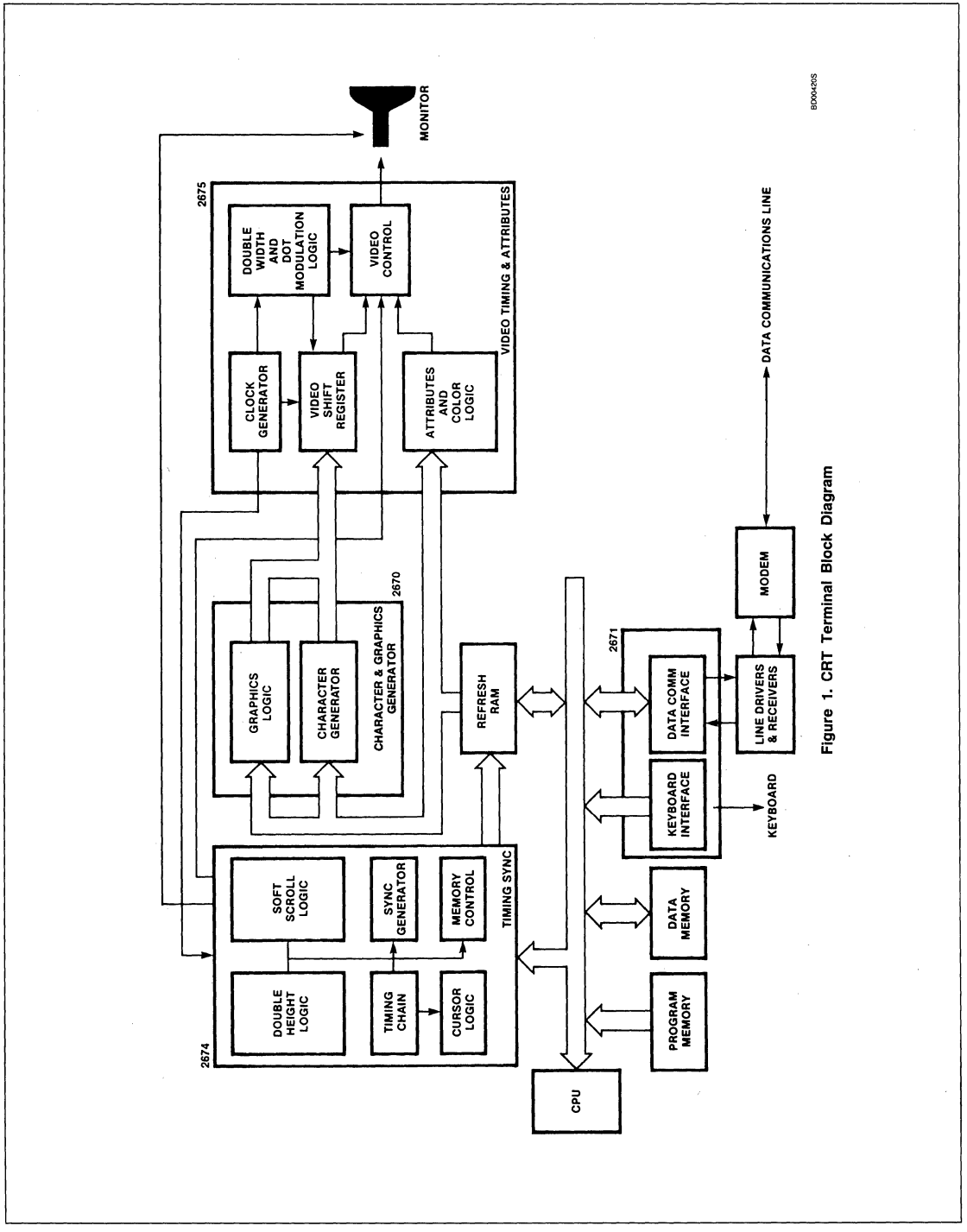


Figure 1. CRT Terminal Block Diagram

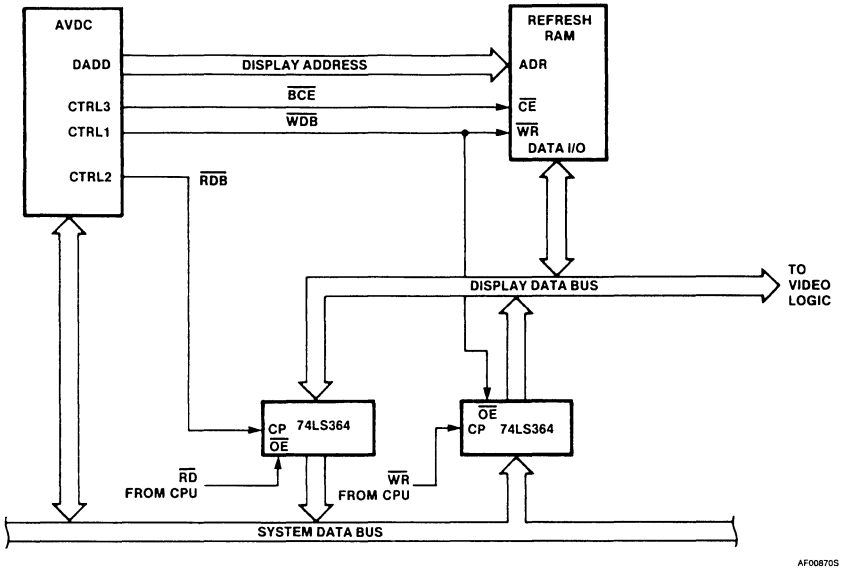
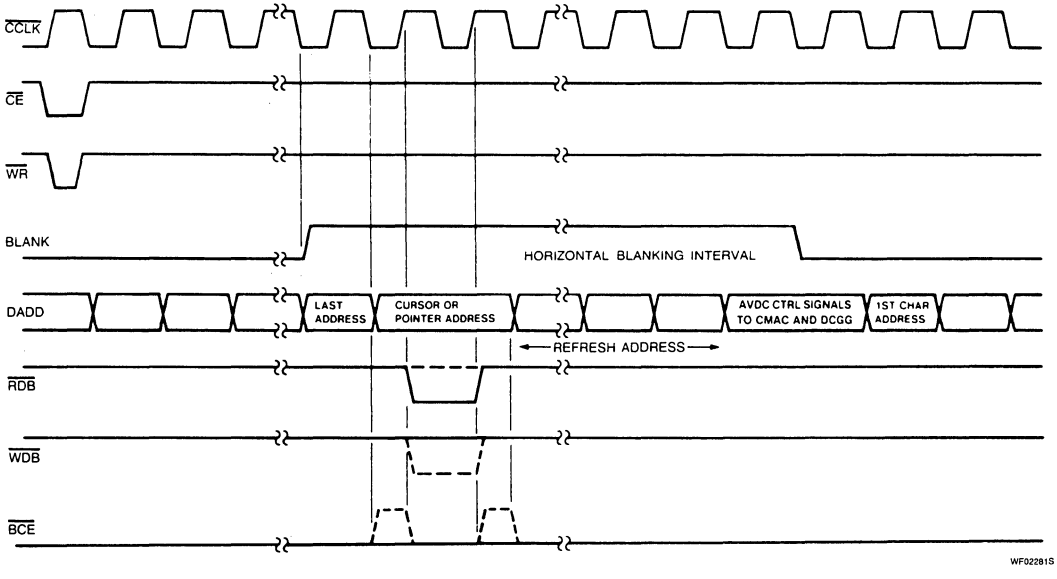


Figure 2. Independent Buffer Mode Configuration



NOTES:
1. Write waveforms shown in dotted lines.
2. If command execution occurs just prior to the first scan line of a character row and row table addressing mode is enabled, execution of the command is delayed by two character clocks from the timing illustrated

Figure 3. Read/Write at Cursor/Pointer Command Timing (Command Received During Active Display Window)

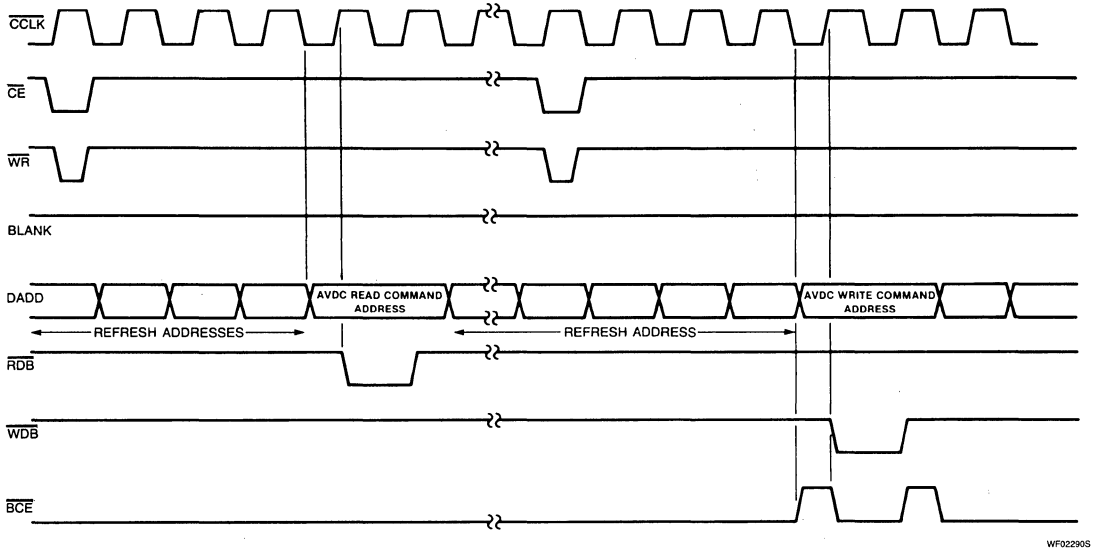


Figure 4. Read/Write at Cursor/Pointer Command Timing (Command Received While Display is Blanked)

Timing for the 'read/write from cursor to pointer' operation is shown in figure 5. The memory is filled at a rate of one location per two character times. The command will execute only during blanking intervals and may require many horizontal or vertical blanking intervals to complete. Additional delayed commands can be asserted immediately after this command has completed.

Immediate commands can be asserted at any time regardless of the state of the ready status/interrupt.

Shared and Transparent Buffer Modes

In these modes, the display buffer RAM is a part of the CPU memory domain and is addressed directly by the CPU. Both modes use the same hardware configuration with the CPU accessing the display buffer via three-state drivers (see figure 6). The processor bus request ($\overline{PBREQ}$) control signal informs the AVDC that the CPU is requesting access to the display buffer. In response to this request, the AVDC raises bus acknowledge ($\overline{BACK}$) until its bus external ($\overline{BEXT}$) output has freed the display address and data busses for CPU access. $\overline{BACK}$, which can be used as a 'hold' input to the CPU, is then lowered to indicate that the CPU can access the buffer.

In transparent mode, the AVDC delays the granting of the buffer to the CPU until a vertical or horizontal blanking interval, thereby causing minimum disturbance of the display. In shared mode, the AVDC will blank the display and grant immediate access to the CPU. Timing for these modes is illustrated in figures 7, 8, and 9.

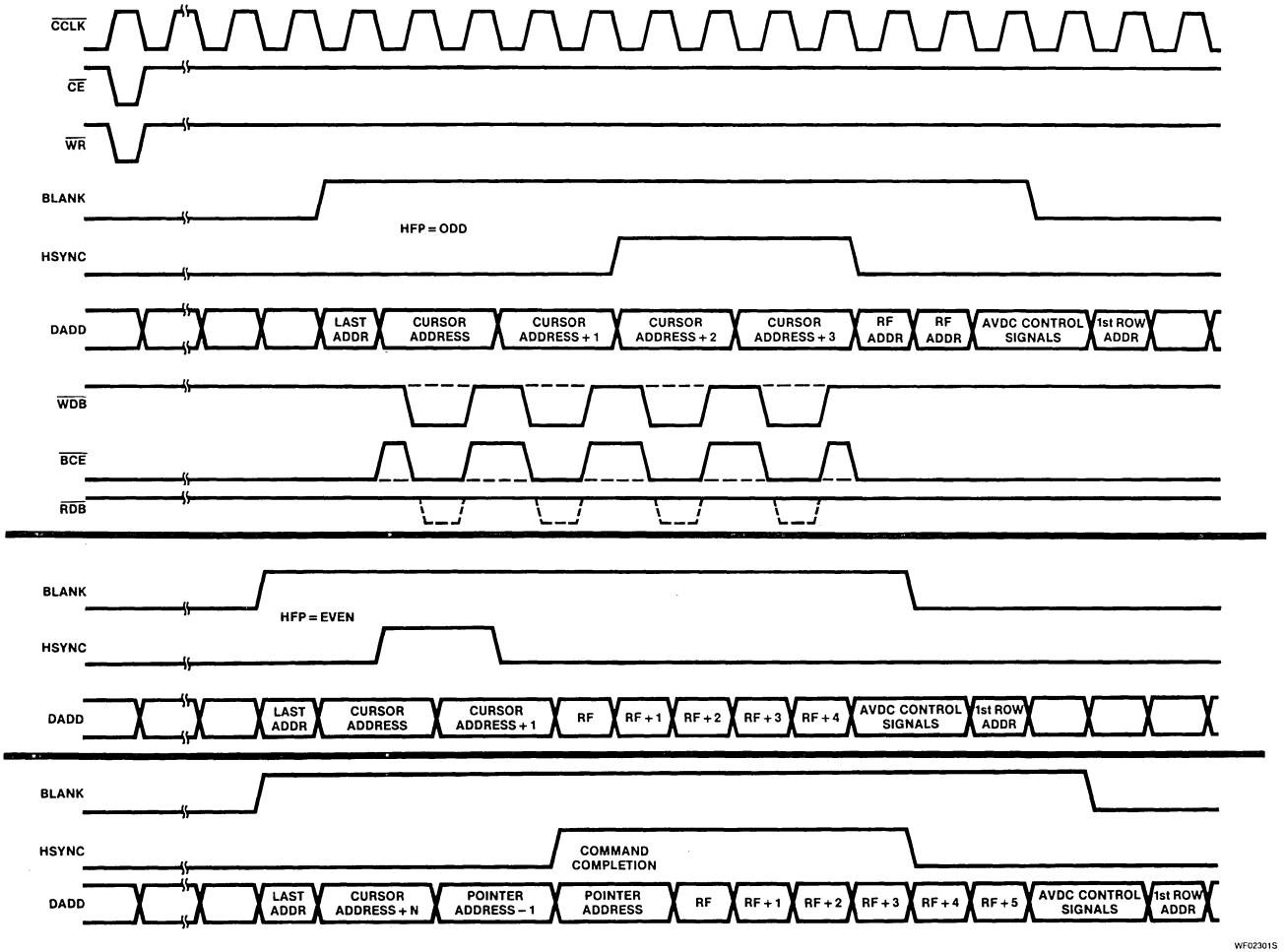
Row Buffer Mode

Figures 10 and 11 show the timing and a typical hardware implementation for the row buffer mode. During the first scan line (line 0) of each character row, the AVDC halts the CPU and DMA's the next row of character data from the system memory to the row buffer memory. The AVDC then releases the CPU and displays the row buffer data for the programmed number of scan lines. The control signal $\overline{BREQ}$ informs the CPU that character addresses and the MBC signal will start at the next falling edge of BLANK. The CPU must release the address and data busses before this time to prevent bus contention. After the row of character data is transferred to the row buffer RAM, $\overline{BREQ}$ returns high to grant memory control back to the CPU.

Row Table Addressing Mode

In this mode, each character row in the screen image memory has a unique starting address. This provides greater flexibility with respect to screen operations, such as editing, than the sequential addressing mode. The row table, figure 12, is a list of starting addresses for each character row and may reside anywhere in the AVDC's addressable memory space. Each entry in the table consists of two bytes: the first byte contains the 8 LSBs of the row starting address and the second byte contains, in its 6 least significant bits, the 6 MSBs of the row starting address. The function of the two MSBs of the second byte is selected by programming IR0[7]. They may be used either as row attribute bits to control double width and double height for that character row, or as an additional two address bits to extend the usable display memory to 64K.

The first address of the row table is designated in screen start register two (SSR2). If row table addressing is enabled via IR2[7], and the display is on, the AVDC fetches the next row's starting address from the table during the blanking interval prior to the first scan line of each character row, while simultaneously incrementing the contents of SSR2 by two so as to point to the next table entry. The fetching of the row starting address from the row table is indicated by the assertion of the CURSOR output during BLANK. The address read from the table by the AVDC is loaded into screen start register 1 (SSR1) for use internally. Since the contents of SSR2 changes as the table entries are fetched, it must be re-initialized to point to the first table entry during each vertical retrace interval.



NOTES:
1. If command execution occurs just prior to the first scan line of a character row and row table addressing mode is enabled, execution of the command is delayed by two character clocks from the timing illustrated.
2. Read waveforms shown in dotted line.

Figure 5. Read/Write from Cursor to Pointer Command Timing

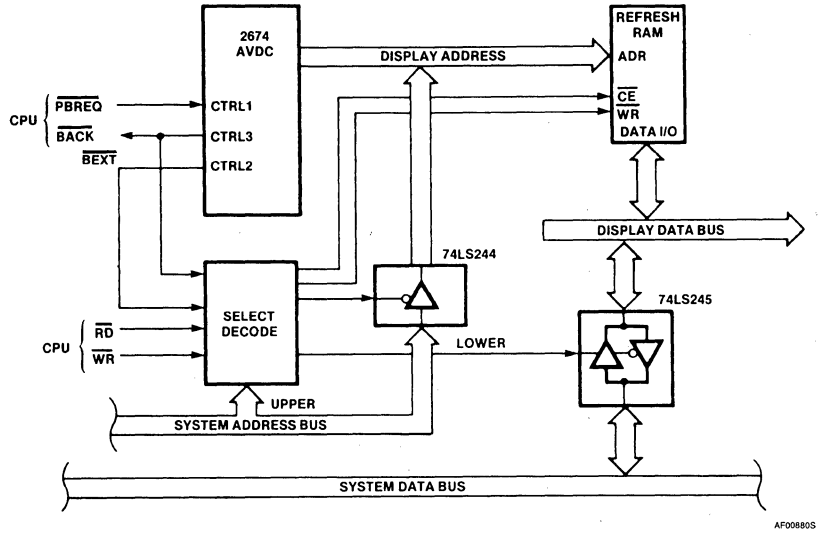
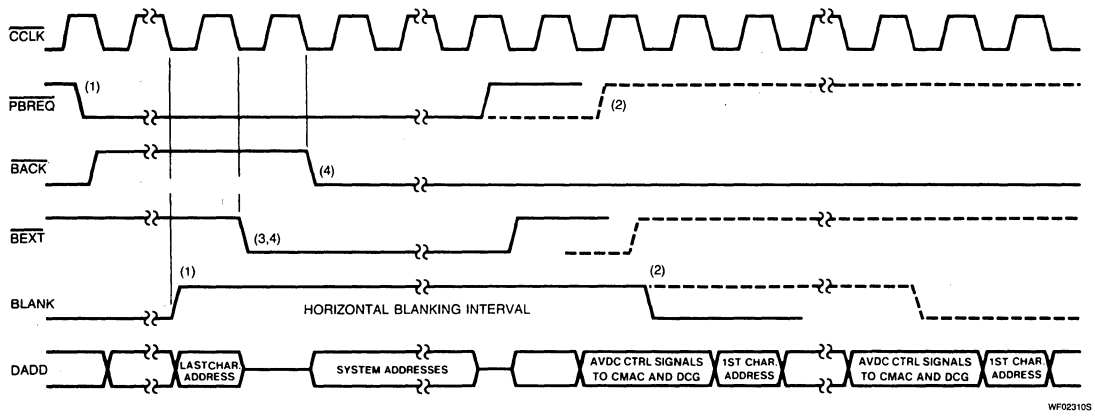
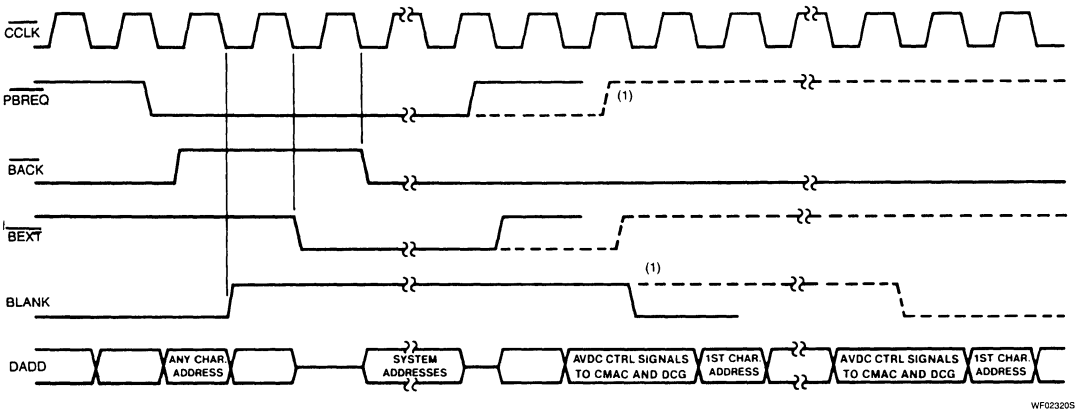


Figure 6. AVDC Shared or Transparent Buffer Modes



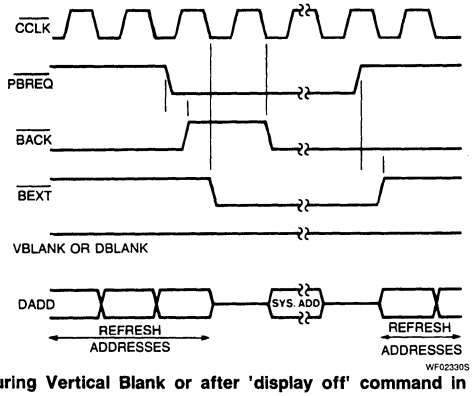
NOTES:
1. PBREQ must be asserted prior to the rising edge of BLANK in order for sequence to begin during that blanking period.
2. If PBREQ is negated after the next to last CCLK of the horizontal blanking interval, the next scan line will also be blanked.
3. Accesses during vertical blank or "display off" are granted only at the beginning of the horizontal front porch.
4. If row table addressing is enabled, CPU access is delayed by two character clocks prior to the first scan line of each character row.

Figure 7. Transparent Buffer Mode Timing

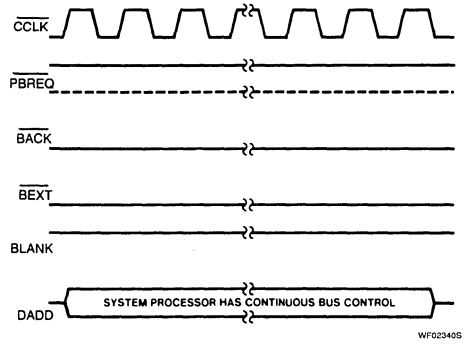


NOTE:
1. If $\overline{PBREQ}$ is negated after the next to last $\overline{CCLK}$ of the horizontal blanking interval, the next scan line will also be blanked.

Figure 8. Shared Buffer Mode Timing



a. During Vertical Blank or after 'display off' command in shared mode only. See figure 7 for transparent timing.



b. After 'display off and 3-state' command.

Figure 9. Shared and Transparent Mode Timing

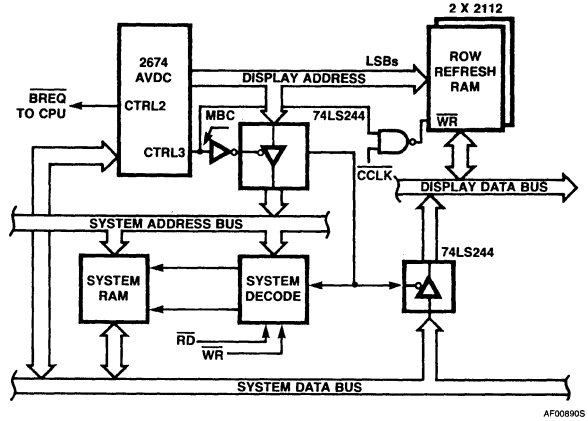
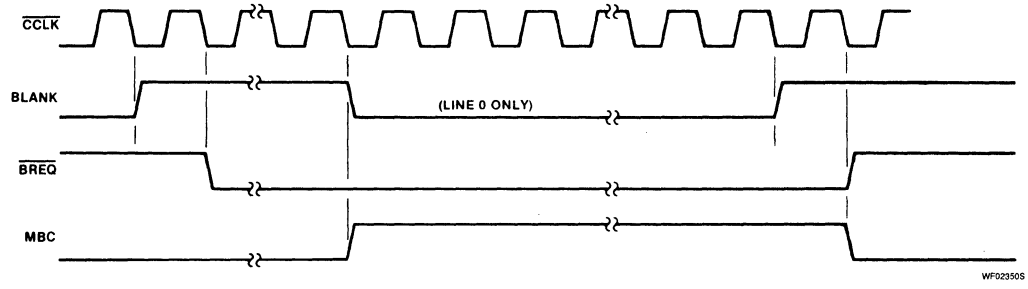


Figure 10. Row Buffer Mode Configuration



NOTE:
If row table addressing is enabled, $\overline{BREQ}$ will be asserted at the middle of the last scan line of the prior row, and MBC will be asserted at the beginning of BLANK.

Figure 11. Row Buffer Mode Timing

Row table addressing is intended primarily for use in conjunction with the row buffer mode of operation and requires no additional circuitry in that case. It may also be used with the independent and transparent buffer modes, but circuitry must be added to route the data from the display memory to the data bus inputs of the AVDC. Additionally, when not operating in row buffer mode, care must be taken to assure that the CPU does not attempt to access the AVDC while it is reading the row table. One way of preventing this is to latch the last line output which is multiplexed on DADD13 and to test this latch prior to reading or writing the AVDC. The AVDC should only be accessed if the latch is low, indicating that the last line of the row is not active.

Figure 13 illustrates a typical hardware implementation for use in conjunction with independent and transparent modes, and figure 14 shows the timing for row table operation.

OPERATION

After power is applied, the AVDC will be in an inactive state. Two consecutive 'master reset' commands are necessary to release this circuitry and ready the AVDC for operation. Two register groups exist within the AVDC: the initialization registers and the display control registers. The initialization registers select the system configuration, monitor timing, cursor shape, display memory domain, pointer address, scrolling region, double height and width condition, and screen format. These are loaded first and normally require no modification except for certain special visual effects. The display control registers specify the memory address of the base character (upper left corner of screen), the cursor position, and the split screen addresses associated with the scrolling area or an alternate memory. These may require modification during operation.

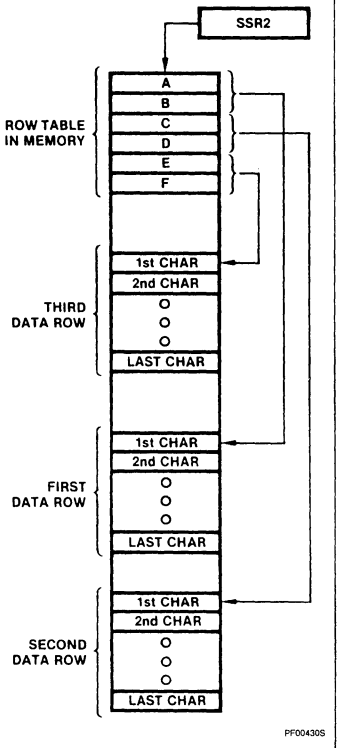


Figure 12. Row Table Address Format

After initial loading of the two register groups, the AVDC is ready to control the monitor screen. Prior to executing the AVDC commands which turn on the display and cursor, the user should load the display memory with the first data to be displayed. During operation, the AVDC will sequentially address the display memory within the limits programmed into its registers. The memory outputs character codes to the system character and graphics generation logic, where they are converted to the serial video stream necessary to display the data on the CRT. The user effects changes to the display by modifying the contents of the display memory, the AVDC display control and command registers, and the initialization registers, if required. Interrupts and status conditions generated by the AVDC supply the 'handshaking' information necessary for the CPU to effect real time display changes in the proper time frame if required.

Initialization Registers

There are 15 initialization registers (IR0 – IR14) which are accessed sequentially via a single address. The AVDC maintains an internal pointer to these registers which is incremented after each write at this address until the last register (IR14) is accessed. The pointer then continues to point to IR14 for further accesses. Upon a power-on or a master reset command, the internal pointer is reset to point to the first register (IR0) of the initialization register group. The internal pointer can also be preset to any register of the group via the 'load IR address pointer' command. These registers are write only and are used to specify parameters such as the system configuration, display format, cursor shape, and monitor timing. Register formats are shown in table 2.

IR0[7] – Double Height/Width Enable
When this bit is set, the value in IR14[7:6] is used to control the double height and width conditions of each character row. Assertion of this bit also allows IR14[7:6] to be programmed in two ways:

1. By the CPU writing to IR14 directly.
2. When the contents of screen start register 1 (SSR1) upper are changed, either by the CPU writing to this register or by the automatic loading of SSR1 when operating in row table mode, the two MSBs of SSR1 upper are copied into IR14[7:6]. Thus, the MSBs of each row table entry can be used to control double height and double width attributes on a row by row basis.

IR14[5:4] are not active when this bit is set. When this bit is reset, the double height and width attributes operate as described in IR[14].

IR0[6:3] – Scan Lines Per Character Row
Both interlaced and non-interlaced scanning are supported by the AVDC. For interlaced mode, two different formats can be implemented, depending on the interconnection between the AVDC and the character generator (see IR1[7]). This field defines the number of scan lines used to compose a character row for each technique. As scanning occurs, the scan line count is output on the LA0-LA3 and ODD pins.

IR0[2] – VSYNC/CSYNC Enable
This bit selects either vertical sync pulses or composite sync pulses on the VSYNC/CSYNC output (pin 18). The composite sync waveform conforms to EIA RS170 standards, with the vertical interval composed of six equalizing pulses, six vertical sync pulses, and six more equalizing pulses.

IR0[1:0] – Buffer Mode Select
Four buffer memory modes may be selectively enabled to accommodate the desired system configuration. See System Configurations.

IR1[7] – Interlace Enable
Specifies interlaced or noninterlaced timing operation. Two modes of interlaced operation are available, depending on whether LA0 – LA3 or ODD, LA0 – LA2 are used as the line address for the character generator. The resulting displays are shown in figure 15.

For 'interlaced sync' operation, the same information is displayed in both odd and even fields, resulting in enhanced readability. The AVDC outputs successive line numbers in ascending order on the LA0-LA3 lines, one per scan line for each field.

The 'interlaced sync and video' format doubles the character density on the screen. The AVDC outputs successive line numbers in ascending order on the ODD and LA0 – LA2 lines, one per scan line for each field. The number of scan lines per character row is always even. Assume that the first character row is row 0 (even). When scanning through the odd field, the scan line numbers being displayed are odd for both the even and odd character rows. When scanning through the even field, the scan line numbers being displayed are even for both even and odd character rows (see figure 15).

IR1[6:0] – Equalizing Constant
This field indirectly defines the horizontal front porch and is used internally to generate the equalizing pulses for the RS170 compatible CSYNC. The value for this field is the total number of character clocks ($\overline{CCLK}$s) during a horizontal line period divided by two, minus two times the number of character clocks in the horizontal sync pulse:

$$EC = \frac{H_{ACT} + H_{FP} + H_{SYNC} + H_{BP}}{2} - 2(H_{SYNC})$$

The definition of the individual parameters is illustrated in figure 16. The minimum value of H_{FP} is three character clocks, four $\overline{CCLK}$'s for row table addressing. Note that when using the 2675 CMAC, it will delay the blank pulse three $\overline{CCLK}$s relative to the HSYNC pulse. Because of this delay, the actual HFP and HBP values will be different from the values programmed into the AVDC. The actual HFP will be decreased by 3 character clocks. The actual HBP will be increased by 3 character clocks.

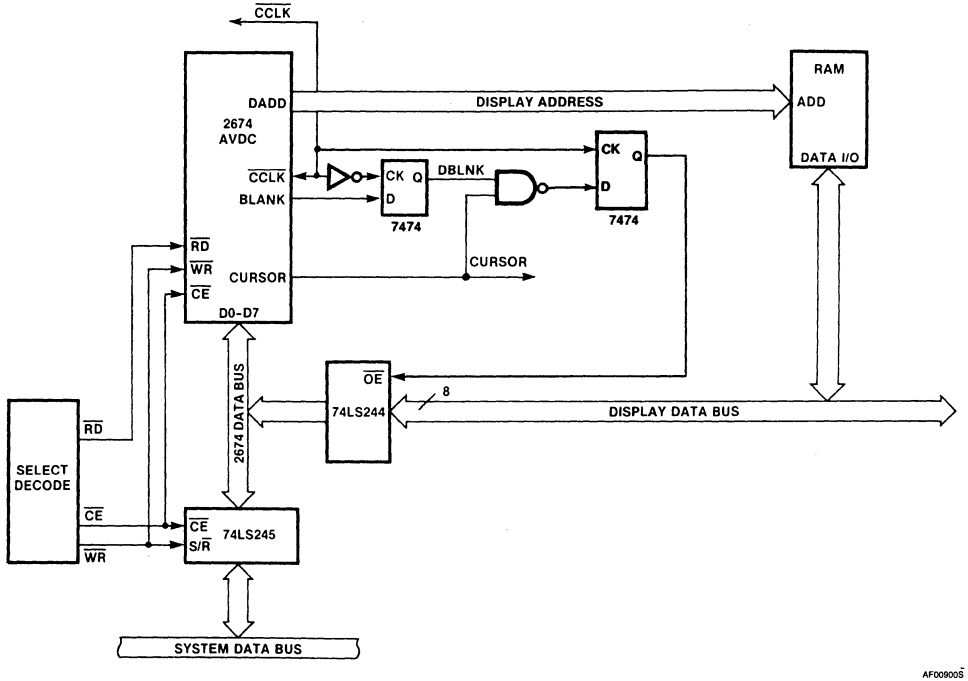


Figure 13. Row Table Mode Configuration (non-Row Buffer Modes)

IR2[7] – Row Table Mode Enable
Assertion/negation of this bit causes the AVDC to begin/terminate operating in row table mode starting at the next character row. See Row Table Addressing Mode section. By using the split interrupt capability of the AVDC, this mode can be enabled and disabled on a particular character row. This allows a combination of row table and sequential addressing to be utilized to provide maximum flexibility in generating the display.

IR2[6:3] – Horizontal Sync Pulse Width
This field specifies the width of the HSYNC pulse in $\overline{\text{CCLK}}$ periods.

IR2[2:0] – Horizontal Back Porch
This field defines the number of $\overline{\text{CCLK}}$s between the trailing edge of HSYNC and the trailing edge of BLANK.

IR3[7:5] – Vertical Front Porch
Specifies the number of scan line periods between the rising edges of BLANK and VSYNC during the vertical retrace interval. The vertical front porch will be extended in increments of scan lines if the ACLL input is low at the end of the programmed value.

IR3[4:0] – Vertical Back Porch
This field determines the number of scan line periods between the falling edges of the VSYNC and BLANK outputs.

IR4[7] – Character Blink Rate
Specifies the frequency for the character blink attribute timing. The blink rate can be specified as 1/64 or 1/128 of the vertical field rate. The timing signal has a duty cycle of 50% and is multiplexed onto the DADD11/BLINK output at the falling edge of each BLANK.

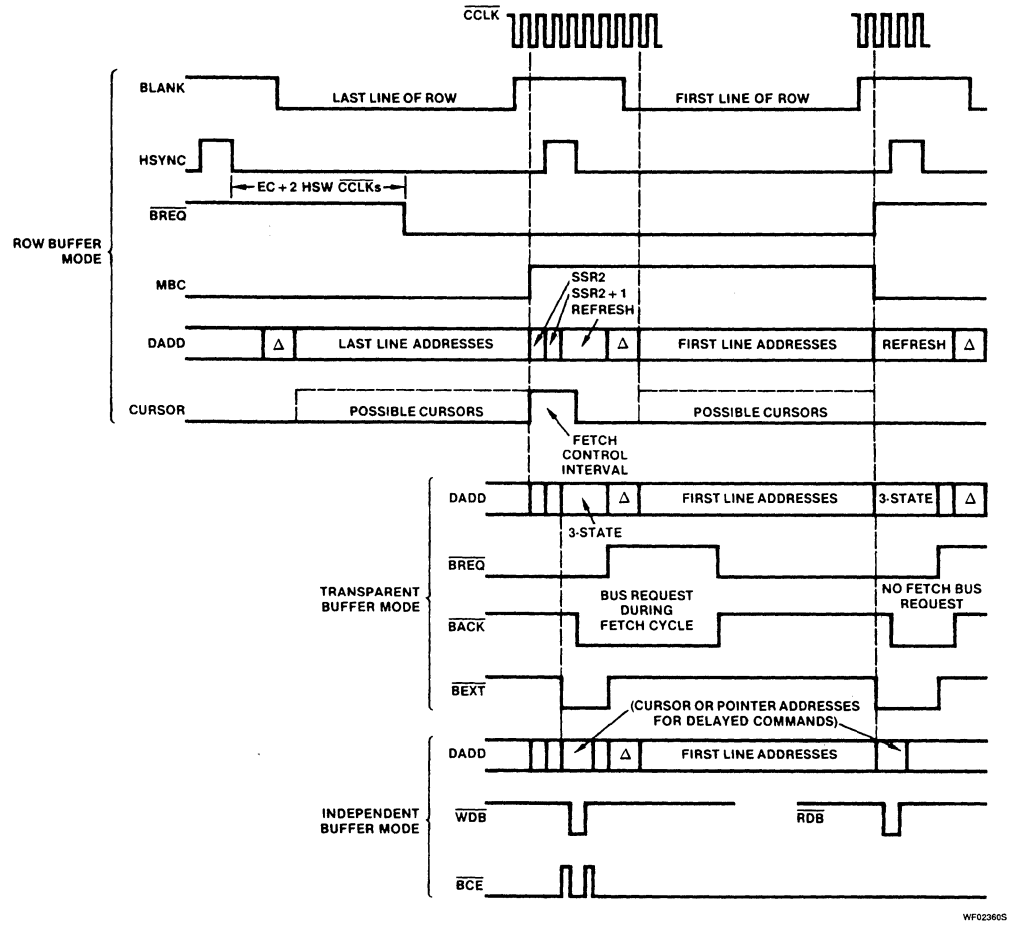
IR4[6:0] – Character Rows Per Screen
This field defines the number of character rows to be displayed. This value multiplied by the scan lines per character row, plus the vertical front porch, the vertical back porch values, and the vertical sync pulse width is the vertical scan period in scan lines.

IR5[7:0] – Active Characters Per Row
This field determines the number of characters to be displayed on each row of the CRT screen. The sum of this value, the horizontal front porch, the horizontal sync width, and the horizontal back porch is the horizontal scan period in $\overline{\text{CCLK}}$s.

IR6[7:4], IR6[3:0] – First and Last Scan Line of Cursor
These two fields specify the height and position of the cursor on the character block. The 'first' line is the topmost line when scanning from the top to the bottom of the screen. The value of the 'first line of cursor' must be less than the 'last line of cursor' value.

IR7[7:6] – Vertical Sync Pulse Width
This field specifies the width of the VSYNC pulse in scan line periods.

IR7[5] – Cursor Blink Enable
This bit controls whether or not the cursor output pin will be blinked at the selected rate (IR7[4]). The blink duty cycle for the cursor is 50%.



NOTES:
Δ = Multiplexed control signals (LG,DW, etc.)
EC = Equalizing constant
HSW = Horizontal sync width

Figure 14. Row Table Mode Timing

Table 2. INITIALIZATION REGISTER BIT FORMATS

IR0

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DOUBLE HT/WD	SCAN LINES PER CHARACTER ROW				SYNC SELECT	BUFFER MODE SELECT	
	NON-INTERLACED		INTERLACED				
0 = OFF 1 = ON	0000 = 1 LINE 0001 = 2 LINES 0010 = 3 LINES • • 1110 = 15 LINES 1111 = 16 LINES		0000 = 2 LINES 0001 = 4 LINES 0010 = 6 LINES • • 1110 = 30 LINES 1111 = UNDEFINED		0 = VSYNC 1 = CSYNC	00 = INDEPENDENT 01 = TRANSPARENT 10 = SHARED 11 = ROW	

IR1

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
INTERLACE ENABLE	EQUALIZING CONSTANT						
0 = NON-INT 1 = INTER	00000000 = 1 CCLK 00000001 = 2 CCLK • • 11111110 = 127 CCLK 11111111 = 128 CCLK		CALCULATED FROM: $EC = 0.5(H_{ACT} + H_{FP} + H_{SYNC} + H_{BP}) - 2(H_{SYNC})$				

IR2

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ROW TABLE	HORIZONTAL SYNC WIDTH				HORIZONTAL BACK PORCH		
0 = OFF 1 = ON	0000 = 2 CCLK 0001 = 4 CCLK • • 1110 = 30 CCLK 1111 = 32 CCLK				000 = NOT ALLOWED 001 = 3 CCLK • • 110 = 23 CCLK 111 = 27 CCLK		

IR3

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
VERTICAL FRONT PORCH			VERTICAL BACK PORCH				
000 = 4 SCAN LINES 001 = 8 SCAN LINES • • 110 = 28 SCAN LINES 111 = 32 SCAN LINES			00000 = 4 SCAN LINES 00001 = 6 SCAN LINES • • 11110 = 64 SCAN LINES 11111 = 66 SCAN LINES				

IR4

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CHARACTER BLINK RATE	ACTIVE CHARACTER ROWS PER SCREEN						
0 = 1/64 VSYNC 1 = 1/128 VSYNC	00000000 = 1 ROW 00000001 = 2 ROWS • • 11111110 = 127 ROWS 11111111 = 128 ROWS						

Table 2. INITIALIZATION REGISTER BIT FORMATS (Continued)

IR5

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ACTIVE CHARACTERS PER ROW							
00000010 = 3 CHARACTERS 00000011 = 4 CHARACTERS • • 11111110 = 255 CHARACTERS 11111111 = 256 CHARACTERS							

IR6

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
FIRST LINE OF CURSOR				LAST LINE OF CURSOR			
0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15				0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15			

IR7

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
VSYNC WIDTH		CURSOR BLINK	CURSOR RATE	UNDERLINE POSITION			
00 = 3 SCAN LN 01 = 1 SCAN LN 10 = 5 SCAN LN 11 = 7 SCAN LN		0 = OFF 1 = ON	0 = 1/32 1 = 1/64	0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15			

IR8

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DISPLAY BUFFER FIRST ADDRESS LSB'S							
H'000 = 0 H'001 = 1 • • H'FFE = 4,094 H'FFF = 4,095				NOTE: MSB'S ARE IN IR9[3:0]			

IR9

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DISPLAY BUFFER LAST ADDRESS				DISPLAY BUFFER FIRST ADDRESS MSB'S			
0000 = 1,023 0001 = 2,047 • • 1110 = 15,359 1111 = 16,383				SEE IR8			

Table 2. INITIALIZATION REGISTER BIT FORMATS (Continued)

IR10

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DISPLAY POINTER ADDRESS LOWER							
SEE IR11							

IR11

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LZ DOWN	LZ UP	DISPLAY POINTER ADDRESS UPPER					
0 = OFF 1 = ON	0 = OFF 1 = ON	H'0000' = 0 H'0001' = 1 • • H'3FFF' = 16,383					

IR12

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
SCROLL START	SPLIT REGISTER 1						
0 = OFF 1 = ON	00000000 = ROW 1 00000001 = ROW 2 • • 11111111 = ROW 128						

IR13

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
SCROLL END	SPLIT REGISTER 2						
0 = OFF 1 = ON	00000000 = ROW 1 00000001 = ROW 2 • • 11111111 = ROW 128						

IR14

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DOUBLE 1		DOUBLE 2		LINES TO SCROLL			
00 = NORMAL 01 = DOUBLE WIDTH 10 = DB WD & TOPS 11 = DB WD & BOTS		00 = NORMAL 01 = DOUBLE WIDTH 10 = DB WD & TOPS 11 = DB WD & BOTS		0000 = 1 0001 = 2 • • 1110 = 15 1111 = 16		SCAN LINE 0 SCAN LINE 1 • • SCAN LINE 14 SCAN LINE 15	

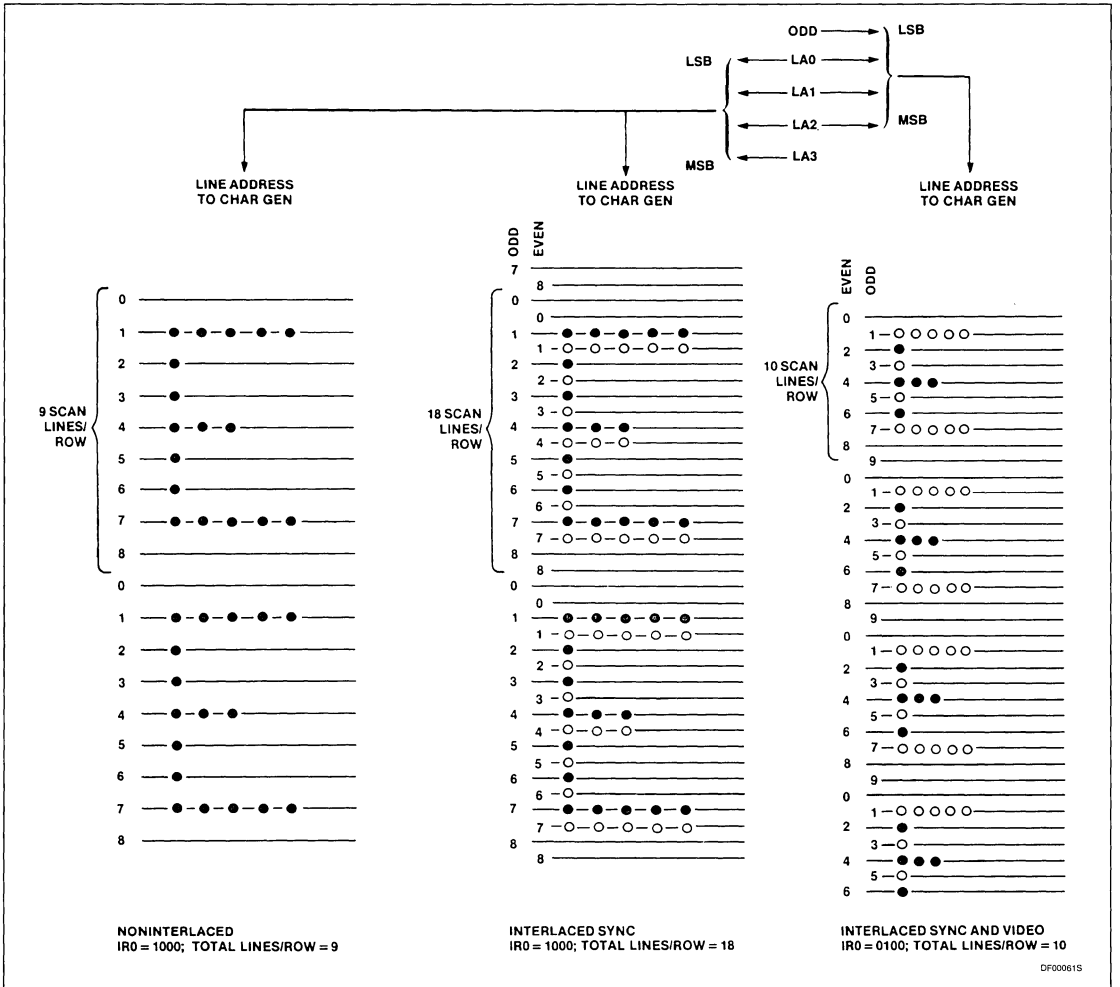


Figure 15. Interlaced Display Modes

IR7(4) – Cursor Blink Rate
The cursor blink rate can be specified at 1/32 or 1/64 of the vertical scan frequency. Blink is effective only if blink is enabled by IR7[5].

IR7[3:0] – Underline Position
This field defines which scan line of the character row will be used for the underline attribute by the 2675 CMAC. The timing signal is multiplexed onto the DADD10/UL output during the falling edge of BLANK.

IR9[3:0], IR8[7:0] – Display Buffer First Address

IR9[7:4] – Display Buffer Last Address
These two fields define the area within the buffer memory where the display data will reside. When the data at the 'display buffer last address' is displayed, the AVDC will wraparound and obtain the data to be displayed at the next screen position from the 'display buffer first address'. If 'last address' is the end of a character row and a new screen start address has been loaded into the screen start register, or if 'last address' is the last character position of the screen, the next data is obtained from the address contained in the screen start register.

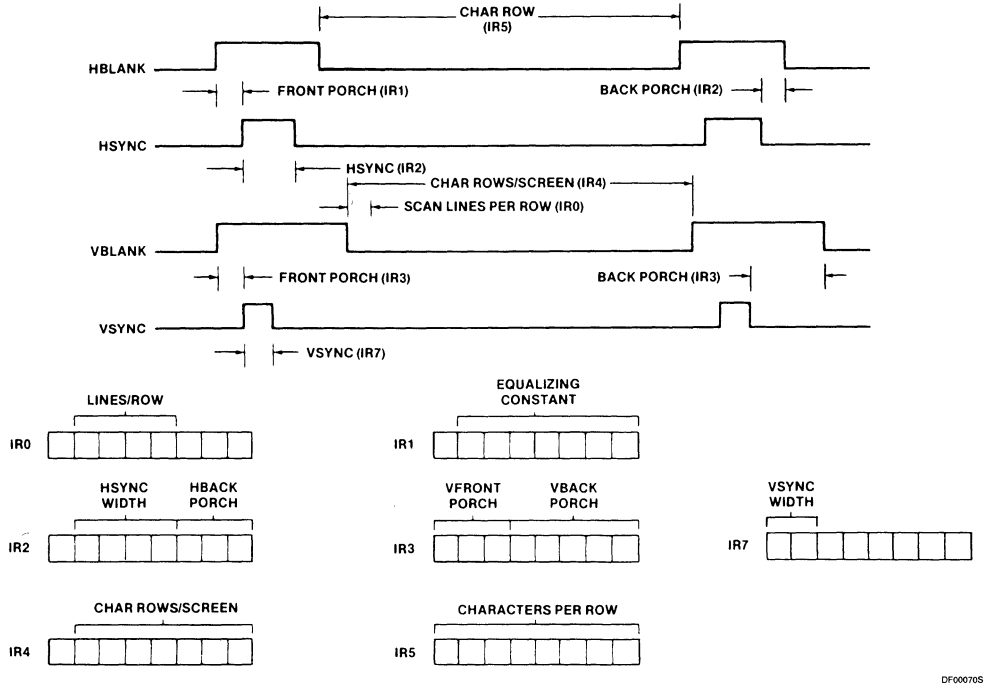


Figure 16. Horizontal and Vertical Timing

Note that there is no restriction in displaying data from other areas of the addressable memory. Normally, the area between these two bounds is used for data which can be overwritten (e.g., as a result of scrolling), while data that is not to be overwritten would be contained outside these bounds and accessed by means of the automatic split screen or split screen interrupt feature of the AVDC.

IR10[7:0] – Display Pointer Address Lower

IR11[5:0] – Display Pointer Address Upper

These two fields define a buffer memory address for AVDC controlled accesses in response to 'read/write at pointer' commands. They also define the last buffer memory address to be written for the 'write from cursor to pointer' command.

In the independent mode, the RDFLG bit of the status register should be checked for the ready state (bit 5 equal to a logic one) before writing to the display pointer address registers. Checking the status register will prevent the pointer address from being changed while a delayed command (e.g. write from cursor to pointer) is still being executed.

IR11[7] – Scan Line Zero During Scroll Down

During a scroll down operation, the new character row will appear at the top of the scrolling region. If this bit is set (logic one), the scan line count pins (LA0 through LA3) will be forced to zero for every scan line of the partial row. If the character generator provides blanks for scan line zero, the new row being scrolled into the screen will be blanked. This feature can be used to blank the new row to give the CPU time to load the new data in the display buffer. When this bit is set to a logic zero, the new data will be displayed.

IR11[6] – Scan Line Zero During Scroll Up

During a scroll up operation, the new character row will appear at the bottom of the scrolling region. If this bit is set (logic one), the scan line count pins (LA0 through LA3) will be forced to zero for every scan line of the partial row. If the character generator provides blanks for scan line zero, the new row being scrolled into the screen will be blanked. This feature can be used to blank the new row to give the CPU time to load the new data in the display buffer. When this bit is set to a logic zero, the new data will be displayed.

IR12[7] – Scroll Start

This bit is asserted when soft scroll is to take place. The scrolling area begins at the row specified in split register 1 (IR12[6:0]). If set, the first row to scroll scan line count will be reduced by the value in the lines to scroll register (IR14[3:0]). The scan line count of this row will start at the programmed offset value. When this bit is asserted, scroll end IR13[7] must be set before split 2.

IR12[6:0] – Split Register 1

Split register 1 can be used to provide special screen effects such as soft (scan line by scan line) scrolling, double height/width rows, or to change the normal addressing sequence of the display memory. The contents of this field are compared, in real time, to the current row number. Upon a match, the AVDC sets the split screen 1 status bit, and issues an interrupt request if so programmed. The status change/interrupt request is made at the beginning of scan line zero of the split screen character row. If enabled by the SPL1 bit of screen start register 2, an automatic split screen to the address specified in screen start register 2 will be made for the designated character row. During a scroll operation,

this field defines the first character row of the scrolling area.

IR13[7] – Scroll End
This field specifies that the row programmed in split register 2 (IR13[6:0]) is to be the last scrolling row of the scrolling area. Note that this bit must be asserted for a valid row only when the scroll start bit IR12[7] is also asserted.

IR13[6:0] – Split Register 2
This field is similar to the split register 1 field except for the following:

1. Split screen 2 status bit is set.
2. During a scroll operation, this field defines the last character row of the scrolling area. This row will be followed by a partial row. The LTSR (IR14) value replaces the normal scan lines/row value for the partial row, thus keeping the total scan lines/screen the same.
3. If enabled by the SPL2 bit of screen start register 2, an automatic split to the address contained in screen start register 2 will occur in one of two ways: a) If not scrolling an automatic split will occur for the next character row. b) If scrolling, the automatic split will occur after the partial row being scrolled onto or off the screen.
4. The specified double width and height conditions (IR14) are also asserted in two possible ways: a) Automatic split will assert the programmed condition for the current row. b) During soft scroll operation the programmed conditions are asserted for the partial row scrolling onto or off the screen.

IR14[7:6] – Double 1
This field specifies the conditions (double width/height or normal) of the row designated in split register 1 (IR12[6:0]). When double height tops or bottoms has been specified, the AVDC will automatically toggle between tops and bottoms until another split 1 or 2 occurs which changes the double height/width condition. If a double height tops row is specified, the scan line count will start at zero and increment the scan line count every other scan line. If a double height bottom row is specified, the AVDC will start at one half the normal scan line total. If double width is specified, the AVDC will assert the DADD9/DW output at the falling edge of blank. This condition will also remain active until the next split 1 or 2. When IR0[7] = 1, the values written into bits 7 and 6 of screen start 1 upper will also be written into IR14[7:6] and the automatic toggling between tops and bottoms is disabled.

The AVDC still addresses the RAMs on a single width character basis. The clock rate of the AVDC does not change. The display RAMs must have data as if two single wide characters are to be displayed. The first data bits addressed by the AVDC will specify the double wide character. The next data bits addressed are not displayed. The 2675 CMAC will ignore the second clock cycle data when the ADOUBLE pin is high.

IR14[5:4] – Double 2
This field specifies the conditions (double width/height or normal) of the row designated in split register 2 (IR13[6:0]). Not used when IR0[7] = 1.

IR14[3:0] – Lines To Scroll
This field defines the scan line increment to be used during a soft scroll operation. These 4 bits control the scroll rate. When smooth scrolling up by scan line increments of one, the initial value is 0000 (scan line 0) and is increased every vertical frame according to the number of lines per character row. When smooth scrolling down by scan line increments of one, the initial value is 1110 hex (scan line 14, assuming 15 scan lines per character row) and is decreased by one every vertical frame. This value will only be used when scroll start (IR12[7]) and scroll end (IR13[7]) are enabled.

Timing Considerations
Normally, the contents of the initialization registers are not changed during normal operation. However, this may be necessary to implement special display features such as multiple cursors and horizontal scrolling. Table 3 describes timing details for these registers which should be considered when implementing these features.

Display Control Registers

There are seven registers in this group, each with an individual address. Their formats are illustrated in table 4. The command register is used to invoke one of 19 possible AVDC commands as described in the COMMANDS section of this data sheet. The remaining registers in the group store address values which specify the cursor location, the location of the first character to be displayed on the screen, and any split screen address locations. The user initializes these registers after powering on the system and changes their values to control the data which is displayed.

Screen Start Registers 1 and 2
The screen start 1 registers contain the address of the first character of the first row (upper left corner of the active display). At the beginning of the first scan line of the first row, this address is transferred to the row start register (RSR) and into the memory address counter (MAC). The counter is then advanced sequentially at the character clock rate for the number of times programmed into the active characters per row register (IR5), thus reaching the address of the last character of the row plus one. At the beginning of each subsequent scan line of the first row, the MAC is reloaded from the RSR and the above sequence is repeated. At the end of the last scan line of the first row, the contents of the MAC are loaded into the RSR to serve as the starting memory address for the second character row. This process is repeated for the programmed number of rows per screen. Thus, the data in the display memory is displayed sequentially starting from the address contained in the screen start register. After the ensuing vertical retrace interval, the contents of the screen start registers are reloaded into the RSR and MAC, and the process is repeated.

During vertical blanking, the address counter operation is modified by stopping the automatic load of the contents of the RSR into the counter, thereby allowing the address outputs to free-run. This allows dynamic memory refresh to occur during the vertical retrace interval. The refresh addressing starts at the last address displayed on the screen and increments by one for each character clock during the retrace interval. If the display buffer last address is encountered, wraparound will occur. Refreshing will continue from the display buffer first address. In the independent mode, the refresh addressing will occur if no delayed commands are being executed. In the transparent and shared modes, refresh will occur during the blanking interval unless the CPU has control of the display address bus. In the row buffer mode, refresh will occur during all blanking intervals except for the first character clock time in the BLANK after the first scan line (scan line 0) of a character row.

The sequential addressing operation described above will be modified upon the occurrence of the following:

1. After reaching the 'display buffer last address.'
2. Rewriting the contents of the screen start 1 registers.
3. Setting the split register 1 or split register 2 bits of screen start register 2 upper.
4. Enabling the row table addressing mode.

First, if during the incrementing of the memory address counter the 'display buffer last address' (IR9[7:4]) is reached, the MAC will be loaded from the 'display buffer first address' register (IR9[3:0] and IR8[7:0]) at the next character clock. Sequential operation will then resume starting from this address. This wraparound operation allows portions of the display buffer to be used for purposes other than storage of displayable data and is completely automatic without any CPU intervention (see figure 17a).

Second, if the contents of screen start register 1 (upper, lower, or both) are changed during any character row (e.g., row 'n'), the starting address of the next character row (row 'n + 1') will be the new value of the screen start register and addressing will con-

tinue sequentially from there. This allows features such as split screen operation, partial scroll, or status line display to be implemented. The split screen interrupt feature of the AVDC is useful in controlling the CPU initiated operations. Note that in order to obtain the correct screen display, screen start register 1 must be reloaded with the original (origin of display) value prior to the end of the vertical retrace. See figure 17b.

The screen start two registers contain a 14-bit display address. The SSR2 address is implemented on the occurrence of item 3 above. If bit 6 of SSR2 upper is set, the SSR2 contents will be automatically loaded into the RSR at the beginning of the first scan line of the row designated by split register 1 (IR12[6:0]). If bit 7 of SSR2 upper is set, the SSR2 contents will be automatically loaded into the RSR at the beginning of the first scan line of the row specified by split register 2 (IR13[6:0]). SPL1 and SPL2 are write only bits and will read as zero when reading screen start register 2. If these bits are not used, they should be set to zeros after power up.

Lastly, when row table addressing mode is enabled, the first address of the row table is designated in SSR2. The AVDC fetches the next row's starting address from the table during the blanking interval prior to the first scan line of each character row and loads it into SSR1 for use as the starting address of the next row. Since the contents of SSR2 change as the table entries are fetched, it must be re-initialized to point to the first table entry during each vertical retrace interval.

The values in the two MSBs of SSR1 upper are multiplexed onto the DADD1/DADD14 and DADD2/DADD15 outputs during the falling edge of BLANK. If IR0[7] = 0, these two bits act as memory page select bits which may be used to extend the display memory addressing range of the AVDC up to 64K. In that case, these two bits act as a two-bit counter which is incremented each time that 'wraparound' occurs (see above). Note that the counter is incremented at the falling edge of BLANK and that for proper display operation the wraparound address should be programmed to occur at the last character position of a row. Also, the first address accessed in the new page will be the address contained in the display buffer first address register (IR9[3:0] and IR8[7:0]). DADD14 and DADD15 should only be used in the bit mapped graphics mode.

Table 3. TIMING CONSIDERATIONS

PARAMETER	TIMING CONSIDERATIONS
First line of cursor Last line of cursor Underline line	These parameters must be established at a minimum of two character times prior to their occurrence
Double height character rows Double width character rows Rows to scroll	Set/reset prior to the row specified in split 1 or 2 registers
Cursor blink Cursor blink rate Character blink rate	New values become effective within one field after values are changed
Split register 1 Split register 2	Change anytime prior to line zero of desired row
Character rows per screen	Change only during vertical blanking period
Vertical front porch	Change prior to first line of VFP
Vertical back porch	Change prior to fourth line after VSYNC
Screen start register 1 Row table mode enable	Change prior to the horizontal blanking interval of the last line of character row before row where new value is to be used

Cursor Address Registers

The contents of these registers define the buffer memory address of the cursor. The cursor output will be asserted when the memory address counter matches the value of the cursor address registers for the scan lines specified in IR6. The cursor address registers can be read or written by the CPU or incremented via the 'increment cursor address' command. In independent buffer mode, these registers define a buffer memory address for AVDC controlled access in response to 'read/write at cursor with/without increment' commands, or the first address to be used in executing the 'write from cursor to pointer' command.

In the independent mode, the RDFLG bit of the status register should be checked for the ready state (bit 5 equal to a logic one) before writing to the cursor address registers. Checking the status register will prevent the cursor address from being changed while a cursor delayed command (e.g., write from cursor to pointer) is still being executed.

Interrupt/Status Registers

The interrupt and status registers provide information to the CPU to allow it to interact with the AVDC to effect desired changes that implement various display operations. The interrupt register provides information on five possible interrupting conditions, as shown in Table 5. These conditions can be selectively enabled or disabled (masked) from causing interrupts by certain AVDC commands. An interrupt condition which is enabled (mask bit equal to one) will cause the $\overline{INTR}$ output to be asserted and will cause the corresponding bit in the interrupt register to be set-upon the occurrence of the interrupting condition. An interrupt condition which is disabled (mask bit equal to zero) has no effect on either the $\overline{INTR}$ output or the interrupt register.

The status register provides six bits of status information: the five possible interrupting conditions plus the RDFLG bit. For this register, however, the contents are not affected by the state of the mask bits.

2

Table 4. DISPLAY CONTROL REGISTER BIT FORMATS

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
COMMAND CODE							
SEE COMMANDS SECTION FOR COMMAND CODES							

COMMAND REGISTERS (WRITE ONLY)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
UPPER REGISTER							
Not used				MSB's			

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LOWER REGISTER (LBS's)							
H'0000' = 0 H'0001' = 1 THRU H'3FFE' = 16,382 H'3FFF' = 16,383							Note: MSB's are in Upper Register [5:0]

NOTE:
1. Bits 7 and 6 of upper register are not used in the cursor address register.

CURSOR ADDRESS REGISTERS (READ AND WRITE)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
UPPER REGISTER							
DADD15	DADD14			MSB's			

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LOWER REGISTER (LSB)							
H'0000' = 0 H'0001' = 1 THRU H'3FFE' = 16,382 H'3FFF' = 16,383							NOTE: MSB's ARE IN UPPER REGISTER [5:0]

SCREEN START 1 REGISTERS (READ AND WRITE)

NOTES:
2. Bits 7 and 6 of upper register are always zero when read by the CPU.
3. When IR0(7) = 1, the values written into bits 7 and 6 of screen start 1 upper will also be written into IR14(7:6) to control the double width and double height attributes of the display as follows:

7	6	Attribute
0	0	None
0	1	Double width only
1	0	Double width and double
1	1	Double width and double

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
UPPER REGISTER							
SPL2 0 = OFF 1 = ON	SPL1 0 = OFF 1 = ON			MSB's			

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LOWER REGISTER							
H'0000' = 0 H'0001' = 1 THRU H'3FFE' = 16,382 H'3FFF' = 16,383							NOTE: MSB's ARE IN UPPER REGISTER [5:0]

SCREEN START 2 REGISTERS (READ AND WRITE)

NOTES:
1. Bit 7 and bit 6 are always zero when read by CPU.
2. These bits should be set to zero after power up by the user.

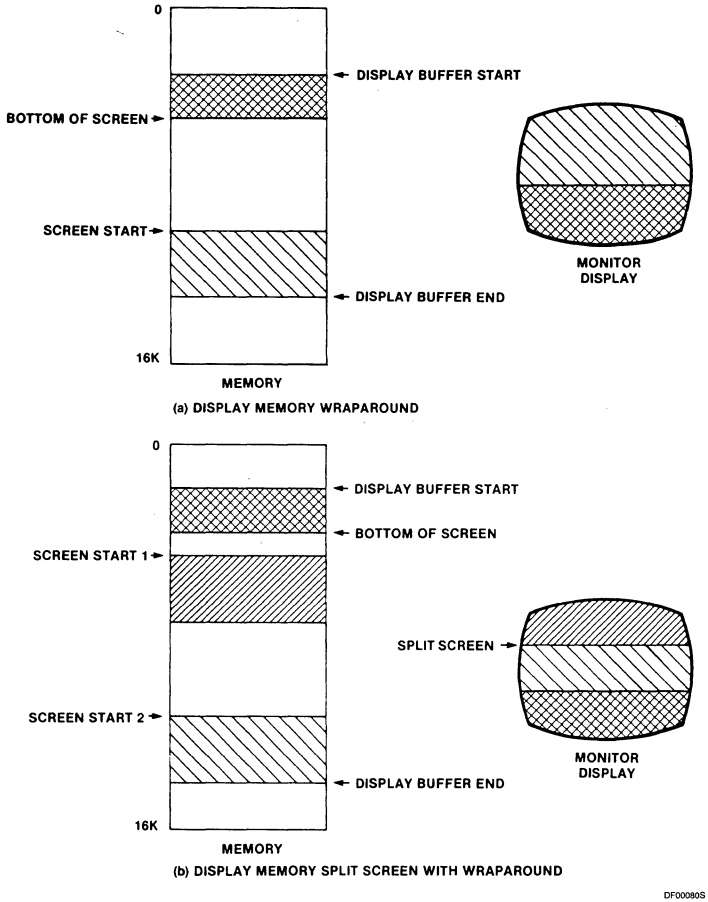


Figure 17. Display Addressing Operation

Table 5. INTERRUPT AND STATUS REGISTER BIT FORMATS

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
NOT USED ALWAYS READ AS 0		RDFLG	VBLANK	LINE ZERO	SPLIT 1	READY	SPLIT 2
		0 = BUSY 1 = READY *	0 = NO 1 = YES	0 = NO 1 = YES	0 = NO 1 = YES	0 = BUSY 1 = READY	0 = NO 1 = YES

* Status register only. Always 0 when reading interrupt register.

Descriptions of each interrupt/status register bit follow. Unless otherwise indicated, a bit, once set, will remain set until reset by the CPU by issuing a 'reset interrupt/status bits' command. The bits are also reset by a 'master reset' command and upon power-up.

Sr[5] – RDFLG
This bit is present in the status register only. A zero indicates that the AVDC is currently executing the previously issued delayed command. A one indicates that the AVDC is ready to accept a new delayed command. This bit is set to a one upon a master reset.

I/SR[4] – VBLANK
Indicates the beginning of a vertical blanking interval. Set to one at the beginning of the first scan line of the vertical front porch.

I/SR[3] – Line Zero
Set to one at the beginning of the first scan line (line 0) of each active character row.

I/SR[2] – Split Screen 1
This bit is set when a match occurs between the current character row number and the value contained in split register 1, IR12[6:0]. The equality condition is only checked at the beginning of line zero of each character row.

I/SR[1] – Ready
The delayed commands affect the display and may require the AVDC to wait for a blanking interval before enacting the command. This bit is set to one when execution of a delayed command has been completed. No other delayed command should be invoked until the prior delayed command is completed. This bit is set to a zero upon a master reset.

I/SR[0] – Split Screen 2
This bit is set when a match occurs between the current character row number and the value contained in split register 2 (IR13[6:0]) when you are not scrolling. It is set for the value contained in (split screen register 2) + 1 when scrolling.

COMMANDS

The AVDC commands are divided into two classes: the instantaneous commands which are executed immediately after they are invoked, and the delayed commands which may need to wait for a blanking interval prior to their execution. Command formats are shown in table 6. The commands are asserted by performing a write operation to the command register with the appropriate bit pattern as the data byte.

Instantaneous Commands

The instantaneous commands are executed immediately after the trailing edge of the $\overline{WR}$ pulse during which the command is issued. These commands do not affect the state of the RDFLG or READY interrupt/status bits and can be invoked at any time.

Master Reset
This command initializes the AVDC and can be invoked at any time to return the AVDC to its initial state. Upon power-up, two successive master reset commands must be applied to release the AVDC's internal power on circuits. In transparent and shared buffer modes, the CNTRL1 input must be high when the command is issued. The command causes the following:

1. VSYNC and HSYNC are driven low for the duration of the command and BLANK goes high. After command completion, HSYNC and VSYNC will begin operation and BLANK will remain high until a 'display on' command is received.
2. The interrupt and status bits and masks are set to zero, except for the RDFLG flag which is set to a one.
3. The row buffer mode, cursor-off, display-off, and the line graphics disable states are set.
4. The initialization register pointer is set to address IR0.
5. IR2[7] is reset.

Load IR Address
This command is used to preset the initialization register pointer with the value 'V' defined by D3 – D0. Allowable values are 0 to 14.

Enable Graphics
After invoking this command, the AVDC will increment the MAC to the next consecutive memory address for each scan line even if more than one scan line per row is programmed. In other words, each scan line begins with a consecutive address from the last displayed address of the previous scan line. This mode can be used for bit-mapped graphics where each location in the display buffer within the defined area contains the bit pattern to be displayed. The graphics mode will be enabled on the next character row after the 'graphics enable' command has been executed. This allows the user to enter and exit graphics mode on character row boundaries.

To perform split screen operations while in graphics mode use SSR2 only.

DADD0/LG is asserted during the trailing edge of BLANK for each scan line while this mode is active.

The AVDC allows up to 128 character rows (initialization register 4) by 256 characters (initialization register 5). For a higher resolution bit mapped screen, the AVDC is programmed as if there are characters and character rows. For example, screen size of 240 x 512 pixels is possible by programming the AVDC for 20 rows with 12 scan lines per row by 64 characters with 8 dots per character.

In the graphics mode, SSR1 should only be updated during the last scan line of the defined 'character row.'

The bit mapped graphics mode will work only in the independent and transparent modes. Row table addressing will be available as an option with the graphics mode in version T of the AVDC.

Disable Graphics
Normal addressing resumes at the next row boundary.

Display Off
Asserts the BLANK output. The DADD0 through DADD13 display address bus outputs can be optionally placed in the three-state condition by setting bit 2 to a '1' when invoking the command.

Display On
Restores normal blanking operation either at the beginning of the next field (bit 2 = 1) or at the beginning of the next scan line (bit 2 = 0). Also returns the DADD0 – DADD13 drivers to their active state.

Cursor Off
Disables cursor operation. Cursor output is placed in the low state.

Cursor On
Enables normal cursor operation.

Reset Interrupt/Status Bits
This command resets the designated bits in the interrupt and status registers. The bit positions correspond to the bit positions in the registers:

Bit 0 – Split 2

Bit 1 – Ready

Bit 2 – Split 1

Bit 3 – Line zero

Bit 4 – Vertical blank

Disable Interrupts
Sets the interrupt mask to zeros for the designated conditions, thus disabling these conditions from being set in the interrupt register and asserting the $\overline{INTR}$ output. Bit position correspondence is as above.

Enable Interrupts
This command writes the associated interrupt mask bits to a one. This enables the corresponding conditions to be set in the interrupt register and asserts the $\overline{INTR}$ output. Bit position correspondence is as above.

Delayed Commands

This group of commands is utilized for the independent buffer mode of operation, although the 'increment cursor' command can also be used in other modes. With the excep-

Table 6. AVDC COMMAND FORMATS

D7	D6	D5	D4	D3	D2	D1	D0		COMMAND
Instantaneous Commands:									
0	0	0	0	0	0	0	0		Master reset
0	0	0	1	V	V	V	V		Load IR pointer with value V (V = 0 to 14)
0	0	1	d	d	d	1	0[1]		Disable graphics
0	0	1	d	d	d	1	1[2]		Enable graphics
0	0	1	d	1	N	d	0[1]		Display off. Float DADD bus if N = 1
0	0	1	d	1	N	d	1[2]		Display on: Next field (N = 1) or scan line (N = 0)
0	0	1	1	d	d	d	0[1]		Cursor off
0	0	1	1	d	d	d	1[2]		Cursor on
0	1	0	N	N	N	N	N		Reset interrupt/status: Bit reset where N = 1
1	0	0	N	N	N	N	N		Disable interrupt: Disable where N = 1
0	1	1	N	N	N	N	N		Enable interrupt: Enables interrupts where N = 1
			V B	L Z	S P 1	R D Y	S P 2		**Interrupt Bit Assignments**
Delayed Commands:								**Hex**	
1	0	1	0	0	1	0	0	A4	Read at pointer address
1	0	1	0	0	0	1	0	A2	Write at pointer address
1	0	1	0	1	0	0	1	A9	Increment cursor address
1	0	1	0	1	1	0	0	AC	Read at cursor address
1	0	1	0	1	0	1	0	AA	Write at cursor address
1	0	1	0	1	1	0	1	AD	Read at cursor address and increment address
1	0	1	0	1	0	1	1	AB	Write at cursor address and increment address
1	0	1	1	1	0	1	1	BB	Write from cursor address to pointer address
1	0	1	1	1	1	0	1	BD	Read from cursor address to pointer address

NOTES:
1. Any combination of these three commands is valid.
2. Any combination of these three commands is valid.
3. d = don't care.

tion of the 'write from cursor to pointer' and 'increment cursor' commands, all the commands of this type will be executed immediately or will be delayed depending on when the command is invoked. If invoked during the active screen time, the command is executed at the next horizontal blanking interval. If invoked during a vertical retrace interval or a 'display off' state, the command is executed immediately.

The 'increment cursor' command is executed immediately after it is issued and requires approximately three $\overline{CCLK}$ periods for completion. The 'write from cursor to pointer' command executes during blanking intervals. The AVDC will execute as many writes as possible during each blanking interval. If the command is not completed during the current blanking interval, the command will be held in suspension during the next active portion of the screen and continues during the next blanking interval until the command is completed.

In all cases, the AVDC will assert the READY/RDFLG status to signify completion of the delayed command. No other delayed command should be given until the previous delayed command has completed. Therefore, the READY interrupt or RDFLG status flag should be used for handshaking control between the AVDC and CPU when using the delayed commands.

Read/Write at Pointer
Transfers data between the display buffer and the bus interface latch using the address contained in the pointer registers.

Read/Write at Cursor
Transfers data between the display buffer and the bus interface latch using the address contained in the cursor registers.

Increment Cursor
Adds one (modulo 16K) to the cursor address registers.

Read/Write at Cursor and Increment
Transfers data between the display buffer and the bus interface latch using the address contained in the cursor registers and then adds one (modulo 16K) to the cursor address registers.

Write from Cursor to Pointer
Writes the data contained in the bus interface latch into the block of display memory designated by the cursor address and pointer address registers, inclusive. After completion of the command, the pointer address will be unchanged, but the cursor register contents will be equal to the pointer address.

Read from Cursor to Pointer
Writes the data from the block of display memory designated by the cursor and pointer addresses inclusive into the bus interface latch. This command can be used for a DMA dump of memory into RAM from the cursor location to the pointer location. After completion of the command, the cursor register contents will equal the pointer register contents.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6]

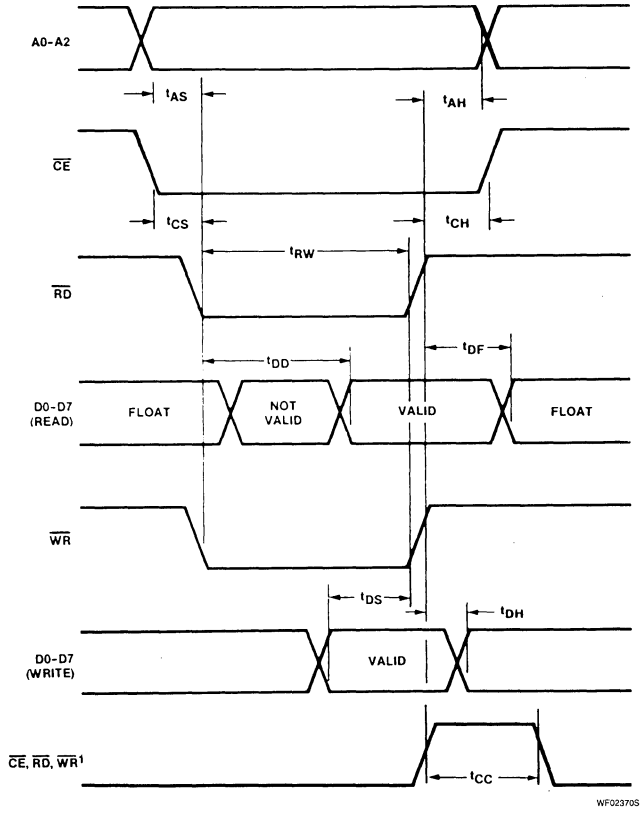
PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		2.0			V
V_{OL}	Output low voltage	$I_{OL} = 2.4mA$			0.4	V
V_{OH}	Output high voltage (except $\overline{INTR}$ output)	$I_{OH} = -200\mu A$	2.4			V
I_{IL}	Input leakage current	$V_{IN} = 0$ to V_{CC}	−10		10	μA
I_{LL}	Data bus 3-state leakage current	$V_O = 0$ to V_{CC}	−10		10	μA
I_{OD}	$\overline{INTR}$ open drain output leakage current	$V_O = 0$ to V_{CC}			10	μA
I_{CC}	Power supply current				185	mA

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6,7]

PARAMETER		TEST CONDITIONS[8]	2.7MHz		4.0MHz		UNIT
			Min	Max	Min	Max	
Bus timing (figure 18)[9]							
t_{AS}	A0 – A2 set-up time to $\overline{WR}$, $\overline{RD}$ low		30		30		ns
t_{AH}	A0 – A2 hold time from $\overline{WR}$, $\overline{RD}$ high		0		0		ns
t_{CS}	$\overline{CE}$ set-up time to $\overline{WR}$, $\overline{RD}$ low		0		0		ns
t_{CH}	$\overline{CE}$ hold time from $\overline{WR}$, $\overline{RD}$ high		0		0		ns
t_{RW}	$\overline{WR}$, $\overline{RD}$ pulse width		250		200		ns
t_{DD}	Data valid after $\overline{RD}$ low			200		200	ns
t_{DF}	Data bus floating after $\overline{RD}$ high			100		100	ns
t_{DS}	Data set-up time to $\overline{WR}$ high		150		150		ns
t_{DH}	Data hold time from $\overline{WR}$ high		10		5		ns
t_{CC}	High time from $\overline{CE}$ to $\overline{CE}$						
	Consecutive commands		t_{CCP}		t_{CCP}		ns
	Other accesses		300		300		ns
CCLK timing (figure 19, 20, 21)							
t_{CCP}	$\overline{CCLK}$ period		370	10,000	250	10,000	ns
t_{CCH}	$\overline{CCLK}$ high time		125		100		ns
t_{CCL}	$\overline{CCLK}$ low time		125		100		ns
	Output delay from $\overline{CCLK}$ edge[11]						
t_{CCD1}	DADD0 – 13, MBC		40	175	40	150	ns
t_{CCD2}	BLANK, HSYNC, VSYNC/CSYNC, CURSOR, $\overline{BEXT}$, $\overline{BREQ}$, $\overline{BACK}$, $\overline{BCE}$, $\overline{WDB}$, $\overline{RDB}$[10]		40	225	40	200	ns
Other timings (figure 20)							
t_{RDL}	READY/RDFLG low from $\overline{WR}$ high[9]			$t_{CCP}+30$		$t_{CCP}+30$	ns
t_{BAK}	$\overline{BACK}$ high from $\overline{PBREQ}$ low			225		200	ns
t_{BXT}	$\overline{BEXT}$ high from $\overline{PBREQ}$ high			225		200	ns
t_{IRL}	$\overline{INTR}$ low from $\overline{CCLK}$ low			225		200	ns
t_{IRH}	$\overline{INTR}$ high from $\overline{WR}$, $\overline{RD}$ high[9]			600		600	ns
t_{AC}	ACLL from HSYNC		$3xt_{CCP}$		$3xt_{CCP}$		ns
Row table input timing (figure 21)							
t_{DSRT}	Data set-up time to $\overline{CCLK}$ low		100		60		ns
t_{DHRT}	Data hold time from $\overline{CCLK}$ low		60		60		ns

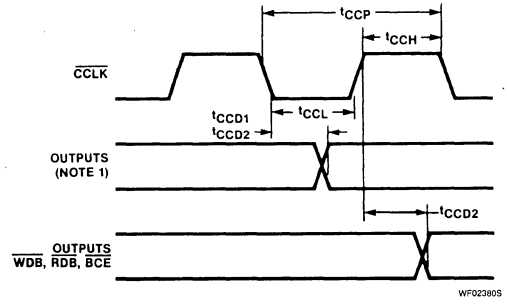
NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any conditions above those in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND).
6. Typical values are at +25°C, typical supply voltages, and typical processing parameters.
7. For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
8. Test condition for outputs: $C_L = 150pF$.
9. Timing is illustrated and specified referenced to $\overline{WR}$ and $\overline{RD}$ inputs. Device may also be operated with $\overline{CE}$ as the 'strobing' input. In this case, all timing specifications apply referenced to falling and rising edges of $\overline{CE}$.
10. $\overline{BCE}$, $\overline{WDB}$, and $\overline{RDB}$ delays track each other within 10nsec. Also, these output delays will tend to follow direction (min/max) of DADD0 – 13 delays.
11. These values were measured with a capacitance load of 150pF. To adjust the output delay, use the following correction factor: $50pF \leq C_L < 150pF$: −0.15ns/pF



NOTE:
1. Any two must be high for t_{CC}.

Figure 18. Bus Timing



NOTES:
1. DADD0 – DADD13, BLANK, HSYNC, CSYNC/VSYNC, CURSOR, $\overline{EXT}$, $\overline{REQ}$, $\overline{BCE}$, MBC, $\overline{BACK}$.
2. $\overline{BCE}$ changes state on both $\overline{CCLK}$ edges — (see figures 3 and 4).

Figure 19. $\overline{CCLK}$ Timing

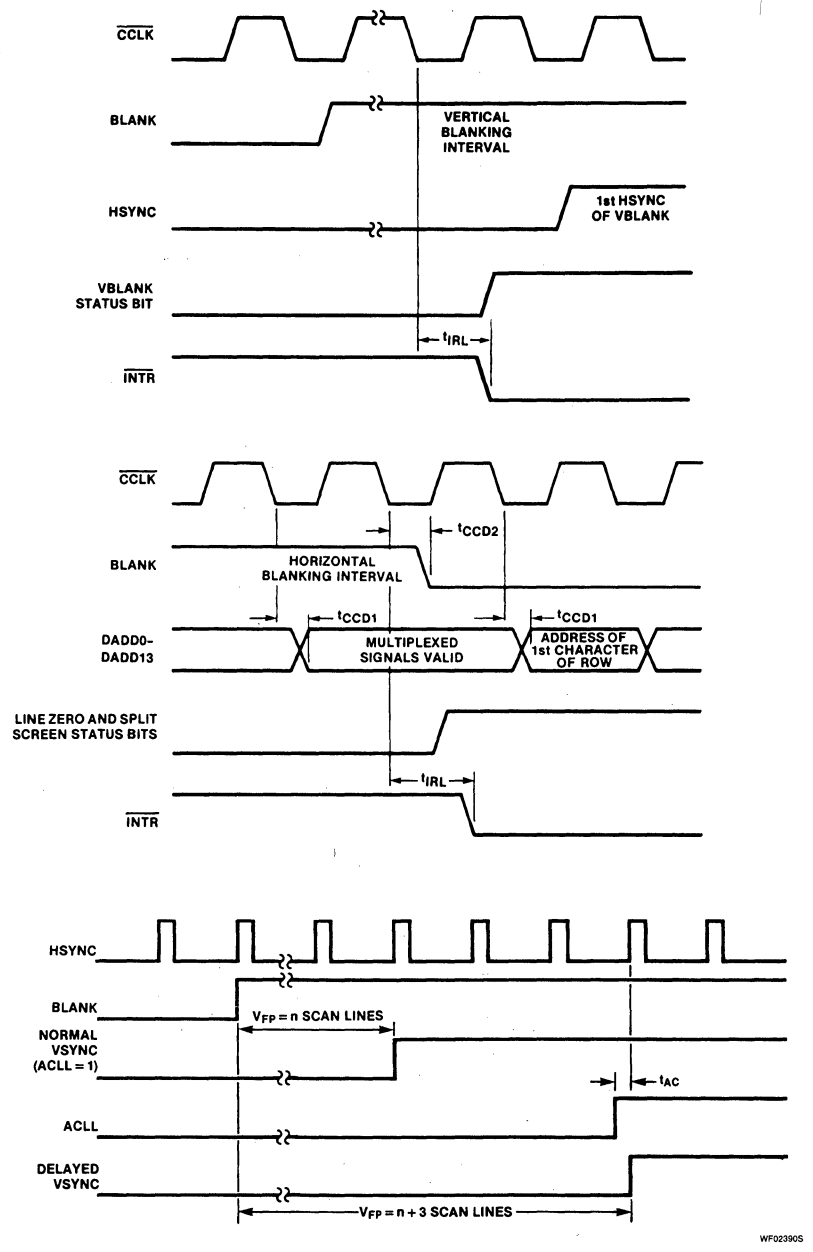


Figure 20. Other Timings

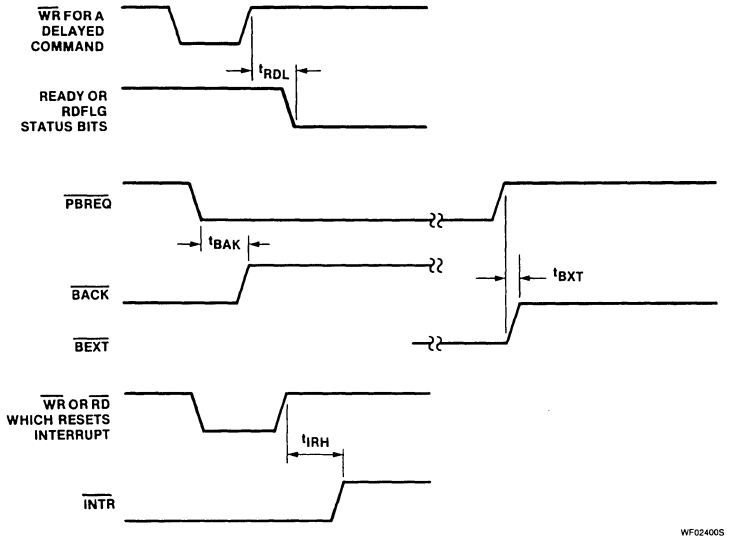


Figure 20. Other Timings (Continued)

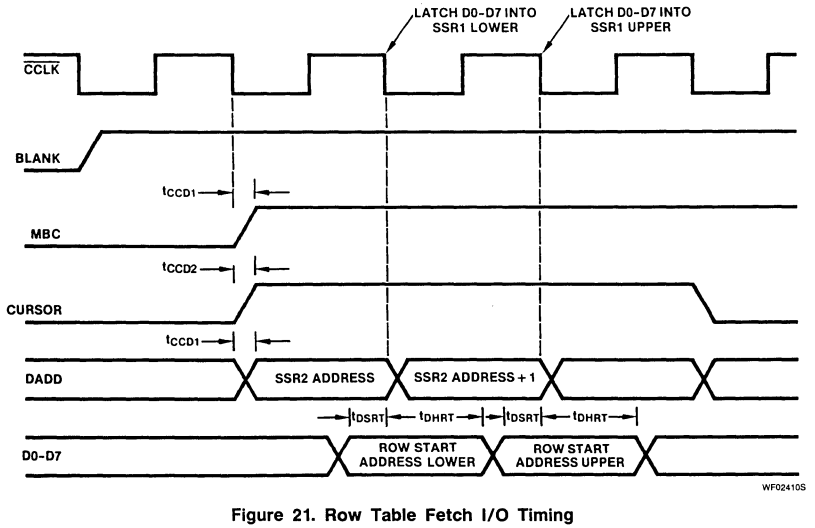
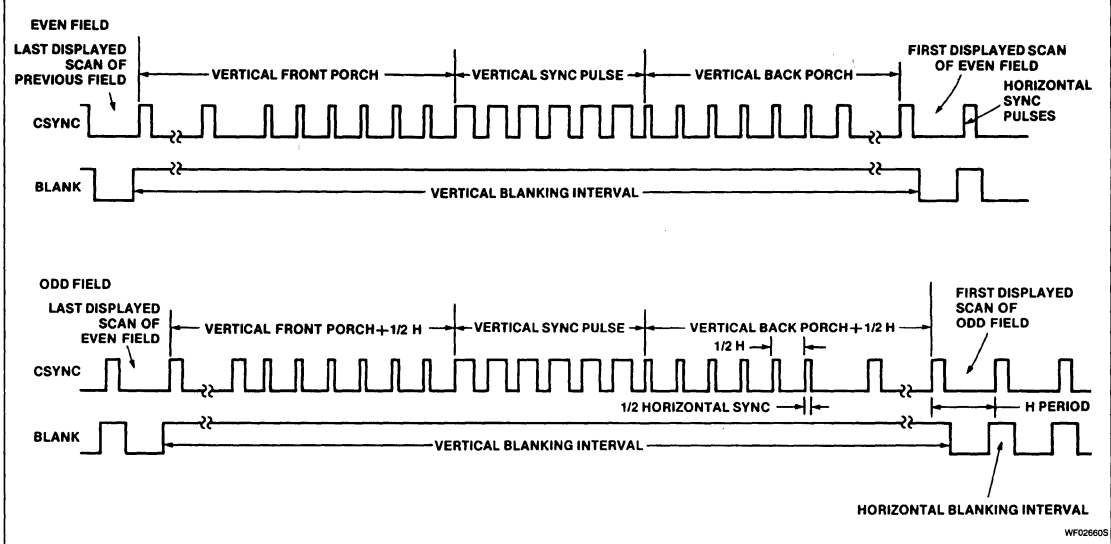


Figure 21. Row Table Fetch I/O Timing



NOTES:
1. In non-interlaced operation the even field is repeated continuously.
2. In Interlaced operation the even field alternates with the odd field.

Figure 22. Composite Sync Timing

Signetics

SCB2675
Color/Monochrome Attributes Controller (CMAC)

Product Specification

Microprocessor Products

2

DESCRIPTION

The Signetics SCB2675 Color/Monochrome Attributes Controller (CMAC) is a bipolar LSI device designed for CRT terminals and display systems that employ raster scan techniques. It contains a programmable dot clock divider to generate a character clock, a high speed shift register to serialize input dot data into a video stream, latches and logic to apply visual attributes to the resulting display, and logic to display a cursor on the display.

The CMAC provides control of visual attributes on a character by character basis for two operating modes: monochrome and color. The monochrome mode provides reverse video, blank, highlight and two general purpose user definable attributes. In this mode, the display characters can be specified to appear on either a light or dark screen background. Retrace video suppression can be automatically or externally controlled. The color mode provides eight colors for foreground (character) video and eight colors for background video together with a luminance output for external color set selection or to simultaneously drive a monochrome monitor. Additionally, both modes provide double width, underline, blink, dot stretching and dot width attributes. In monochrome mode, the SCB2675 emulates the attribute characteristics of Digital Equipment Corporation's VT100 terminal.

The horizontal dot frequency is the basic timing input to the CMAC. This clock is divided internally to provide a character clock output for system synchronization. Up to nine bits of dot data are parallel loaded into the video shift register on each character boundary. The two TTL video data outputs in monochrome mode are encoded to provide four video intensities (black, gray, white and highlight). The video data in color mode is encoded to provide eight foreground colors and shifted out on three TTL outputs, together with the luminance output.

FEATURES

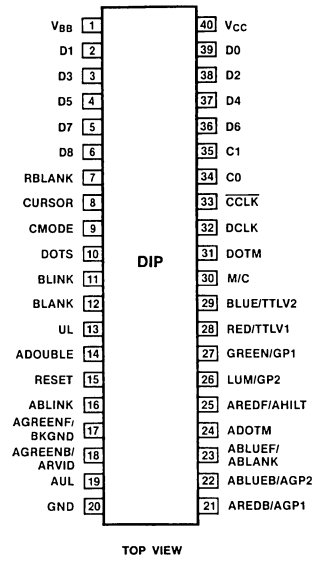
- **25MHz and 18MHz video dot rate versions***
- **Four video intensities encoded on two TTL outputs (monochrome mode)**
- **Eight foreground and background colors encoded on three TTL outputs (color mode)**
- **Internally latched character attributes:**
  - **Reverse video**
  - **Blank**
  - **Blink**
  - **Underline**
  - **Highlight**
  - **Two general purpose**
  - **Eight foreground colors**
  - **Eight background colors**
  - **Dot width control**
  - **Double width characters**
- **VT100 compatible attributes**
- **Reverse video cursor with optional white cursor in color mode**
- **Up to 10 dots per character**
- **Light or dark background in monochrome mode**
  - **Automatic retrace blanking**
- **Programmable dot stretching**
- **Compatible with SCN2674 AVDC and SCN2670 DCGG**
- **TTL compatible**
- **40-pin dual in-line package**

APPLICATIONS

- **CRT terminals**
- **Word processing systems**
- **Small business computers**

*40MHz SCB2675T also available.

PIN CONFIGURATION



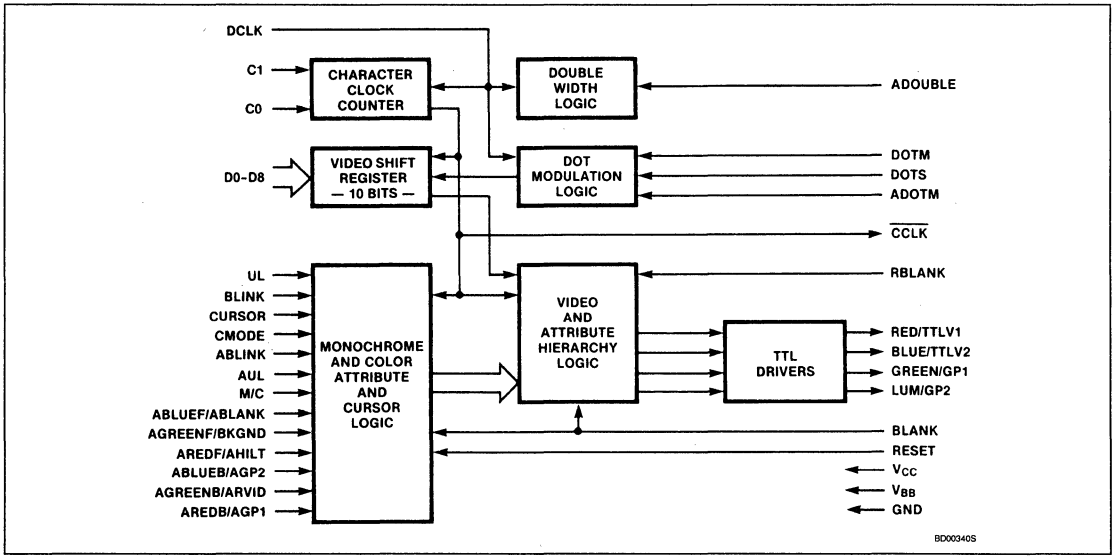
Pin	Function	Pin	Function
2	V_{BB}	24	AREDB/AGP1
3	D1	25	ABLUEB/AGP2
4	D3	26	ABLUEF/ABLANK
5	D5	27	ADOTM
6	D7	28	AREDF/AHILT
7	D8	29	LUM/GP2
8	RBLANK	30	GREEN/GP1
9	CURSOR	31	RED/TTLV1
10	CMODE	32	BLUE/TTLV2
11	DOTS	33	M/C
13	BLINK	35	DOTM
14	BLANK	36	DCLK
15	UL	37	$\overline{CCLK}$
16	ADOUBLE	38	C0
17	RESET	39	C1
18	ABLINK	40	D6
19	AGREENF/BKGND	41	D4
20	AGREENB/ARVID	42	D2
21	AUL	43	D0
22	GND	44	V_{CC}

Color/Monochrome Attributes Controller (CMAC) SCB2675

ORDERING CODE

PACKAGES	DOTS PER CHARACTER	V_{CC} = 5V ±5%, 0°C to +70°C	
		25MHz	18MHz
Ceramic DIP Plastic DIP Plastic LCC	7, 8, 9, 10	SCB2675BC5I40 SCB2675BC5N40 SCB2675BC5A44	SCB2675BC8I40 SCB2675BC8N40 SCB2675BC8A44
Ceramic DIP Plastic DIP Plastic LCC	6, 8, 9, 10	SCB2675CC5I40 SCB2675CC5N40 SCB2675CC5A44	SCB2675CC8I40 SCB2675CC8N40 SCB2675CC8A44

BLOCK DIAGRAM



Color/Monochrome Attributes Controller (CMAC) SCB2675

PIN DESCRIPTION

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
V_{CC}	40	44	I	**Power supply:** +5VDC
V_{BB}	1	2	I	**Bias Supply:** See figure 5
GND	20	22	I	**Ground:** 0V reference
DCLK	32	36	I	**Dot Clock:** Dot frequency input. Video output shift rate.
$\overline{CCLK}$	33	37	O	**Character Clock:** An output which is a submultiple of DCLK. The period ranges from 6 to 10 DCLK periods per cycle and is determined by the state of the C0 – C1 inputs.
RED/TTLV1	28	31	O	**Red/TTL Video 1:** In color mode, this output provides the red gun serial video. In monochrome mode, it should be used with the blue/TTL video 2 output to decode four video intensities.
BLUE/ TTLV2	29	32	O	**Blue/TTL Video 2:** In color mode, this output provides the blue gun serial mode. In monochrome mode, it should be used with the red/TTL video 1 output to decode four video intensities.
GREEN/GP1	27	30	O	**Green/General Purpose 1:** In color mode, this output provides the green gun serial video. In monochrome mode, it is a general purpose TTL output which is asserted if the AREDB/AGP1 input is asserted when the corresponding character dot data is loaded into the video shift register. GP1 can be active in either active scan or blank time.
LUM/GP2	26	29	O	**Luminance/General Purpose 2:** In color mode, this output is the logical-OR of the RGB foreground video. It is low during a blanking interval and during the foreground portion of the cursor display. In monochrome mode, it is a general purpose TTL output which is asserted if the ABLUEB/AGP2 input is asserted when the corresponding character dot data is loaded into the video shift register. GP2 can be active in either active scan or blank time.
UL	13	15	I	**Underline Timing:** Indicates the scan line(s) for the underline attribute. Latched on the falling edge of BLANK. The underline will be displayed on the specified scan line for every character where AUL = 1.
BLINK	11	13	I	**Blink Timing:** This input is sampled on the falling edge of BLANK to provide the blink rate for the blink attribute. Should be a submultiple of the frame rate.
BLANK	12	14	I	**Screen Blank:** When high, this input forces the video outputs to the specified background color in color mode and to the level specified by the BKGND input (either black or gray) in monochrome mode.
RBLANK	7	8	I	**Retrace Blank:** This input is used to force the video outputs to a low during retrace periods. If pulled high, it will automatically suppress video during the retrace periods when BLANK is high. The user may also pulse this input while BLANK is high to selectively suppress raster video.
AGREENF/ BKGND	17	19	I	**Green Foreground/Background Intensity:** In color mode, this input activates the GREEN/GP1 output during the foreground (character video) portion of the associated character block. In monochrome mode, this input specifies gray or black screen background.
ABLUEF/ ABLANK	23	26	I	**Blue Foreground/Blank Attribute:** In color mode, this input activates the BLUE/TTLV2 output during the foreground (character video) portion of the associated character block. In monochrome mode, this input generates a blank space for the associated character. The blank space intensity is controlled by the AGREENF/ BKGND input, the reverse video attribute and cursor input.
AREDF/ AHILT	25	28	I	**Red Foreground/Highlight Attribute:** In color mode, this input activates the RED/TTLV1 output during the foreground (character video) portion of the associated character block. In monochrome mode, this input highlights the associated character (including underline).
CURSOR	8	9	I	**Cursor Timing:** This input provides the timing for the cursor video. In color mode, with CURSOR and CMODE high, the RGB outputs are driven high (white cursor). If CMODE is low, or in monochrome mode, this input reverses the intensities of the video and attributes (the foreground and background intensities are reversed). Cursor position, shape, and blink rate are controlled by this input.
CMODE	9	10	I	**Cursor Mode:** Used in color mode only. When CURSOR and CMODE are high, the RGB outputs are driven high (white cursor) When CURSOR is high and CMODE is low, the RGB outputs are logically inverted (reverse video cursor).
AUL	19	21	I	**Underline Attribute:** Specifies a line to be displayed in the character block. The specific line(s) are specified by the UL input. All other attributes apply to the underline video.
ABLINK	16	18	I	**Blink Attribute:** In color mode, this active high input will drive the foreground RGB combination to the background RGB combination. In monochrome mode, the associated character or background is driven to the intensity determined by BKGND, reverse video attribute and the cursor input.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
ADOUBLE	14	16	I	**Double Width Attribute:** This active high input causes the associated character video to be shifted out of the serial shift register at one half the dot frequency (DCLK). The $\overline{CCLK}$ output is not affected.
AREDB/ AGP1	21	24	I	**Red Background/General Purpose Attribute 1:** In color mode, this input activates the RED/TTLV1 output during the background portion of the associated character block. In monochrome mode, it activates the GREEN/ GP1 output for the associated character block.
ABLUEB/ AGP2	22	25	I	**Blue Background/General Purpose Attribute 2:** In color mode, this input activates the BLUE/TTLV2 output during the background portion of the associated character block. In monochrome mode, it activates the LUM/GP2 output for the associated character block.
AGREENB/ ARVID	18	20	I	**Green Background/Reverse Video Attribute:** In color mode, this input activates the GREEN/GP1 output during the background portion of the associated character block. In monochrome mode, it causes the associated character block video intensities to be reversed.
D0 - D8	39, 2, 38, 3, 37, 4, 36, 5, 6	43, 3, 42, 4, 41, 5, 40, 6, 7	I	**Dot Data Input:** These are parallel inputs corresponding to the character/graphic symbol dot data for a given scan line. These inputs are strobed into the video shift register on the trailing (falling) edge of each character clock ($\overline{CCLK}$).
C0, C1	34, 35	38, 39	I	**Character Clock Control:** The states of these two static inputs determine the internal divide factor for the $\overline{CCLK}$ output rate.
RESET	15	17	I	**Reset:** This active high input initializes the internal logic and resets the attribute latches.
M/C	30	33	I	**Monochrome/Color Mode:** This input selects whether the CMAC operates in monochrome or color mode. A low selects color mode and a high selects monochrome mode.
ADOTM	24	27	I	**Dot Modulation Attribute:** When DOTM and this input are high, the active dot width of the associated character video is one DCLK. When DOTM is high and this input is low, the active dot width of the associated character video is two DCLKs.
DOTM	31	35	I	**Dot Width Modulation:** When this input is high, two DCLKs are used for each dot shifted through the shift register. When this input is low, one DCLK is used.
DOTS	10	11	I	**Dot Stretching:** Sampled at the falling edge of BLANK. When this input is high, one extra dot is appended to individual dots or groups of dots of the input parallel data and then transferred through the shift register. When this input is low, normal transfer of input parallel data results.

FUNCTIONAL DESCRIPTION

The CMAC consists of seven major sections (see block diagram). The high speed dot clock input is applied to a programmable divider to provide a character clock output for system timing. Parallel dot data is loaded into the video shift register on character boundaries and shifted into the video logic block at the dot rate specified by the dot modulation section. The appropriate attribute control inputs are selected by the mode select logic, latched internally on character boundaries, and combined with the serial dot data to provide monochrome or color video outputs.

The BLANK input defines the active screen and retrace areas. In color mode, the video outputs are forced to the specified background color when this signal is asserted; in monochrome mode the video outputs are forced to the states defined by the BKGND input, i.e., black if dark background is selected and gray if light background is selected. A separate RBLANK input allows the user to select the amount of border around the active area when operating in color mode or in monochrome mode with light background. This input can be tied high, in which case the

Table 1. MONOCHROME MODE ATTRIBUTE CHARACTERISTICS

REV[1]	AHILT	ABLINK[2]	FOREGROUND VIDEO	BACKGROUND VIDEO
0	0	0	W	B
0	0	1	W/G	B
0	1	0	H	B
0	1	1	H/W	B
1	0	0	B	G
1	0	1	B/W	G/B
1	1	0	B	W
1	1	1	B/H	W/B

NOTES:

1. REV = (BKGND) XOR (ARVID):

BKGND	ARVID	REV
0	0	0
0	1	1
1	0	1
1	1	0

2. For blinking, the video outputs are shown as 0/1, where 0 and 1 are the blink timing input states.
3. Foreground includes underline when underlining is specified by AUL = 1.
4. When ABLANK = 1, foreground component becomes same as background component.
5. Codes for video outputs are as follows:

CODE	TTLV2	TTLV1	BEAM INTENSITY
B	0	0	Black (B)
G	0	1	Gray (G)
W	1	0	White (W)
H	1	1	Highlight (H)

area outside the active area will be dark, or it may be pulsed during BLANK periods to externally control the border widths.

In color mode, eight colors for the character (foreground) and eight colors for the background (area other than character) can be selected by the attribute inputs. In monochrome mode, the intensities of foreground and background are a function of the attribute and BKGND inputs, i.e., characters may be black, gray, white, or highlight (very white) while background may be black, gray, or white (see table 1).

Character Clock Counter

The character clock counter divides the DCLK input to generate the character clock ($\overline{CCLK}$). The divide factor is specified by the clock control inputs (C1-C0) as follows:

C1	C0	SCB2675B	
		Dots/ Char.	$\overline{CCLK}$ Duty Cycle*
0	0	10	5/5
0	1	7	3/4
1	0	8	4/4
1	1	9	4/5

*High/low

C1	C0	SCB2675C	
		Dots/ Char.	$\overline{CCLK}$ Duty Cycle*
0	0	10	5/5
0	1	6	3/3
1	0	8	4/4
1	1	9	4/5

*High/low

The number of dot clocks/character is normally the number of dots/character as listed above. However, when dot width control is specified, the DCLK input is divided by two before it is applied to the character clock counter resulting in the number of dot clocks/character being double those listed above, although the number of displayed dots/character remains the same. The number of dots per character (C1 – C0) can be dynamically changed on either edge of the character clock as shown in figure 10. See Dot Modulation Logic.

Video Shift Register

On each character boundary, the parallel input dot data (D0 – D8) is loaded into the video shift register. The data is shifted out least significant bit first (D0) at the DCLK rate. If 10 dots/character are specified (C1 – C0 = 00), the tenth dot will be the same as D8. The serial dot data from the video shift register is routed to the video logic where it is combined with the cursor and attribute control bits to produce the video data outputs.

Mode Select, Attribute And Cursor Control

The mode select logic multiplexes the monochrome and color attribute inputs and outputs as specified by the M/C input. The monochrome mode provides blank, reverse video, highlight and two general purpose attributes. The latter may be used, with external logic, to combine other attributes (e.g., overscore) into the video stream. The color mode provides RGB foreground and background color attributes. Both modes provide double width characters, blink, underline, dot width control and dot stretching.

The cursor and attribute inputs are pipelined internally to allow for system pipeline propagations. The cursor input and the attribute inputs are delayed for one $\overline{CCLK}$ to account for the delay of the character data through the character generator latches. The attribute timing inputs (BLINK, UL and DOTS) are clocked into the 2675 at the beginning of each scan line time by the falling edge of BLANK. Thus, these inputs must be in their proper state at the falling edge of BLANK preceding the scan line where they are required to be active. The BLANK signal itself is also delayed internally to provide for the RAM and character generator delays (see figures 6 and 7). Internal delays cause the video outputs to be delayed relative to $\overline{CCLK}$ as illustrated in figure 8.

Video Logic

Each character block consists of the three components shown in figure 1. Symbol video is generated from the dot data inputs D0-D8. Underline video is enabled by the AUL attribute and is generated during the scan lines for which the UL input is active. Underline and symbol video are always the same intensity or color, and other attributes (e.g., ABLINK) apply to them equally. The combination of underline and symbol video is also referred to as foreground video. Background video is the area of the character block corresponding to the absence of foreground video. The assertion of the non-display attribute in the monochrome mode (ABLANK) causes the entire character block to be displayed as background.

In monochrome mode, the serial dot data and pipelined cursor and attributes are combined to generate four video intensities (black, gray, white and highlight) which are encoded on the TTLV1 and TTLV2 outputs as follows:

TTLV2	TTLV1	VIDEO INTENSITY
0	0	Black
0	1	Gray
1	0	White
1	1	Highlight

Table 1 describes the relationship between attributes and video intensity of the foreground and background components of the character block in monochrome mode.

In color mode, the colors of the foreground and background components are specified by the corresponding attribute inputs; AREDF, AGREENF and ABLUEF dictate the color of the foreground component while AREDB, AGREENB and ABLUEB do the same for the background component. In this mode, the serial dot data and pipelined cursor and attributes are combined to generate four video outputs. The RED, GREEN and BLUE outputs separately contain the corresponding foreground and background components. The LUM output is the logical-OR of the foreground colors and can be used to drive a separate monochrome monitor or to select a different set of colors for the foreground.

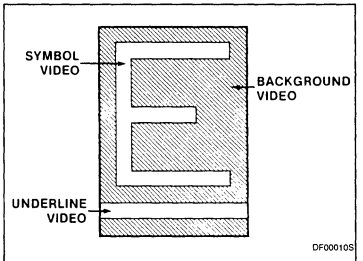


Figure 1. Character Block Definition

Dot Modulation Logic

The dot modulation logic controls the video shift register to supply dot stretching and dot width control.

Dot stretching is controlled by the DOTS input which is sampled each scan line at the trailing (falling) edge of BLANK. If DOTS is asserted at that time, all characters on the following scan line will have dot stretching applied. Dot stretching causes an extra dot to be added to individual dots or groups of dots as shown in figures 2 and 3. Dot stretching can be used to:

1. Compensate for low video bandwidth monitors (since the minimum active displayed segment with dot stretching is two DCLKs).
2. Assure crisp black characters when operating in white background mode.
3. Provide thick characters as a means of distinguishing areas of the display.

Dot width is controlled by the DOTM and ADOTM inputs. DOTM is tied either high, which enables the feature on the entire display, or low, which disables the feature. With ADOTM high, the dot width of characters can be selectively controlled by assertion of the ADOTM attribute input. When operating in this mode, the dot clock input is divided by two before being applied to other circuits in the CMAC. The $\overline{CCLK}$ output is also divided by two.

The truth table for the possible combinations of ADOTM and DOTM is as follows:

ADOTM	DOTM	OPERATION
0	0	Normal mode
0	1	Dot width control, 100% duty cycle (DCLK÷2, CCLK÷2)
1	0	Not allowed
1	1	Dot width control, 50% duty cycle (DCLK÷2, CCLK÷2)

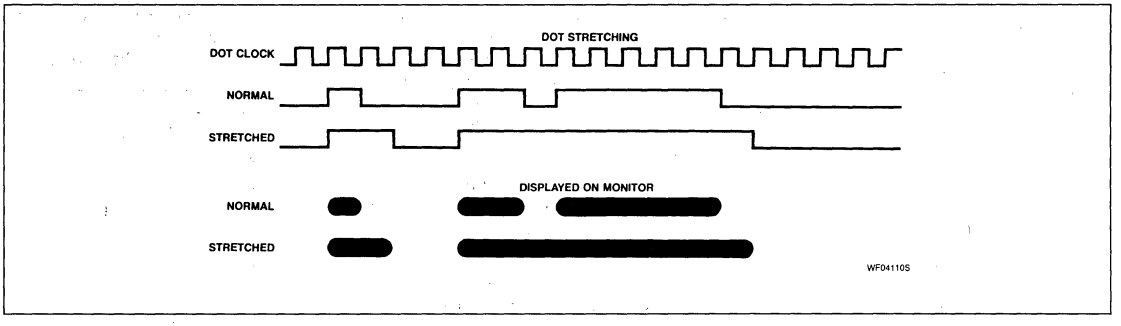
When dot width control is enabled as above, two DCLKs are used for each video dot period. Asserting ADOTM for a particular character will cause each active video dot of the displayed character to be turned on for one DCLK and off for the other DCLK, while if ADOTM is negated for that character, the active video dot for that character will be turned on (black background) or off (white background) for both DCLK times (see figures 2 and 4). Only the character video component of the character block is modulated. Underline video and background are not affected by ontime modulation. Width control can be used to:

1. Make horizontal lines and vertical lines appear the same brightness on the display.
2. Provide two different brightness levels for characters without requiring a monitor with analog brightness inputs.

However, note that the effects produced by this feature are highly dependent on the video amplifier characteristics of the monitor used.

Double Width Logic

The double width logic controls the rate at which dots are shifted through the video shift register. When the ADOUBLE input is asserted, the associated character video will be shifted at one half the DCLK rate, and the dot information for the next character will be loaded into the shift register two $\overline{CCLK}$s later. The character and attribute data that is present on the first CCLK after the ADOUBLE input is asserted will be ignored. The $\overline{CCLK}$ output is not affected. If a double width character is specified at the last location of a character row, the second half of the double width character (one $\overline{CCLK}$) will extend into the horizontal front porch.



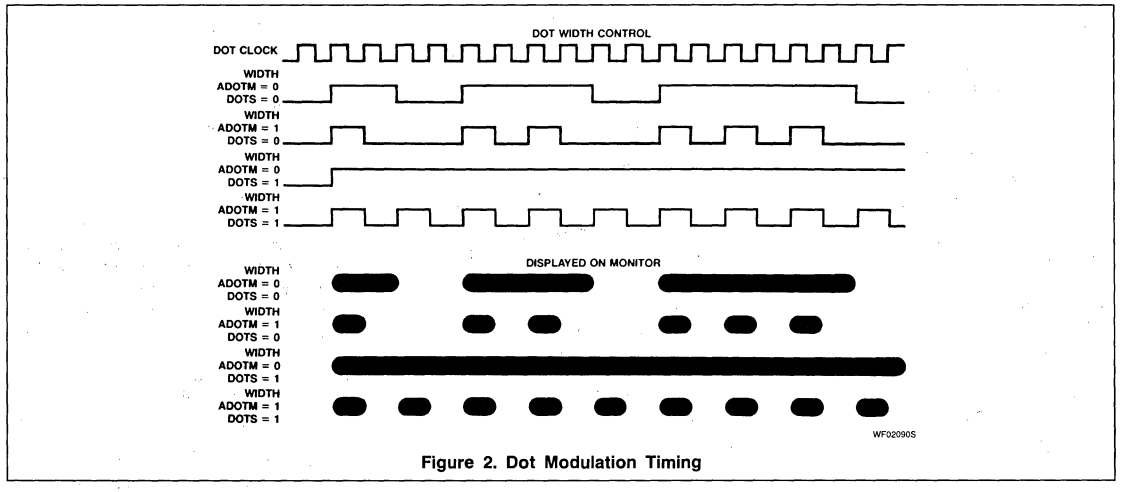


Figure 2. Dot Modulation Timing

Figure 3. Dot Stretching

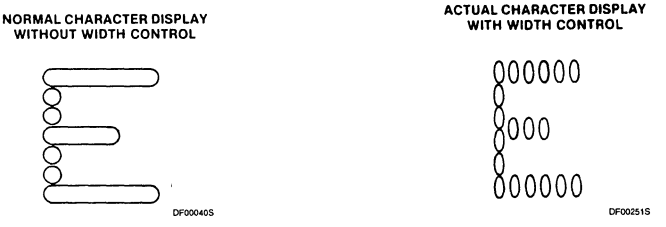


Figure 4. Dot Width Control

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = figure 5[4,5]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		2.0			V
V_{OL}	Output low voltage	$I_{OL} = 4mA$			0.4	V
V_{OH}	Output high voltage	$I_{OH} = -400\mu A$	2.4			V
I_{IL}	Input low current	$V_{IN} = 0.4V$				
	DCLK				−800	μA
	All other inputs				−400	μA
I_{IH}	Input high current	$V_{IN} = 2.4V$				
	DCLK				40	μA
	All other inputs				20	μA
I_{CC}	V_{CC} supply current	$V_{IN} = 0V$, V_{CC} = max			80	mA
I_{BB}	V_{BB} supply current	figure 5			120	mA

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = figure 5[4,5]

PARAMETER		TEST CONDITIONS	LIMITS				UNIT
			25MHz Version		18MHz Version		
			Min	Max	Min	Max	
Dot clock timing (see figure 8)							
f_D	Frequency			25		18	MHz
t_{DH}	High time		15		22		ns
t_{DL}	Low time		15		22		ns
Set-up times (see figures 6, 7, 9, and 10)							
t_{SB}	BLANK to $\overline{CCLK}$		40		50		ns
t_{SA}	Attributes to $\overline{CCLK}$		40		50		ns
t_{SD}	D0 – D9 to $\overline{CCLK}$		70		70		ns
t_{SK}	CURSOR to $\overline{CCLK}$		40		50		ns
t_{SC}	C0, C1 to DCLK		30		35		ns
t_{SR}	RBLANK to DCLK		20		20		ns
t_{SM}	BLINK, UL, DOTS to BLANK		20		20		ns
Hold times (see figures 6, 7, 9, and 10)							
t_{HB}	BLANK from $\overline{CCLK}$		20		20		ns
t_{HA}	Attributes from $\overline{CCLK}$		20		20		ns
t_{HD}	D0 – D8 from $\overline{CCLK}$		30		30		ns
t_{HK}	CURSOR from $\overline{CCLK}$		20		20		ns
t_{HC}	C0, C1 from DCLK		20		20		ns
t_{HR}	RBLANK from DCLK		20		20		ns
t_{HM}	BLINK, UL, DOTS from BLANK		20		20		ns
Delay times (see figure 8)		$C_L = 50pF$					
t_{DC}	$\overline{CCLK}$ from DCLK			55		70	ns
t_{DV}	Other outputs from DCLK			60		70	ns

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying voltages greater than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground. For testing, all input signals swing between 0.4V and 2.4V with a transition time of 3ns maximum and output voltages are checked at 0.8V and 2.0V.

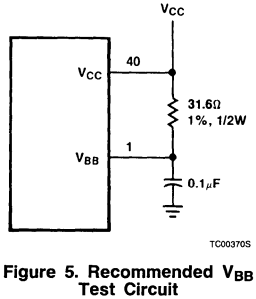
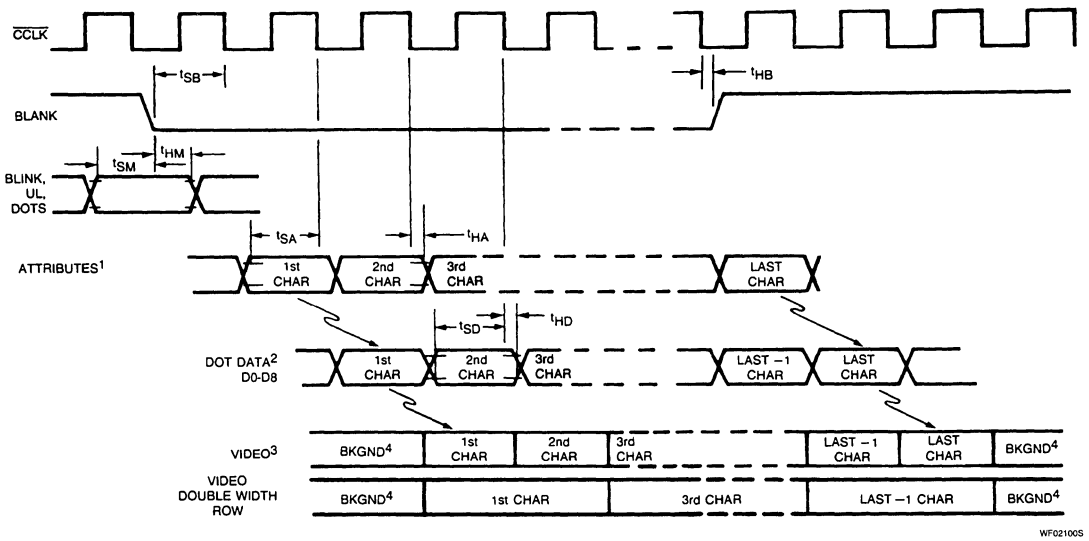


Figure 5. Recommended V_{BB} Test Circuit



NOTES:
1. Attributes include: ABLINK, ABLANK, ARVID, AUL, AHILT, ADOUBLE, ADOTM, two general purpose, and foreground/background colors.
2. One $\overline{CCLK}$ delay for dot data (obtained from delay through character generator).
3. See figure 8 for detail timing of video outputs.
4. Non-active scan time. VIDEO reverts to polarity selected by the BKGND input in monochrome mode.

Figure 6. CMAC Pipeline Timing

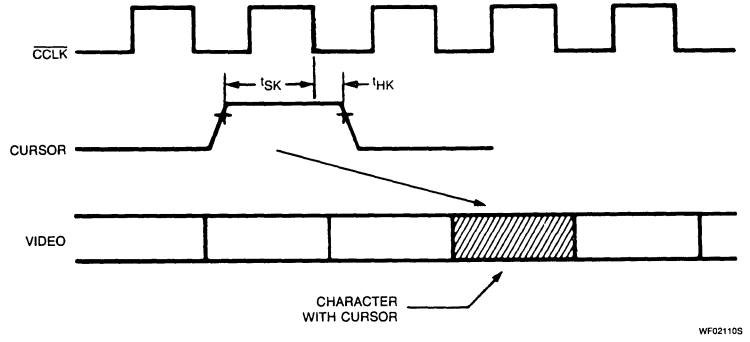


Figure 7. Cursor Pipeline Timing

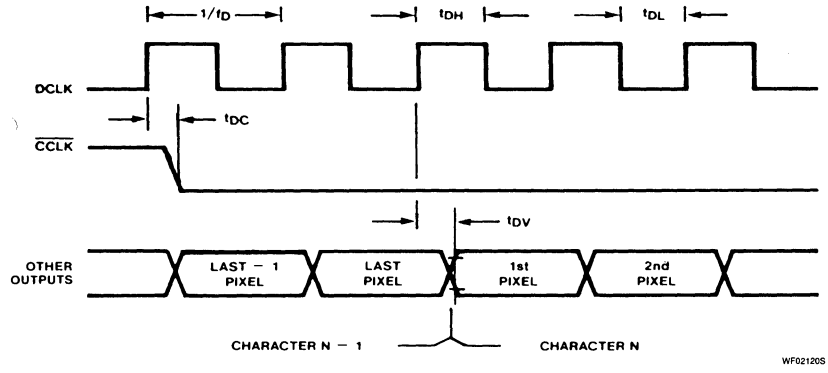


Figure 8. Output Pipeline Timing

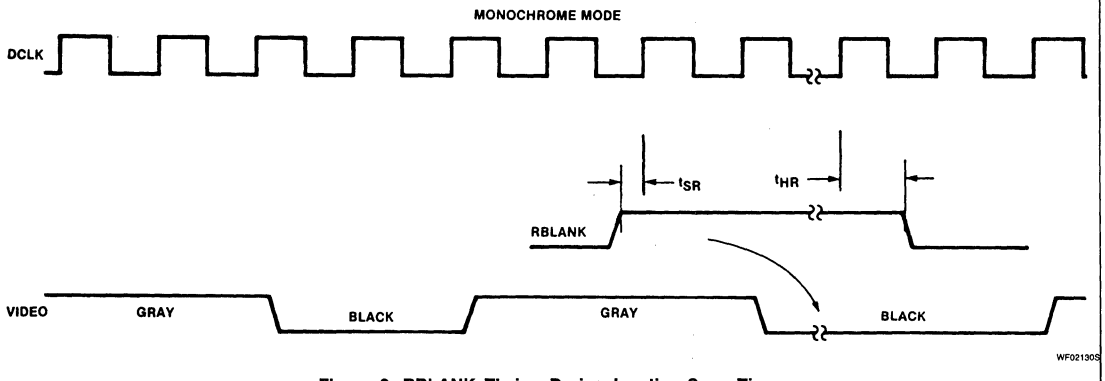
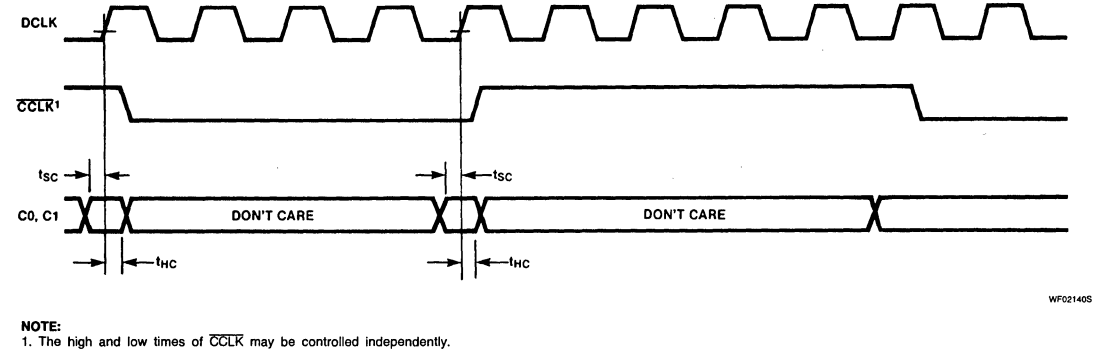


Figure 9. RBLANK Timing During Inactive Scan Time (Blank = 1)



NOTE:
1. The high and low times of $\overline{CCLK}$ may be controlled independently.

Figure 10. Clock Divider Timing

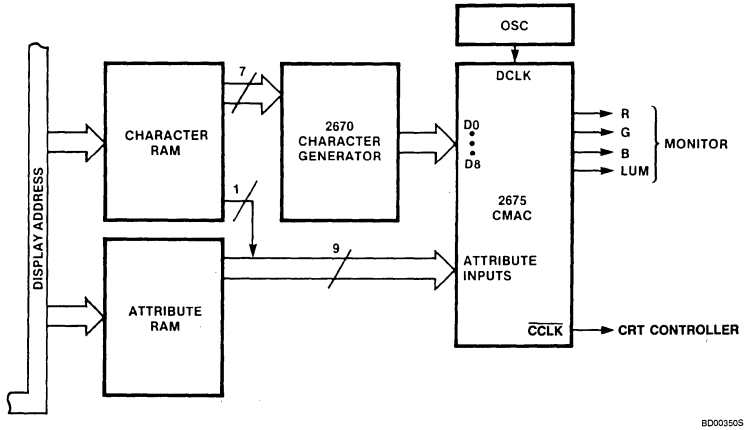


Figure 11. System Block Diagram of SCB2675 in Color Mode

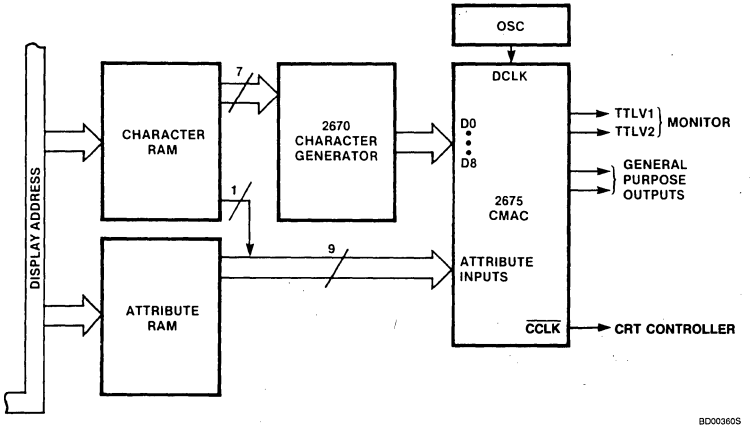


Figure 12. System Block Diagram of SCB2675 in Monochrome Mode

Signetics

SCB2675T

Turbo Color/Monochrome Attributes Controller (Turbo-CMAC)

Microprocessor Products

Preliminary Specification

DESCRIPTION

HThe Signetics SCB2675T Turbo Color/ Monochrome Attributes Controller (Turbo-CMAC) is a bipolar LSI device designed for CRT terminals and display systems that employ raster scan techniques. It contains a programmable dot clock divider to generate a character clock, a high speed shift register to serialize input dot data into a video stream, latches and logic to apply visual attributes to the resulting display, and logic to display a cursor on the display.

The Turbo-CMAC provides control of visual attributes on a character by character basis for two operating modes: monochrome and color. The monochrome mode provides reverse video, blank, highlight and two general purpose user definable attributes. In this mode, the display characters can be specified to appear on either a light or dark screen background. Retrace video suppression can be automatically or externally controlled. The color mode provides eight colors for foreground (character) video and eight colors for background video together with a luminance output for external color set selection or to simultaneously drive a monochrome monitor. Additionally, both modes provide double width, underline, blink, dot stretching and dot width attributes. In monochrome mode, the SCB2675T emulates the attribute characteristics of Digital Equipment Corporation's VT100 terminal.

The horizontal dot frequency is the basic timing input to the Turbo-CMAC. This clock is divided internally to provide a character clock output for system synchronization. Up to nine bits of dot data are parallel loaded into the video shift register on each character boundary. The two TTL video data outputs in monochrome mode are encoded to provide four video intensities (black, gray, white and highlight). The video data in color mode is encoded to provide eight foreground colors and shifted out on three TTL outputs, together with the luminance output.

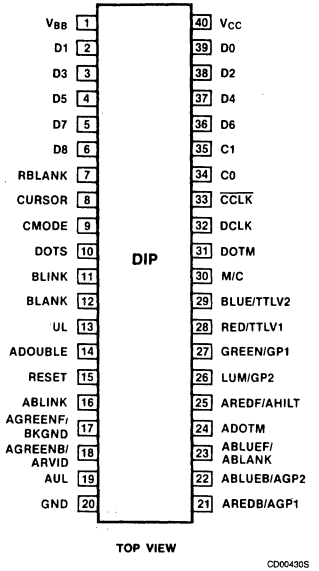
FEATURES

- **40MHz video dot rate version**
- **Four video intensities encoded on two TTL outputs (monochrome mode)**
- **Eight foreground and background colors encoded on three TTL outputs (color mode)**
- **Internally latched character attributes:**
  - **Reverse video**
  - **Blank**
  - **Blink**
  - **Underline**
  - **Highlight**
  - **Two general purpose**
  - **Eight foreground colors**
  - **Eight background colors**
  - **Dot width control**
  - **Double width characters**
- **VT100 compatible attributes**
- **Reverse video cursor with optional white cursor in color mode**
- **Up to 10 dots per character**
- **Light or dark background in monochrome mode**
  - **Automatic retrace blanking**
- **Programmable dot stretching**
- **Compatible with SCN2674 AVDC and SCN2670 DCGG**
- **TTL compatible**
- **40-pin dual in-line package**

APPLICATIONS

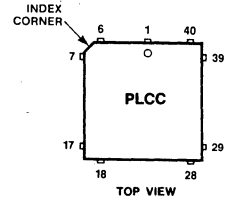
- **CRT terminals**
- **Word processing systems**
- **Small business computers**

PIN CONFIGURATION



TOP VIEW

CD00430S



TOP VIEW

CD0044PS

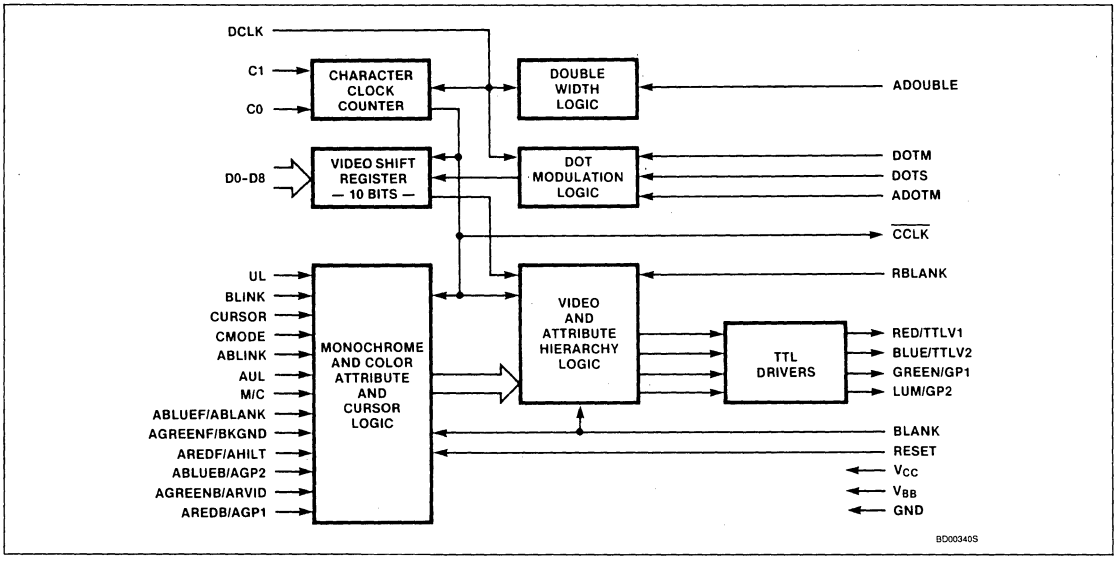
Pin	Function	Pin	Function
2	V_{BB}	24	AREDB/AGP1
3	D1	25	ABLUEB/AGP2
4	D3	26	ABLUEF/ABLANK
5	D5	27	ADOTM
6	D7	28	AREDF/AHILT
7	D8	29	LUM/GP2
8	RBLANK	30	GREEN/GP1
9	CURSOR	31	RED/TTLV1
10	CMODE	32	BLUE/TTLV2
11	DOTS	33	M/C
13	BLINK	35	DOTM
14	BLANK	36	DCLK
15	UL	37	$\overline{CCLK}$
16	ADOUBLE	38	C0
17	RESET	39	C1
18	ABLINK	40	D6
19	AGREENF/BKGND	41	D4
20	AGREENB/ARVID	42	D2
21	AUL	43	D0
22	GND	44	V_{CC}

Turbo Color/Monochrome Attributes Controller (Turbo-CMAC) SCB2675T

ORDERING CODE

PACKAGES	DOTS PER CHARACTER	V_{CC} = 5V ± 5%, 0°C to +70°C
		40MHz
Plastic DIP Plastic LCC	7, 8, 9, 10	SCB2675TC4N40 SCB2675TC4A44

BLOCK DIAGRAM



Turbo Color/Monochrome Attributes Controller (Turbo-CMAC) SCB2675T

PIN DESCRIPTION

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
V_{CC}	40	44	I	**Power Supply:** +5VDC
V_{BB}	1	2	I	**Bias Supply:** See figure 5
GND	20	22	I	**Ground:** 0V reference
DCLK	32	36	I	**Dot Clock:** Dot frequency input. Video output shift rate.
$\overline{CCLK}$	33	37	O	**Character Clock:** An output which is a submultiple of DCLK. The period ranges from 7 to 10 DCLK periods per cycle and is determined by the state of the C0-C1 inputs.
RED/TTLV1	28	31	O	**Red/TTL Video 1:** In color mode, this output provides the red gun serial video. In monochrome mode, it should be used with the blue/TTL video 2 output to decode four video intensities.
BLUE/TTLV2	29	32	O	**Blue/TTL Video 2:** In color mode, this output provides the blue gun serial video. In monochrome mode, it should be used with the red/TTL video 1 output to decode four video intensities.
GREEN/GP1	27	30	O	**Green/General Purpose 1:** In color mode, this output provides the green gun serial video. In monochrome mode, it is a general purpose TTL output which is asserted if the AREDB/AGP1 input is asserted when the corresponding character dot data is loaded into the video shift register. GP1 can be active in either active scan or blank time.
LUM/GP2	26	29	O	**Luminance/General Purpose 2:** In color mode, this output is the logical-OR of the RGB foreground video. It is low during a blanking interval and during the foreground portion of the cursor display. In monochrome mode, it is a general purpose TTL output which is asserted if the ABLUEB/AGP2 input is asserted when the corresponding character dot data is loaded into the video shift register. GP2 can be active in either active scan or blank time.
UL	13	15	I	**Underline Timing:** Indicates the scan line(s) for the underline attribute. Latched on the falling edge of BLANK. The underline will be displayed on the specified scan line for every character where AUL = 1.
BLINK	11	13	I	**Blink Timing:** This input is sampled on the falling edge of BLANK to provide the blink rate for the blink attribute. Should be a submultiple of the frame rate.
BLANK	12	14	I	**Screen Blank:** When high, this input forces the video outputs to the specified background color in color mode and to the level specified by the BKGND input (either black or gray) in monochrome mode.
RBLANK	7	8	I	**Retrace Blank:** This input is used to force the video outputs to a low during retrace periods. If pulled high, it will automatically suppress video during the retrace periods when BLANK is high. The user may also pulse this input while BLANK is high to selectively suppress raster video.
AGREENF/ BKGND	17	19	I	**Green Foreground/Background Intensity:** In color mode, this input activates the GREEN/ GP1 output during the foreground (character video) portion of the associated character block. In monochrome mode, this input specifies gray or black screen background.
ABLUEF/ ABLANK	23	26	I	**Blue Foreground/Blank Attribute:** In color mode, this input activates the BLUE/TTLV2 output during the foreground (character video) portion of the associated character block. In monochrome mode, this input generates a blank space for the associated character. The blank space intensity is controlled by the AGREENF/BKGND input, the reverse video attribute and cursor input.
AREDF/AHILT	25	28	I	**Red Foreground/Highlight Attribute:** In color mode, this input activates the RED/ TTLV1output during the foreground (character video) portion of the associated character block. In monochrome mode, this input highlights the associated character (including underline).
CURSOR	8	9	I	**Cursor Timing:** This input provides the timing for the cursor video. In color mode, with CURSOR and CMODE high, the RGB outputs are driven high (white cursor). If CMODE is low, or in monochrome mode, this input reverses the intensities of the video and attributes (the foreground and background intensities are reversed). Cursor position, shape, and blink rate are controlled by this input.
CMODE	9	10	I	**Cursor Mode:** Used in color mode only. When CURSOR and CMODE are high, the RGB outputs are driven high (white cursor). When CURSOR is high and CMODE is low, the RGB outputs are logically inverted (reverse video cursor).
AUL	19	21	I	**Underline Attribute:** Specifies a line to be displayed in the character block. The specific line(s) are specified by the UL input. All other attributes apply to the underline video.

Turbo Color/Monochrome Attributes Controller (Turbo-CMAC) SCB2675T

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
ABLINK	16	18	I	**Blink Attribute:** In color mode, this active high input will drive the foreground RGB combination to the background RGB combination. In monochrome mode, the associated character or background is driven to the intensity determined by BKGND, reverse video attribute and the cursor input.
ADOUBLE	14	16	I	**Double Width Attribute:** This active high input causes the associated character video to be shifted out of the serial shift register at one half the dot frequency (DCLK). The $\overline{CCLK}$ output is not affected.
AREDB/AGP1	21	24	I	**Red Background/General Purpose Attribute 1:** In color mode, this input activates the RED/TTLV1 output during the background portion of the associated character block. In monochrome mode, it activates the GREEN/GP1 output for the associated character block.
ABLUEB/AGP2	22	25	I	**Blue Background/General Purpose Attribute 2:** In color mode, this input activates the BLUE/TTLV2 output during the background portion of the associated character block. In monochrome mode, it activates the LUM/GP2 output for the associated character block.
AGREENB/ ARVID	18	20	I	**Green Background/Reverse Video Attribute:** In color mode, this input activates the GREEN/GP1 output during the background portion of the associated character block. In monochrome mode, it causes the associated character block video intensities to be reversed.
D0 – D8	39, 2, 38, 3, 37, 4, 36, 5, 6	43, 3, 42, 4, 41, 5, 40, 6, 7	I	**Dot Data Input:** These are parallel inputs corresponding to the character/graphic symbol dot data for a given scan line. These inputs are strobed into the video shift register on the trailing (falling) edge of each character clock ($\overline{CCLK}$).
C0, C1	34, 35	38, 39	I	**Character Clock Control:** The states of these two static inputs determine the internal divide factor for the $\overline{CCLK}$ output rate.
RESET	15	17	I	**Reset:** This active high input initializes the internal logic and resets the attribute latches.
M/C	30	33	I	**Monochrome/Color Mode:** This input selects whether the CMAC operates in monochrome or color mode. A low selects color mode and a high selects monochrome mode.
ADOTM	24	27	I	**Dot Modulation Attribute:** When DOTM and this input are high, the active dot width of the associated character video is one DCLK. When DOTM is high and this input is low, the active dot width of the associated character video is two DCLKs.
DOTM	31	35	I	**Dot Width Modulation:** When this input is high, two DCLKs are used for each dot shifted through the shift register. When this input is low, one DCLK is used.
DOTS	10	11	I	**Dot Stretching:** Sampled at the falling edge of BLANK. When this input is high, one extra dot is appended to individual dots or groups of dots of the input parallel data and then transferred through the shift register. When this input is low, normal transfer of input parallel data results.

FUNCTIONAL DESCRIPTION

The Turbo-CMAC consists of seven major sections (see block diagram). The high speed dot clock input is applied to a programmable divider to provide a character clock output for system timing. Parallel dot data is loaded into the video shift register on character boundaries and shifted into the video logic block at the dot rate specified by the dot modulation section. The appropriate attribute control inputs are selected by the mode select logic, latched internally on character boundaries, and combined with the serial dot data to provide monochrome or color video outputs.

The BLANK input defines the active screen and retrace areas. In color mode, the video outputs are forced to the specified background color when this signal is asserted; in monochrome mode the video outputs are forced to the states defined by the BKGND input, i.e., black if dark background is selected and gray if light background is selected. A separate RBLANK input allows the user to select the amount of border around the active area when operating in color mode or in monochrome mode with light background. This input can be tied high, in which case the area outside the active area will be dark, or it may be pulsed during BLANK periods to externally control the border widths.

In color mode, eight colors for the character (foreground) and eight colors for the background (area other than character) can be selected by the attribute inputs. In monochrome mode, the intensities of foreground and background are a function of the attribute and BKGND inputs, i.e., characters may be black, gray, white, or highlight (very white) while background may be black, gray, or white (see table 1).

Character Clock Counter

The character clock counter divides the DCLK input to generate the character clock ($\overline{CCLK}$). The divide factor is specified by the clock control inputs (C1 – C0) as follows:

C1	C0	SCB2675T Dots/ Char.	SCB2675T $\overline{CCLK}$ Duty Cycle*
0	0	10	5/5
0	1	7	3/4
1	0	8	4/4
1	1	9	4/5

*High/low

The number of dot clocks/character is normally the number of dots/character as listed above. However, when dot width control is specified, the DCLK input is divided by two before it is applied to the character clock counter resulting in the number of dot clocks/character being double those listed above, although the number of displayed dots/character remains the same. See Dot Modulation Logic section. The number of dots per character (C1 – C0) can be dynamically changed on either edge of the character clock as shown in figure 10.

Table 1. MONOCHROME MODE ATTRIBUTE CHARACTERISTICS

REV[1]	AHILT	ABLINK[2]	FOREGROUND VIDEO	BACKGROUND VIDEO
0	0	0	W	B
0	0	1	W/G	B
0	1	0	H	B
0	1	1	H/W	B
1	0	0	B	G
1	0	1	B/W	G/B
1	1	0	B	W
1	1	1	B/H	W/B

NOTES:

1. REV = (BKGND) XOR (ARVID):

BKGND	ARVID	REV
0	0	0
0	1	1
1	0	1
1	1	0

2. For blinking, the video outputs are shown as 0/1, where 0 and 1 are the blink timing output states.
3. Foreground includes underline when underlining is specified by AUL = 1.
4. When ABLANK = 1, foreground component becomes same as background component.
5. Codes for video outputs are as follows:

CODE	TTLV2	TTLV1	BEAM INTENSITY
B	0	0	Black (B)
G	0	1	Gray (G)
W	1	0	White (W)
H	1	1	Highlight (H)

Video Shift Register

On each character boundary, the parallel input dot data (D0 – D8) is loaded into the video shift register. The data is shifted out least significant bit first (D0) at the DCLK rate. If 10 dots/character are specified (C1 – C0 = 00), the tenth dot will be the same as D8. The serial dot data from the video shift register is routed to the video logic where it is combined with the cursor and attribute control bits to produce the video data outputs.

Mode Select, Attribute and Cursor Control

The mode select logic multiplexes the monochrome and color attribute inputs and outputs as specified by the M/C input. The monochrome mode provides blank, reverse video, highlight and two general purpose attributes. The latter may be used, with external logic, to combine other attributes (e.g., overscore) into the video stream. The color mode provides RGB foreground and background color attributes. Both modes provide double width characters, blink, underline, dot width control and dot stretching.

The cursor and attribute inputs are pipelined internally to allow for system pipeline propagations. The cursor input and the attribute inputs are delayed for one $\overline{CCLK}$ to account for the delay of the character data through the character generator latches. The attribute timing inputs (BLINK, UL and DOTS) are clocked into the 2675 at the beginning of each scan line time by the falling edge of BLANK. Thus, these inputs must be in their proper state at the falling edge of BLANK preceding the scan line where they are required to be active. The BLANK signal itself is also delayed internally to provide for the RAM and character generator delays (see figures 6 and 7). Internal delays cause the video outputs to be delayed relative to $\overline{CCLK}$ as illustrated in figure 8.

Video Logic

Each character block consists of the three components shown in figure 1. Symbol video is generated from the dot data inputs D0 – D8. Underline video is enabled by the AUL attribute and is generated during the scan lines for which the UL input is active. Underline and symbol video are always the same intensity or color, and other attributes (e.g., ABLINK) apply to them equally. The combination of underline and symbol video is also referred to as foreground video. Background video is the area of the character block corresponding to the absence of foreground video. The assertion of the non-display attribute (ABLANK) in the monochrome mode causes the entire character block to be displayed as background.

In monochrome mode, the serial dot data and pipelined cursor and attributes are combined to generate four video intensities (black, gray, white and highlight) which are encoded on the TTLV1 and TTLV2 outputs as follows:

TTLV2	TTLV1	VIDEO INTENSITY
0	0	Black
0	1	Gray
1	0	White
1	1	Highlight

Table 1 describes the relationship between attributes and video intensity of the foreground and background components of the character block in monochrome mode.

In color mode, the colors of the foreground and background components are specified by the corresponding attribute inputs; AREDF, AGREENF and ABLUEF dictate the color of the foreground component while AREDB, AGREENB and ABLUEB do the same for the background component. In this mode, the serial dot data and pipelined cursor and attributes are combined to generate four video outputs. The RED, GREEN and BLUE outputs separately contain the corresponding foreground and background components. The LUM output is the logical-OR of the foreground colors and can be used to drive a separate monochrome monitor or to select a different set of colors for the foreground.

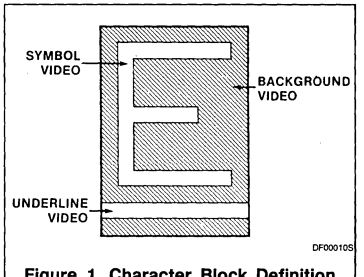


Figure 1. Character Block Definition

Dot Modulation Logic

The dot modulation logic controls the video shift register to supply dot stretching and dot width control.

Dot stretching is controlled by the DOTS input which is sampled each scan line at the trailing (falling) edge of BLANK. If DOTS is asserted at that time, all characters on the following scan line will have dot stretching applied. Dot stretching causes an extra dot to be added to individual dots or groups of dots as shown in figures 2 and 3. Dot stretching can be used to:

1. Compensate for low video bandwidth monitors (since the minimum active displayed segment with dot stretching is two DCLKs).
2. Assure crisp black characters when operating in white background mode.
3. Provide thick characters as a means of distinguishing areas of the display.

Dot width is controlled by the DOTM and ADOTM inputs. DOTM is tied either high, which enables the feature on the entire display, or low, which disables the feature. With ADOTM high, the dot width of characters can be selectively controlled by assertion of the ADOTM attribute input. When operating in this mode, the dot clock input is divided by two before being applied to other circuits in the Turbo-CMAC.

The $\overline{CCLK}$ output is also divided by two. The truth table for the possible combinations of ADOTM and DOTM is as follows:

ADOTM	DOTM	OPERATION
0	0	Normal mode
0	1	Dot width control, 100% duty cycle (DCLK÷2, $\overline{CCLK}$÷2)
1	0	Not allowed
1	1	Dot width control, 50% duty cycle (DCLK÷2, $\overline{CCLK}$÷2)

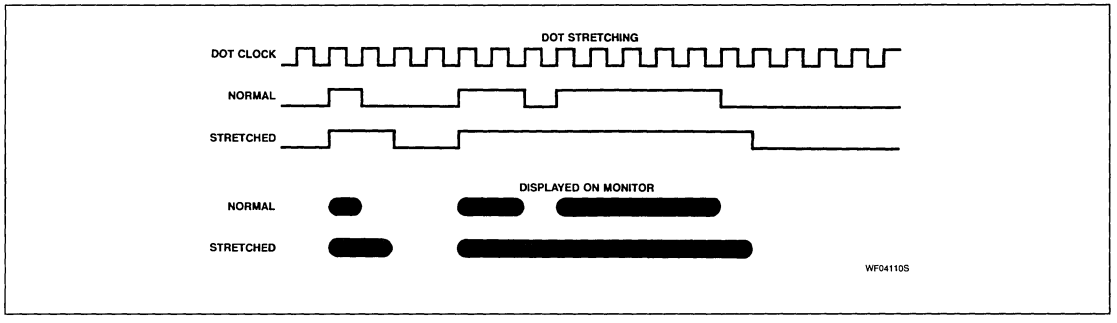
When dot width control is enabled as above, two DCLKs are used for each video dot period. Asserting ADOTM for a particular character will cause each active video dot of the displayed character to be turned on for one DCLK and off for the other DCLK, while if ADOTM is negated for that character, the active video dot for that character will be turned on (black background) or off (white background) for both DCLK times (see figures 2 and 4). Only the character video component of the character block is modulated. Underline video and background are not affected by on-time modulation. Width control can be used to:

1. Make horizontal lines and vertical lines appear the same brightness on the display.
2. Provide two different brightness levels for characters without requiring a monitor with analog brightness inputs.

However, note that the effects produced by this feature are highly dependent on the video amplifier characteristics of the monitor used.

Double Width Logic

The double width logic controls the rate at which dots are shifted through the video shift register. When the ADOUBLE input is asserted, the associated character video will be shifted at one half the DCLK rate, and the dot information for the next character will be loaded into the shift register two $\overline{CCLK}$s later. The character and attribute data that is present on the first $\overline{CCLK}$ after the ADOUBLE input is asserted will be ignored. The $\overline{CCLK}$ output is not affected. If a double width character is specified at the last location of a character row, the second half of the double width character (one $\overline{CCLK}$) will extend into the horizontal front porch.



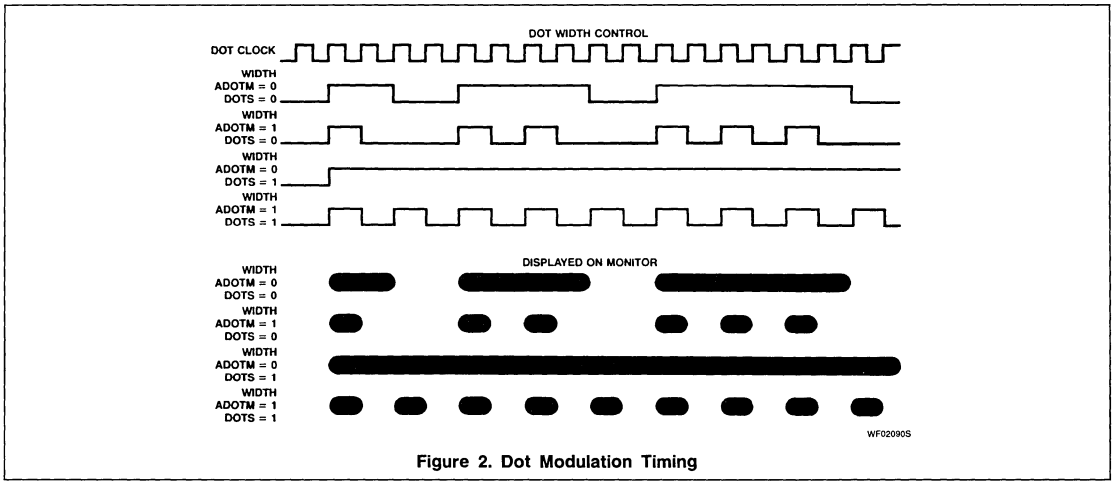


Figure 2. Dot Modulation Timing

Figure 3. Dot Stretching

Figure 4. Dot Width Control

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS T_A = 0°C to +70°C, V_{CC} = 5V ±5%, V_{BB} = figure 5[4,5]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		2.0			V
V_{OL}	Output low voltage	I_{OL} = 4mA			0.4	V
V_{OH}	Output high voltage	I_{OH} = −400μA	2.4			V
I_{IL}	Input low current	V_{IN} = 0.4V				
	DCLK				−800	μA
	All other inputs				−400	μA
I_{IH}	Input high current	V_{IN} = 2.4V				
	DCLK				40	μA
	All other inputs				20	μA
I_{CC}	V_{CC} supply current	V_{IN} = 0V, V_{CC} = max			30	mA
I_{BB}	V_{BB} supply current	Figure 5			110	mA

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = figure 5[4,5]

PARAMETER		TEST CONDITIONS	TENTATIVE LIMITS 40MHz Version Min	TENTATIVE LIMITS 40MHz Version Max	UNIT
Dot clock timing (figure 8)					
f_D	Frequency			40	MHz
t_{DH}	High time		10		ns
t_{DL}	Low time		10		ns
Set-up times (figures 6, 7, 9 and 10)					
t_{SB}	BLANK to $\overline{CCLK}$		35		ns
t_{SA}	Attributes to $\overline{CCLK}$		40		ns
t_{SD}	D0–D9 to $\overline{CCLK}$		50		ns
t_{SK}	CURSOR to $\overline{CCLK}$		35		ns
t_{SC}	C0, C1 to DCLK		15		ns
t_{SR}	RBLANK to DCLK		15		ns
t_{SM}	BLINK, UL, DOTS to BLANK		10		ns
Hold times (figures 6, 7, 9 and 10)					
t_{HB}	BLANK from $\overline{CCLK}$		5		ns
t_{HA}	Attributes from $\overline{CCLK}$		5		ns
t_{HD}	D0-D8 from $\overline{CCLK}$		5		ns
t_{HK}	CURSOR from $\overline{CCLK}$		5		ns
t_{HC}	C0, C1 from DCLK		5		ns
t_{HR}	RBLANK from DCLK		10		ns
t_{HM}	BLINK, UL, DOTS from BLANK		15		ns
Delay times (figure 8)		$C_L = 50pF$			
t_{DC}	$\overline{CCLK}$ from DCLK			45	ns
t_{DV}	Other outputs from DCLK			45	ns

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying voltages greater than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground. For testing, all input signals swing between 0.4V and 2.4V with a transition time of 3ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages are checked at 0.8V and 2.0V.

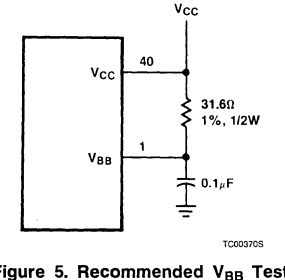
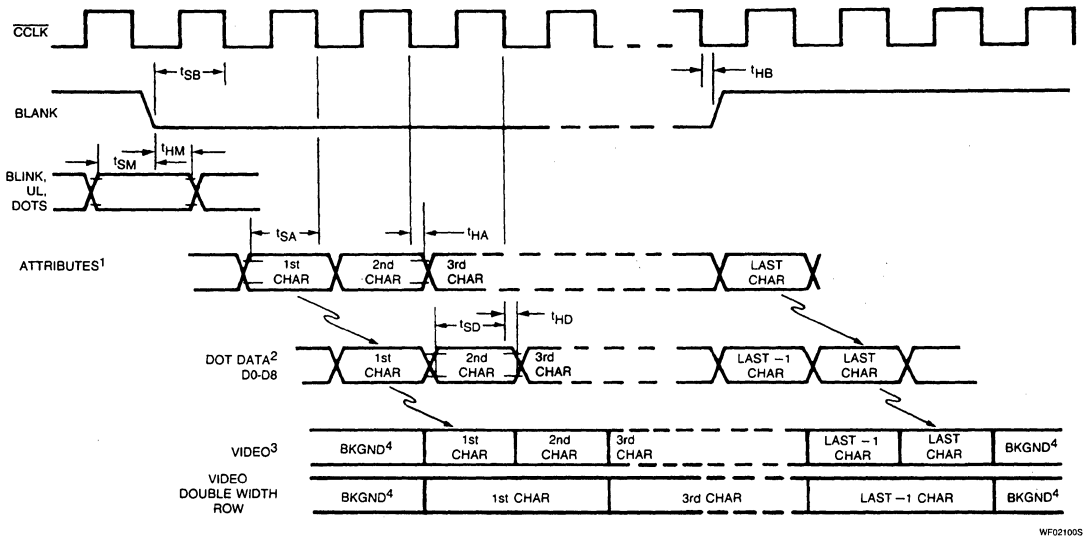


Figure 5. Recommended V_{BB} Test Circuit



NOTES:
1. Attributes include ABLINK, ABLANK, ARVID, AUL, AHILT, ADOUBLE, ADOTM, two general purpose, and foreground/background colors.
2. One $\overline{CCLK}$ delay for dot data (obtained from delay through character generator).
3. See figure 8 for detail timing of video outputs.
4. Non-active scan time. VIDEO reverts to polarity selected by the BKGND input in monochrome mode.

Figure 6. Turbo-CMAC Pipeline Timing

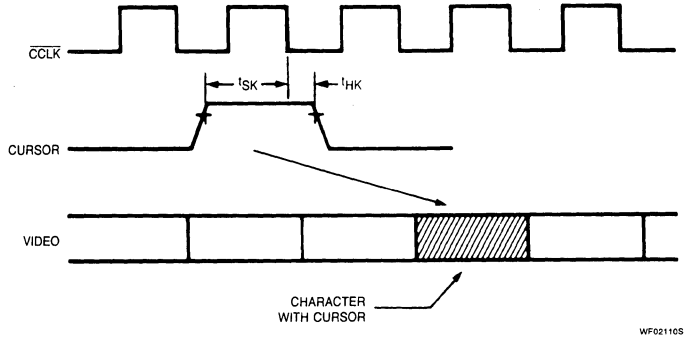


Figure 7. Cursor Pipeline Timing

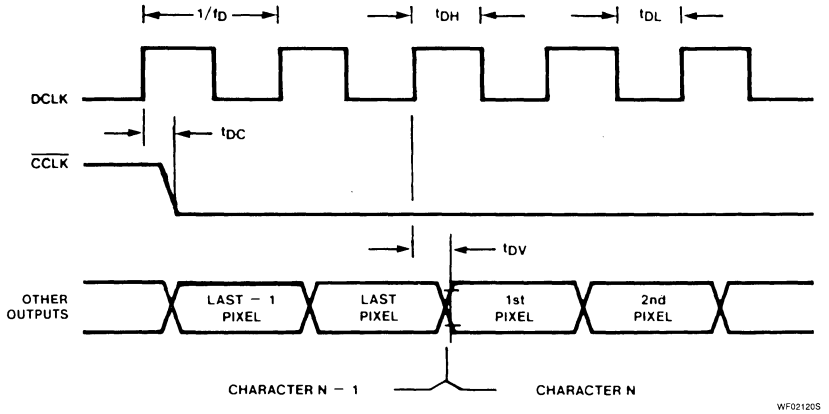


Figure 8. Output Pipeline Timing

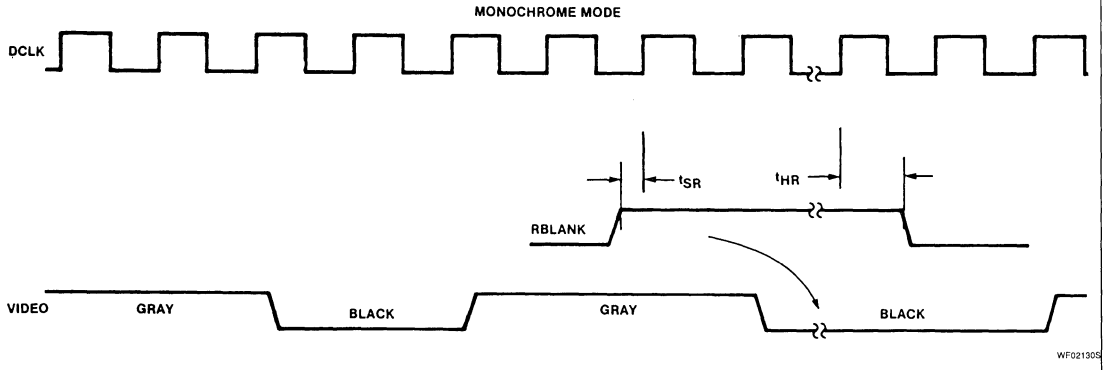
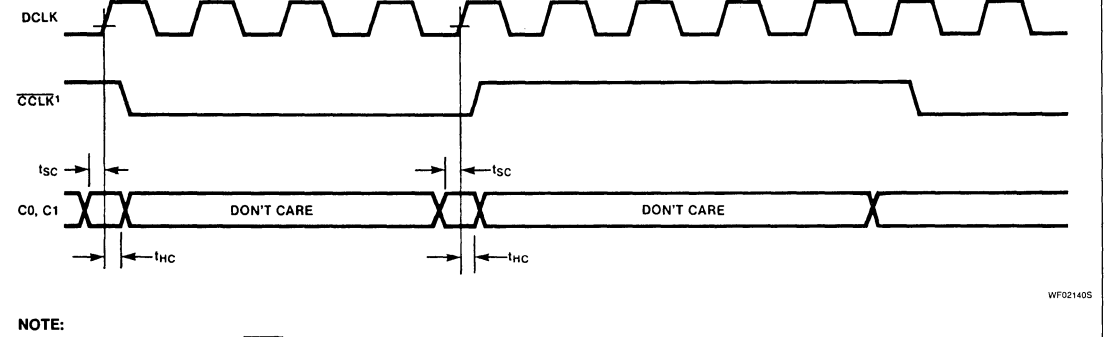


Figure 9. RBLANK Timing During Inactive Scan Time (Blank = 1)



NOTE:
1. The high and low times of $\overline{CCLK}$ may be controlled independently.

Figure 10. Clock Divider Timing

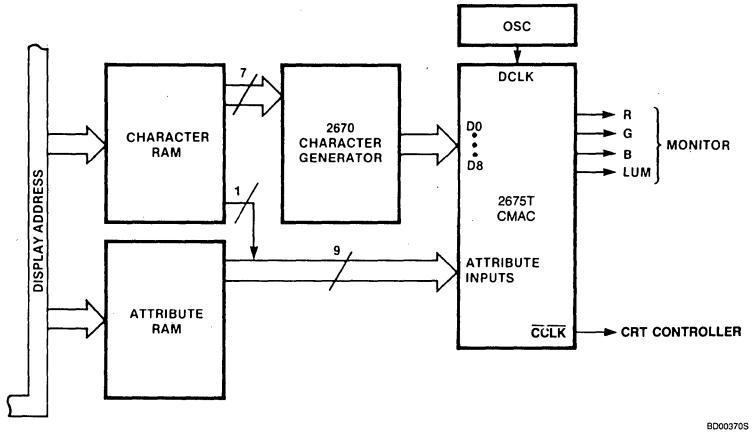


Figure 11. System Block Diagram of SCB2675T in Color Mode

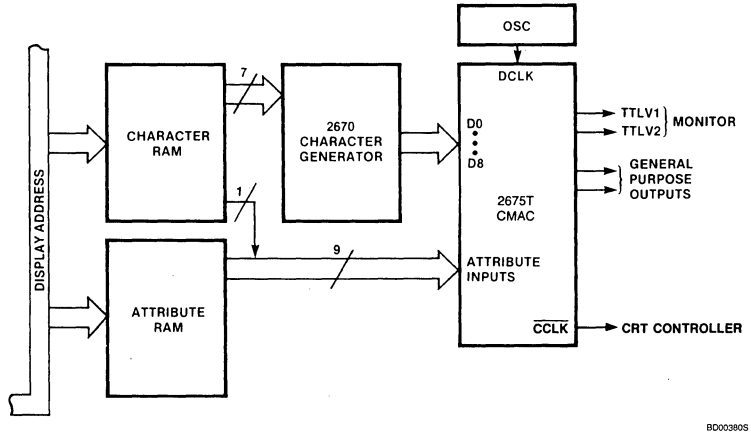


Figure 12. System Block Diagram of SCB2675T in Monochrome Mode

SCB2677
Video Attributes Controller (VAC)

Product Specification

Microprocessor Products

DESCRIPTION

The Signetics SCB2677A and SCB-2677B Video Attributes Controllers (VAC) are bipolar LSI devices designed for CRT terminals and display systems that employ raster scan techniques. Each contains a high speed video shift register, field and character attributes logic, attribute latch, cursor format logic and half dot shift control.

The VAC provides control of visual attributes on a field or character by character. Internal logic preserves field attribute data from character row to character row so that an attribute byte is not required at the beginning of each row. The SCB2677B provides for reverse video, blank (non-display), blink, underline and highlight attributes and a graphics mode attribute to work in conjunction with the Signetics SCN2670 Display Character and Graphics Generators (DCGG). The SCB2677A substitutes a strike-thru attribute for the graphics attribute.

The horizontal dot frequency is the basic timing input to the VAC. Internally, this clock is divided down to provide a character clock output for system synchronization. Up to ten bits of video dot data are parallel loaded into the video shift register on each character boundary. The video data is encoded to three levels of intensity (black, gray and white) and output on two TTL outputs. Light or dark screen background may be specified.

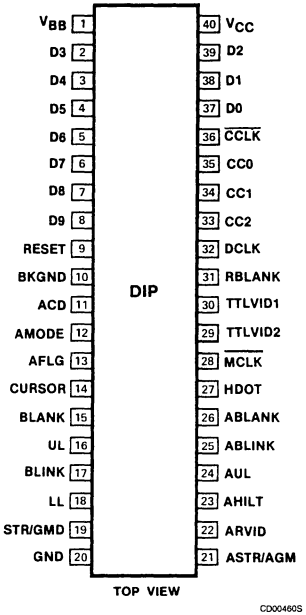
FEATURES

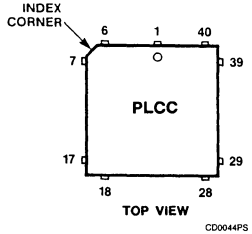
- **18MHz and 25MHz video dot rate**
- **Three level encoded TTL video outputs**
- **Character/field attribute logic**
  - **Reverse video**
  - **Character blank**
  - **Character blink**
  - **Underline**
  - **Highlight**
- **Strike-thru or graphics control**
- **Field attributes extend from row to row**
- **Light or dark field**
- **Cursor reverse video logic**
- **Up to 12 dots per character**
- **Retrace blanking for light field**
- **Optional field graphics control output**
- **High speed bipolar design**
- **TTL compatible**
- **Compatible with Signetics SCN2672 PVTC, SCN2674 AVDC and SCN2670 DCGG**
- **Upgrade of the Signetics SCB2673 VAC**

APPLICATIONS

- **CRT terminals**
- **Word processing systems**
- **Small business computers**

PIN CONFIGURATION





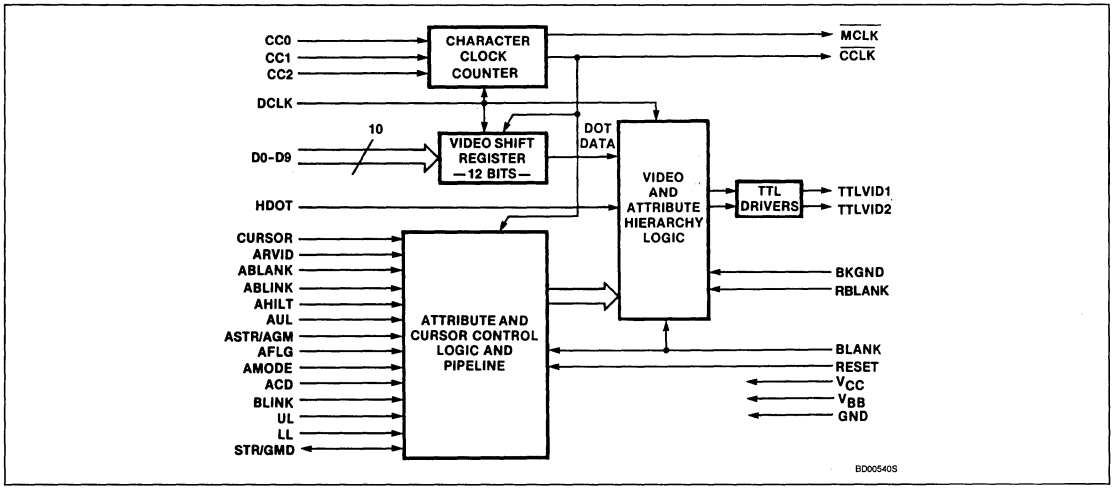
Pin	Function	Pin	Function	Pin	Function
1	NC	16	CURSOR	30	HDOT
2	V_{BB}	17	BLANK	31	MCLK
3	D3	18	UL	32	TTLVID2
4	D4	19	BLINK	33	TTLVID1
5	D5	20	LL	34	NC
6	D6	21	STR/GMD	35	RBLANK
7	D7	22	GND	36	DCLK
8	D8	23	NC	37	CC2
9	D9	24	ASTR/AGM	38	CC1
10	RESET	25	ARVID	39	CC0
11	BKGND	26	AHILT	40	CCLK
12	NC	27	AUL	41	D0
13	ACD	28	ABLINK	42	D1
14	AMODE	29	ABLANK	43	D2
15	AFLG			44	V_{CC}

Video Attributes Controller (VAC) SCB2677

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C			
	Graphics Attribute		Strike-Thru Attribute	
	25MHz	18MHz	25MHz	18MHz
Ceramic DIP	SCB2677BC5I40	SCB2677BC8I40	SCB2677AC5I40	SCB2677AC8I40
Plastic DIP	SCB2677BC5N40	SCB2677BC8N40	SCB2677AC5N40	SCB2677AC8N40
Plastic LCC	SCB2677BC5A44	SCB2677BC8A44	SCB2677AC5A44	SCB2677AC8A44

BLOCK DIAGRAM



Video Attributes Controller (VAC) SCB2677

PIN DESCRIPTION

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
DCLK	32	36	I	**Dot Clock:** Dot frequency input. Video output shift rate.
$\overline{CCLK}$	36	40	O	**Character Clock:** A submultiple of DCLK. The frequency ranges from one sixth to one twelfth of DCLK, as determined by the state of the CC0–CC2 inputs.
CC2 – CC0	33 – 35	37 – 39	I	**Character Clock Control:** The logic state on these three static inputs determine the internal divide factor for the $\overline{CCLK}$ output rate. Character clock rates of 6 through 12 dots per character may be specified.
D0 – D9	37 – 39, 2 – 8	41 – 43, 3 – 9	I	**Dot Data Input:** These are parallel inputs corresponding to the character/graphic symbol dot data for a given scan line. These inputs are strobed into the video shift register on the falling edge of each character clock.
HDOT	27	30	I	**Half Dot Shift:** When this input is high, the serial video output is delayed by one half dot time. This input is latched on the falling edge of each character clock.
CURSOR	14	16	I	**Cursor Timing:** This input provides the timing for the cursor video. When high, effectively reverses the intensities of the video and attributes. Cursor position, shape, and blink rate are controlled by this input.
BKGND	10	11	I	**Background Intensity:** Specifies light or dark video during BLANK and character fields. Affects the intensities of all attributes.
BLANK	15	17	I	**Screen Blank:** When high, this input forces the video outputs to the level specified by the BKGND input (either high or low intensity). Not effective when RBLANK is high.
RBLANK	31	35	I	**Retrace Blank:** This input is used to force the two video outputs to a low intensity (black) during retrace intervals. If held high (1), it will automatically suppress video when BLANK is high (1). The user may pulse this input while BLANK is high to selectively suppress raster video.
ARVID	22	25	I	**Reverse Video Attribute:** The intensity of the associated character or field video is reversed. All other attributes are effectively reversed.
AHILT	23	26	I	**Highlight Attribute:** All dot video (including underline) of the associated character or field is highlighted with respect to the BKGND input and the reverse video attribute.
ABLANK	26	29	I	**Blank Attribute:** Generates a blank space in the associated character or field. The blank space intensity is determined by the BKGND input, the reverse video attribute, and the CURSOR input.
ABLINK	25	28	I	**Blink Attribute:** The associated character or field video is driven to the intensity determined by BKGND and the reverse video attribute when the BLINK input is high.
AUL	24	27	I	**Underline Attribute:** Specifies a line to be displayed on the character or field. The line is specified by the UL input. All other attributes apply to the underline video.
ASTR/AGM	21	24	I	**Strike-Thru Attribute (2677A):** Specifies a line to be displayed on the character or field. The line is specified by the STR input. **Attribute Graphics Mode (2677B):** This input is latched and synchronized to provide a field GMD output for the SCN2670 DCGG.
AMODE	12	14	I	**Attribute Mode:** Specifies character (AMODE = 0) or field (AMODE = 1) attribute mode.
AFLG	13	15	I	**Attributes Flag:** The VAC samples and latches the attributes inputs when this input is high. If field attributes are specified (AMODE = 1), the attributes are double buffered on a row basis. Thus, each scan line of every character row will start with the attributes that were valid at the end of the previous row.
ACD	11	13	I	**Attribute Control Display:** In field attributes mode (AMODE = 1), if ACD = 0, the first character in each new attribute field (the attribute control character) will be suppressed and only the attributes will be displayed. If ACD = 1, the first character and the attributes are displayed. This input has no effect in character mode (AMODE = 0).
BLINK	17	19	I	**Blink:** This input is sampled on the falling edge of the BLANK to provide the blink rate for the character blink attribute. It should be a submultiple of the frame rate.
UL	16	18	I	**Underline:** Indicates the scan line(s) for the underline attribute. Latched on the falling edge of BLANK.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
STR/GMD	19	21	I	**Strike-Thru Line (2677A):** Indicates the scan line(s) for the strike-thru attribute. Latched on the falling edge of BLANK.
			O	**Graphics Mode (2677B):** This output provides a synchronized, latched, field graphics mode corresponding to the AGM input. This output can be used to control the GM input on the SCN2670 DCGG.
LL	18	20	I	**Last Line:** Indicates the last scan line of each character row. Used internally to extend field attributes across row boundaries. Latched on the falling edge of BLANK. This input has no effect in character mode (AMODE = 0).
$\overline{MCLK}$	28	31	O	**Memory Clock:** This output is active for the last dot time for each $\overline{CCLK}$ period. See figure 1.
TTLVID1	30	33	O	**TTL Video 1:** This output corresponds to the serial, non-highlighted video dot pattern.
TTLVID2	29	32	O	**TTL Video 2:** This output corresponds to the highlighted serial video dot pattern. Should be used with TTLVID1 to decode a composite video of three intensities.
RESET	9	10	I	**Manual Reset:** This active high input initializes the internal logic and resets the attribute latches.
V_{CC}	40	44	I	**Power Supply:** +5 volts.
V_{BB}	1	2	I	**Bias Supply:** See figure 14.
GND	20	22	I	**Ground:** 0V reference.

FUNCTIONAL DESCRIPTION

The VAC consists of four major sections (see block diagram). The high speed dot clock input is divided internally to provide a character clock for system timing. The parallel dot data is loaded into the video shift register on each character boundary and shifted into the video logic block at the dot rate. The six attribute inputs are latched internally and combined with the serial dot data to provide a three level video source for the monitor.

A separate BLANK input defines the active screen area. When BLANK = 0, the video levels are derived internally by the combinations of dot data, attributes, cursor, and the state of the BKGND input. Either black, gray or white background can be selected. Symbols (dot data) are normally gray and can be highlighted to white or black as shown in figure 2.

During the inactive screen area (BLANK = 1), the video level produced by the TTL outputs is either gray (BKGND = 1) or black (BKGND = 0). A separate retrace blank (RBLANK) input is provided to suppress raster retrace video when gray background is specified. This input will force the video outputs to a low if RBLANK and BLANK = 1. The user may pulse RBLANK during the retrace interval in order to extend the gray border closer to the monitor edges.

Character Clock Counter

The character clock counter divides the frequency on the DCLK input to generate the character clock ($\overline{CCLK}$). The divide factor is specified by the clock control inputs (CC0 – CC2) as shown in the table above. See figure 1.

CC2	CC1	CC0	$\overline{CCLK}$ Dots/Character	$\overline{CCLK}$ Duty Cycle*
0	0	0	6	3/3
0	0	1	6	3/3
0	1	0	7	4/3
0	1	1	8	4/4
1	0	0	9	5/4
1	0	1	10	5/5
1	1	0	11	6/5
1	1	1	12	6/6

*Low/high

Video Shift Register

On each character boundary, the parallel data (D0 – D9) is loaded into the video shift register. The data is shifted out least significant bit first (D0) by the DCLK. If 11 or 12 dots/character are specified (CC2 – CC0 = 110 or 111), a 0 (blank dot) is always shifted out before D0. For 12 dots/character, a 0 is also shifted out after D9. The serial dot data is shifted into the video logic where it is combined with the cursor and attributes to encode three levels of video.

Attribute And Cursor Control

The VAC visual attributes capabilities include: reverse video, character blank, blink, underline, highlight, and strike-thru. The six attributes and the three attribute control inputs (AMODE, AFLG, and ACD) are clocked into the VAC on the falling edge of $\overline{CCLK}$. If AFLG is high, the attributes are latched internally and are effective for either one character time (AMODE = 0) or until another set of attributes is latched (AMODE = 1). The attributes set is double buffered on a row by row basis internally. Using this technique, field attributes can extend across character row boundaries thereby eliminating the necessity of starting each row with an attribute set.

When field attribute mode is selected, (AMODE = 1), the VAC will accommodate two attribute storage configurations. In one configuration, the attribute control data is stored in the refresh RAM, taking the place of the first character code in the field to be affected. For this mode, the ACD input is tied low and blank characters will be displayed in the screen positions occupied by the attribute data (see figure 12). The display RAM contains intermixed character and attribute data. When new attribute data is written to the SCB2677, the AFLG input is set high. The character at that location will be blanked, and only the attribute information will be displayed. That particular attribute data will be

used for the resultant characters until the next AFLG pulse occurs. In the second configuration (ACD = 1), the character codes and attribute data are presented to the VAC in parallel (i.e., there are separate RAMs for the character and attribute data). In this mode, dot data is displayed at each character position (see figure 13).

The CURSOR and the attribute input signals are pipelined internally to allow for system propagations (one $\overline{CCLK}$ for refresh RAM, one $\overline{CCLK}$ for dot generator). The attribute timing signals BLINK, UL, STR and LL are clocked into the VAC at the beginning of each scan line by the falling edge of the BLANK input. Thus, these signals must be in their proper state at the falling edge of BLANK preceding the scan line at which they are to be active (see figure 5). The SCN2670 DCGG delays the character dot data by one character clock. The VAC assumes that there is a DCGG in the system and latches the dot data one character clock later than the latching of the attribute data.

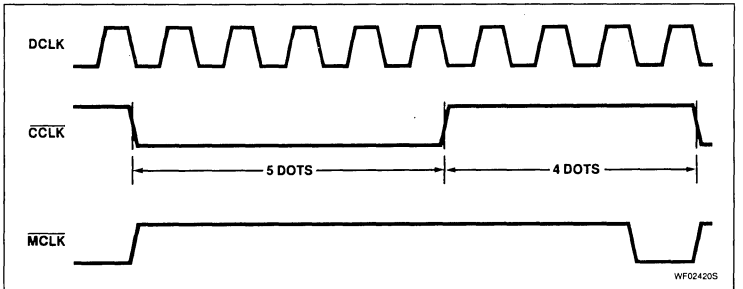


Figure 1. Clock Waveform (9 Dots/Character)

TTLVID2	TTLVID1	INTENSITY
0	0	Black
0	1	Gray
1	0	Not used
1	1	White

NOTE:
The TTLVID1 output can be used independently to generate a two level non-highlighted video.

Video Logic

The serial dot data and the pipelined cursor and attributes are combined to generate three levels (white, gray, and black) on two TTL compatible outputs, TTLVID1 and TTLVID2. The three levels are encoded as shown to the right.

The video is normally shifted out on the leading edge of the DCLK. When the HDOT input is asserted, the corresponding dot data is delayed by one-half DCLK. This half dot shifting, when used on selected lines of character video, can be used to effect eye-pleasing character rounding as shown in figure 3. The half dot shift does not extend into the next character's field boundary.

Attribute Hierarchy

The video of each character block consists of four components as shown in figure 4.

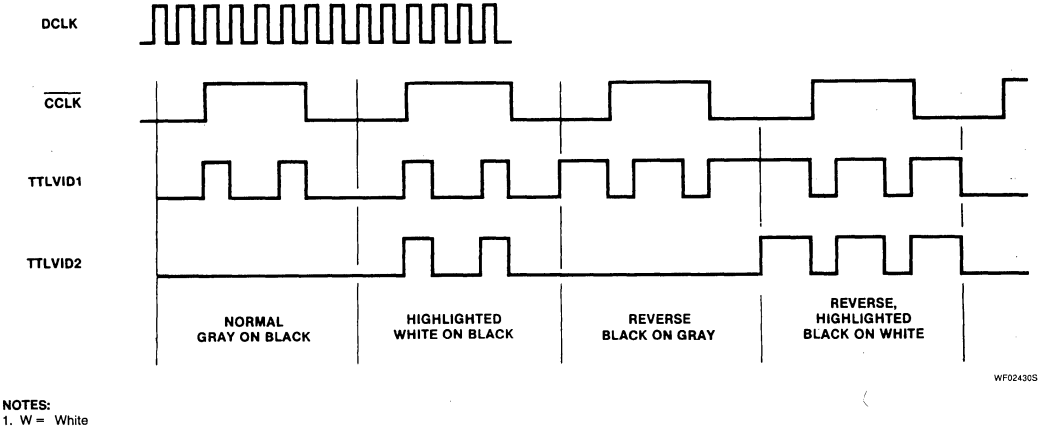
Symbol video is generated from the dot data inputs D0 – D9.

Underline video is enabled by the AUL attribute and is generated when the UL timing input is active. Underline and symbol video are always the same intensity.

Strike-thru video is enabled by the ASTR attribute and is generated when the STR timing input is active. This video is the same intensity as the symbol and underline video. This feature applies to the SCB2677A only.

Surround video is the absence of symbol, underline and strike-thru video or the presence of the non-display attributes (ABLANK or ABLINK • BLINK).

The relative intensities of the four video components are determined by the remaining attributes (AHILT, ABLANK, ABLINK, ARVID) and the BKGND and CURSOR inputs as illustrated in table 1.



NOTES:
1. W = White
   G = Gray
   B = Black

Figure 2. Encoded Video Outputs

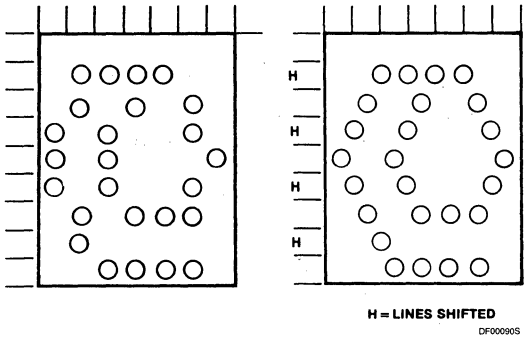


Figure 3. "AT" Symbol with and without Half Dot Shifting

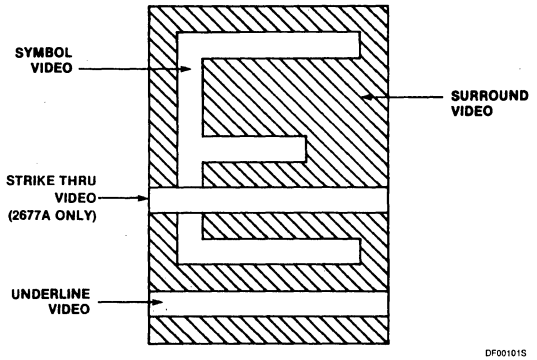


Figure 4. Video Components of Character Block

Table 1. ATTRIBUTES HIERARCHY

BLANK	RBLANK	BKGND	REVERSE[1]	AHILT	"NON-" DISPLAY[2]	SYMBOL, UNDERLINE OR STRIKE-THRU[3,4]	SURROUND VIDEO[3]
0	d	0	0	0	0	G	B
0	d	0	0	0	1	B	B
0	d	0	0	1	0	W	B
0	d	0	0	1	1	B	B
0	d	0	1	0	0	B	G
0	d	0	1	0	1	G	G
0	d	0	1	1	0	B	W
0	d	0	1	1	1	W	W
0	d	1	0	0	0	B	G
0	d	1	0	0	1	G	G
0	d	1	0	1	0	B	W
0	d	1	0	1	1	W	W
0	d	1	1	0	0	G	B
0	d	1	1	0	1	B	B
0	d	1	1	1	0	W	B
0	d	1	1	1	1	B	B
1	0	0	d	d	d	B	B
1	0	1	d	d	d	G	G
1	1	d	d	d	d	B	B

NOTES:
B = Black
G = Gray
W = White
d = Don't care
1. REVERSE = ARVID ⊕ CURSOR
2. Non-display = (ABLINK • BLINK) + ABLANK
3. See figure 4.
4. Symbol, underline and strike-thru are always same intensity.

2

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = See figure 14[3,4,5,6]

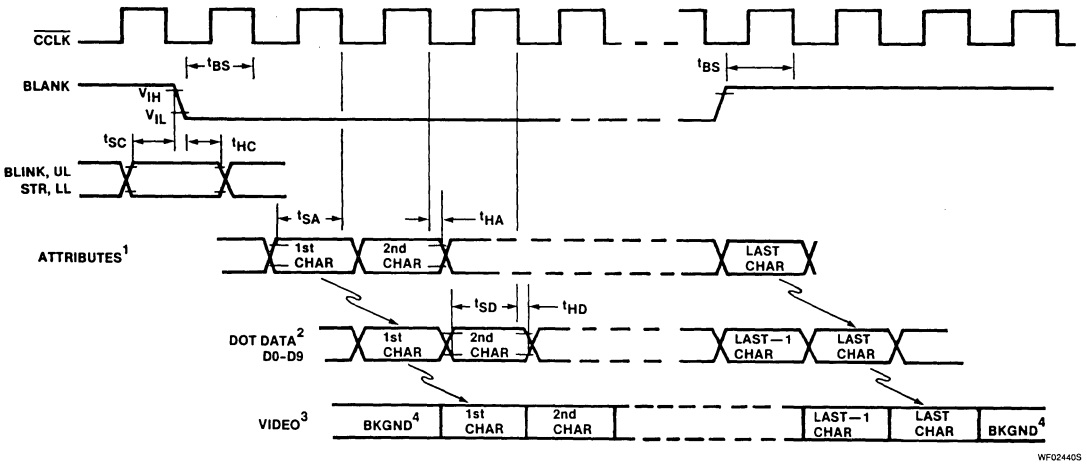
PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		2.0			V
V_{OL}	Output low voltage	$I_{OL} = 4mA$			0.4	V
V_{OH}	Output high voltage	$I_{OH} = -400\mu A$	2.4			V
I_{IL}	Input low current	V_{IN}=0.4V			−400/ −800[6]	μA
I_{IH}	Input high current	V_{IN} = 2.4V			20/40[6]	μA
I_{CC}	V_{CC} supply current	V_{IN} = 0V, V_{CC} = max			80	mA
I_{BB}	V_{BB} supply current	V_{BB} = max			120	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. Operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. Parameters are valid over operating temperature range unless otherwise specified.
4. All voltage measurements are referenced to ground (V_{SS}). All input signals swing between 0.4V and 2.4V. All time measurements are referenced at input voltages of 0.8V, 2.0V and at output voltage of 0.8V, 2.0V as appropriate.
5. Typical values are at +25°C, typical supply voltages and typical processing parameters.
6. For DCLK input.
7. C_L less than 150pF minimum could be faster.

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = See figure 14[3, 4, 5, 6]

PARAMETER		TEST CONDITIONS	LIMITS				UNIT
			25MHz Version		18MHz Version		
			Min	Max	Min	Max	
Dot clock (figure 11)							
f_D	Frequency			25		18	MHz
t_{DH}	High		15		22		ns
t_{DL}	Low		15		22		ns
Set-up times to $\overline{CCLK}$ (figures 5, 6, 7 and 11)							
t_{BS}	BLANK		50		50		ns
t_{SC}	BLINK, UL, STR, LL (ref to BLANK)		20		20		ns
t_{SA}	Attributes		45		55		ns
t_{SD}	Dot data D0 – D9		70		70		ns
t_{SK}	CURSOR		50		50		ns
t_{FS}	AFLG		50		65		ns
t_{SH}	HDOT		45		55		ns
Hold times from $\overline{CCLK}$ (figures 5, 6, 7 and 11)							
t_{HC}	BLINK, UL, STR, LL (ref to BLANK)		20		20		ns
t_{HA}	Attributes		20		20		ns
t_{HD}	Dot data D0 – D9		30		30		ns
t_{HK}	CURSOR		20		20		ns
t_{FH}	AFLG		30		30		ns
t_{HH}	HDOT		20		20		ns
Set-up times to DCLK (figures 9,10)							
t_{SG}	BKGND		15		15		ns
t_{SB}	RBLANK		15		15		ns
t_{CS}	CC0 – CC2		30		35		ns
Hold times from DCLK (figures 9,10)							
t_{HG}	BKGND		15		15		ns
t_{HB}	RBLANK		15		15		ns
t_{CH}	CC0 – CC2		20		20		ns
Delay times (figures 7 and 8)		$C_L = 150pF$					
t_{DGM}	GMD from DCLK			65		65	ns
t_{DC}	$\overline{MCLK}$, $\overline{CCLK}$ from DCLK			65		65	ns
t_{DV}[7]	TTLVID1 and TTLVID2 from DCLK			75		80	ns



NOTES:
1. Attributes include: ABLINK, ABLANK, ARVID, AUL, AHILT, and ASTR.
2. One $\overline{CCLK}$ delay for dot data (obtained from delay through character generator).
3. See figure 8 for detail timing of TTLVID1,TTLVID2
4. Non-active scan time. VIDEO reverts to polarity selected by the BKGND input.

Figure 5. VAC Pipeline Timing

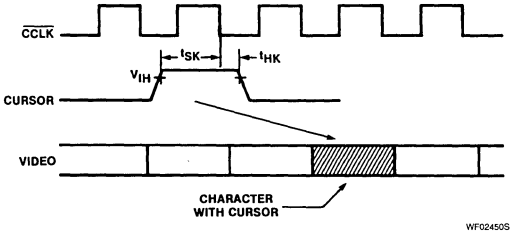
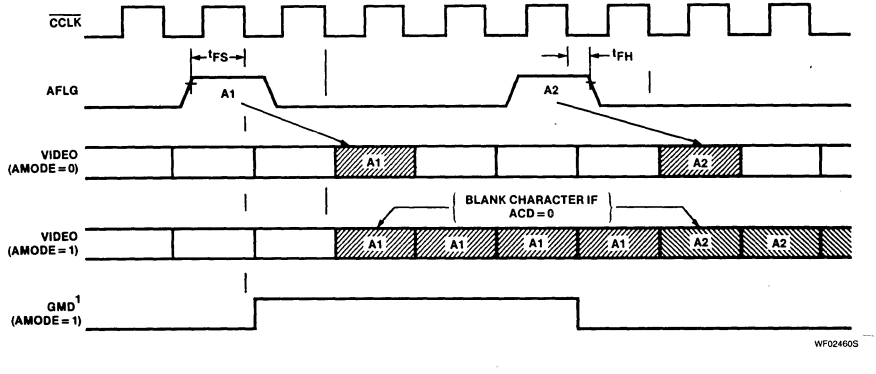
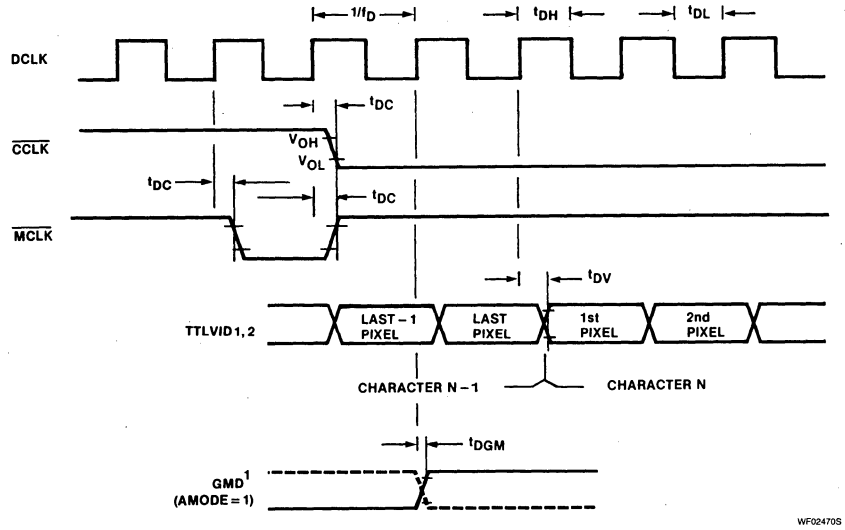


Figure 6. Cursor Pipeline Timing



NOTE:
1. GMD output in SCB2677B version only. See figure 8 for detail timing.

Figure 7. Character (AMODE = 0), Field (AMODE = 1), and GMD Attribute Timing



NOTE:
1. GMD output in SCB2677B version only.

Figure 8. Video and GMD Pipeline Timing

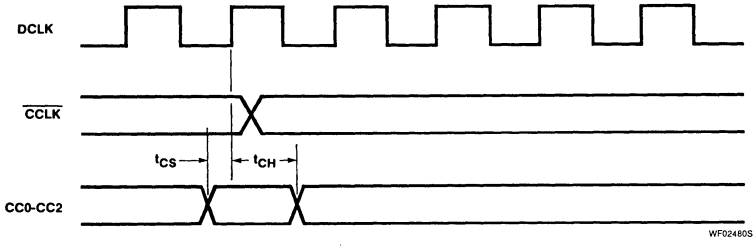


Figure 9. Character Clock Counter Timing

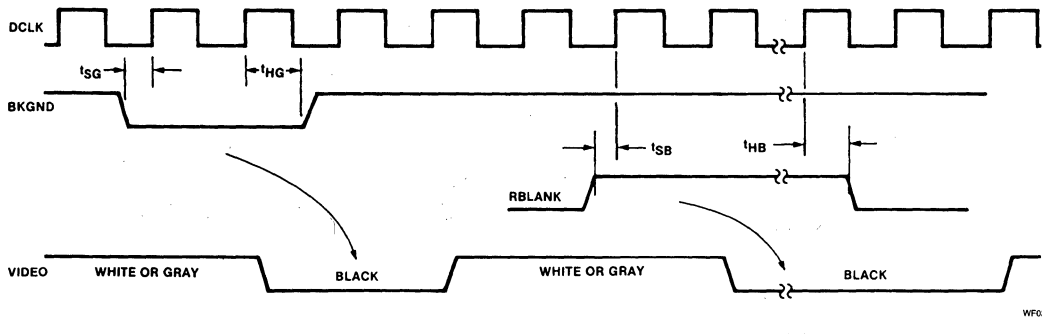


Figure 10. BKGND and RBLANK Timing During Inactive Scan Time (Blank = 1)

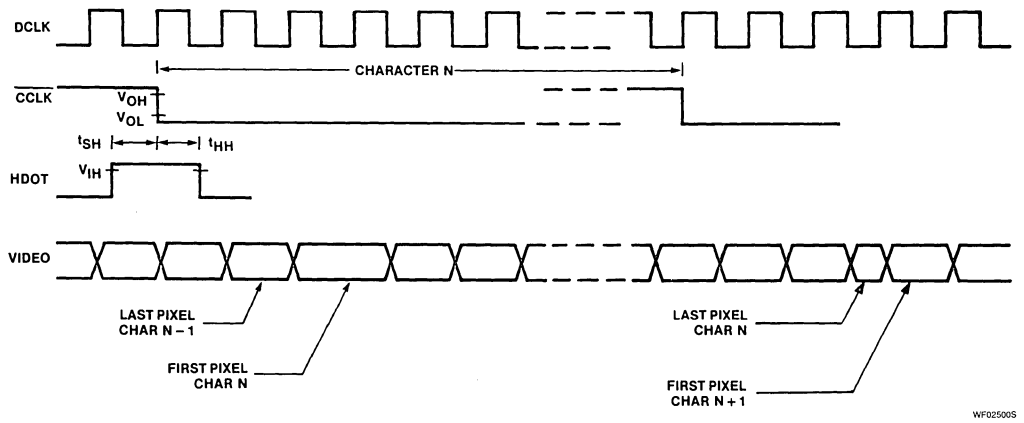


Figure 11. Half Dot Shift Timing

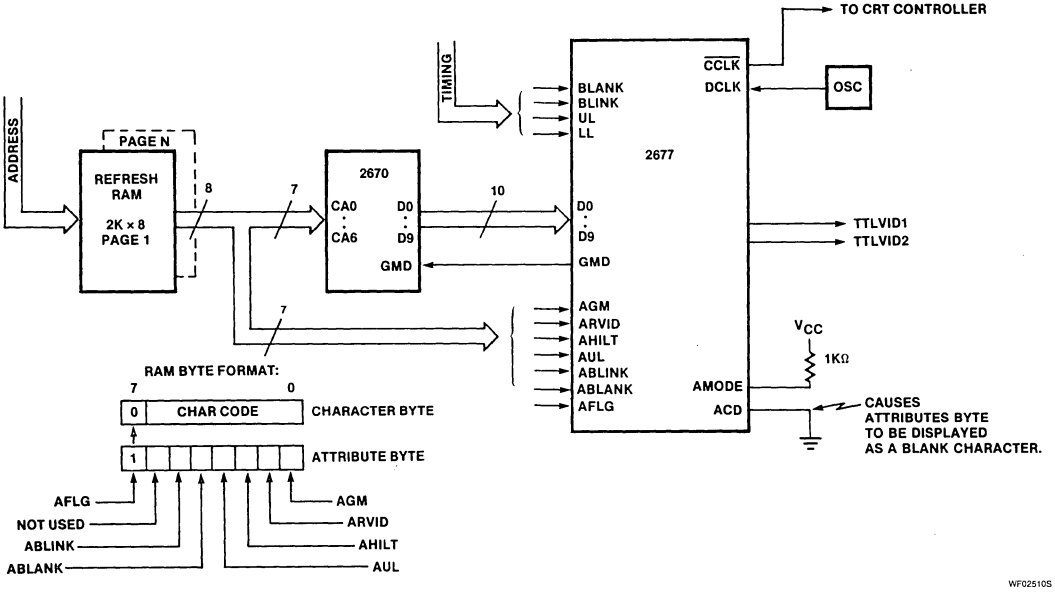
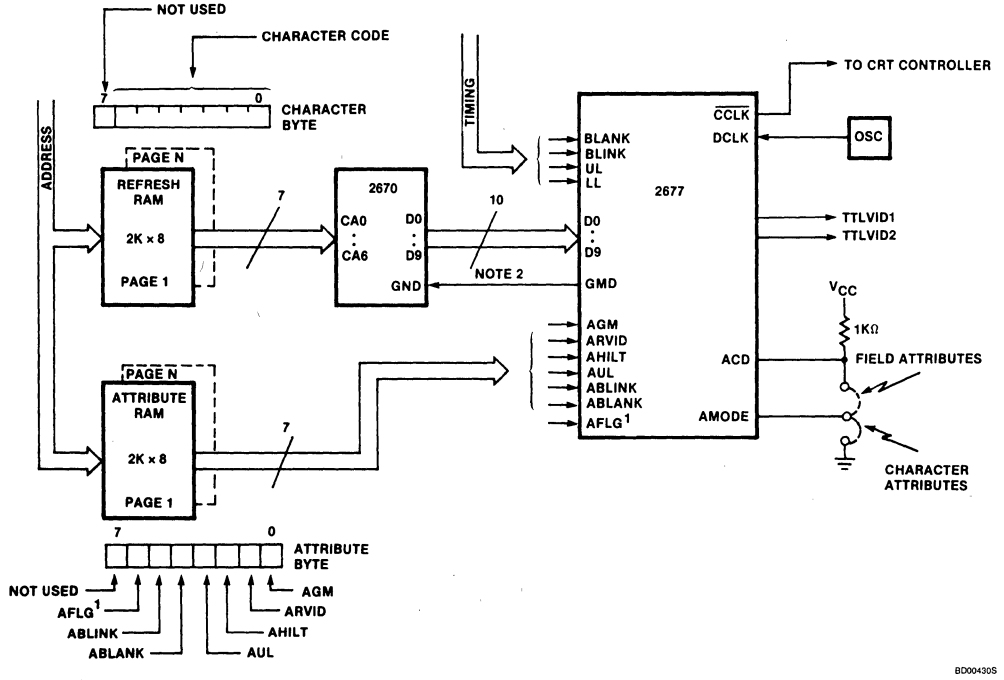


Figure 12. System Block Diagram of the 2677 in Field Attribute Mode Using the Narrow RAM (8 Wide) Configuration

Video Attributes Controller (VAC) SCB2677



NOTE:
1. For operation in character attribute mode, tie AFLG high.
2. In character attribute mode, AGM output from RAM should be connected directly to the SCN2670 GMD input.

Figure 13. System Block Diagram of the 2677 in Field or Character Attribute Mode Using the Wide RAM Configuration

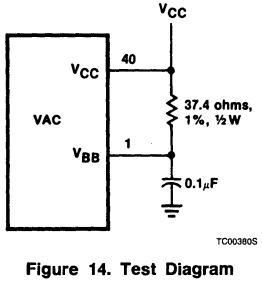


Figure 14. Test Diagram

Signetics

SCN2681

Dual Asynchronous Receiver/ Transmitter (DUART)

Product Specification

Microprocessor Products

2

DESCRIPTION

The Signetics SCN2681 Dual Universal Asynchronous Receiver/Transmitter (DUART) is a single chip MOS-LSI communications device that provides two independent full-duplex asynchronous receiver/transmitter channels in a single package. It interfaces directly with microprocessors and may be used in a polled or interrupt driven system.

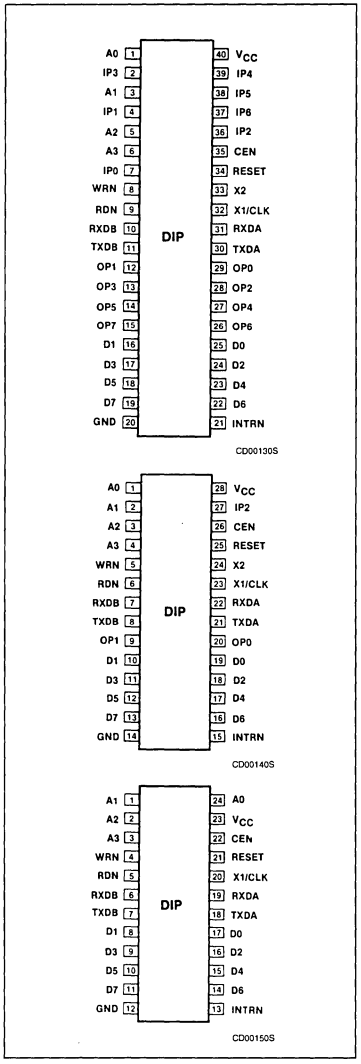
The operating mode and data format of each channel can be programmed independently. Additionally, each receiver and transmitter can select its operating speed as one of eighteen fixed baud rates, a 16x clock derived from a programmable counter/timer, or an external 1x or 16x clock. The baud rate generator and counter/timer can operate directly from a crystal or from external clock inputs. The ability to independently program the operating speed of the receiver and transmitter make the DUART particularly attractive for dual-speed channel applications such as clustered terminal systems.

Each receiver is quadruply buffered to minimize the potential of receiver overrun or to reduce interrupt overhead in interrupt driven systems. In addition, a flow control capability is provided to disable a remote DUART transmitter when the buffer of the receiving device is full.

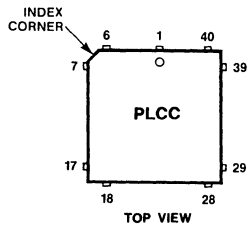
FEATURES

- **Dual full-duplex asynchronous receiver/transmitter**
- **Quadruple buffered receiver data registers**
- **Programmable data format**
  - **5 to 8 data bits plus parity**
  - **Odd, even, no parity or force parity**
  - **1, 1.5 or 2 stop bits programmable in 1/16 bit increments**
- **Programmable baud rate for each receiver and transmitter selectable from:**
  - **18 fixed rates: 50 to 38.4K baud**
  - **One user defined rate derived from programmable timer/ counter**
  - **External 1x or 16x clock**
- **Parity, framing, and overrun error detection**
- **False start bit detection**
- **Line break detection and generation**
- **Programmable channel mode**
  - **Normal (full duplex)**
  - **Automatic echo**
  - **Local loopback**
  - **Remote loopback**
- **Multi-function programmable 16-bit counter/timer**
- **Multi-function 7-bit input port**
  - **Can serve as clock or control inputs**
  - **Change of state detection on four inputs**
- **Multi-function 8-bit output port**
  - **Individual bit set/reset capability**
  - **Outputs can be programmed to be status/interrupt signals**
- **Versatile interrupt system**
  - **Single interrupt output with eight maskable interrupting conditions**
  - **Output port can be configured to provide a total of up to six separate wire-OR'able interrupt outputs**
- **Maximum data transfer: 1X–1MB/ sec, 16X–125KB/sec**
- **Automatic wake-up mode for multidrop applications**
- **Start-end break interrupt/status**
- **Detects break which originates in the middle of a character**
- **On-chip crystal oscillator**
- **TTL compatible**
- **Single +5V power supply**

PIN CONFIGURATION



PIN CONFIGURATION (Continued)



CD0044PS

Pin	Function	Pin	Function
1	NC	23	NC
2	A0	24	INTRN
3	IP3	25	D6
4	A1	26	D4
5	IP1	27	D2
6	A2	28	D0
7	A3	29	OP6
8	IP0	30	OP4
9	WRN	31	OP2
10	RDN	32	OP0
11	RXDB	33	TXDA
12	NC	34	NC
13	TXDB	35	RXDA
14	OP1	36	X1/CLK
15	OP3	37	X2
16	OP5	38	RESET
17	OP7	39	CEN
18	D1	40	IP2
19	D3	41	IP6
20	D5	42	IP5
21	D7	43	IP4
22	GND	44	V_{CC}

Also provided on the SCN2681 are a multipurpose 7-bit input port and a multipurpose 8-bit output port. These can be used as general purpose I/O ports or can be assigned specific functions (such as clock inputs or status/interrupt outputs) under program control.

The SCN2681 is available in four package versions: 40-pin and 28-pin, both 0.6" wide DIPs; a compact 24-pin 0.4" wide DIP; and a 44-pin PLCC.

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C			
	24-Pin[1]	28-Pin[2]	40-Pin[2]	44-Pin
Ceramic DIP	Not available	SCN2681AC1I28	SCN2681AC1I40	Not available
Plastic DIP	SCN2681AC1N24	SCN2681AC1N28	SCN2681AC1N40	Not available
Plastic LCC	Not available	Not available	Not available	SCN2681AC1A44

[1]400 mil wide DIP
[2]600 mil wide DIP

PIN DESCRIPTION

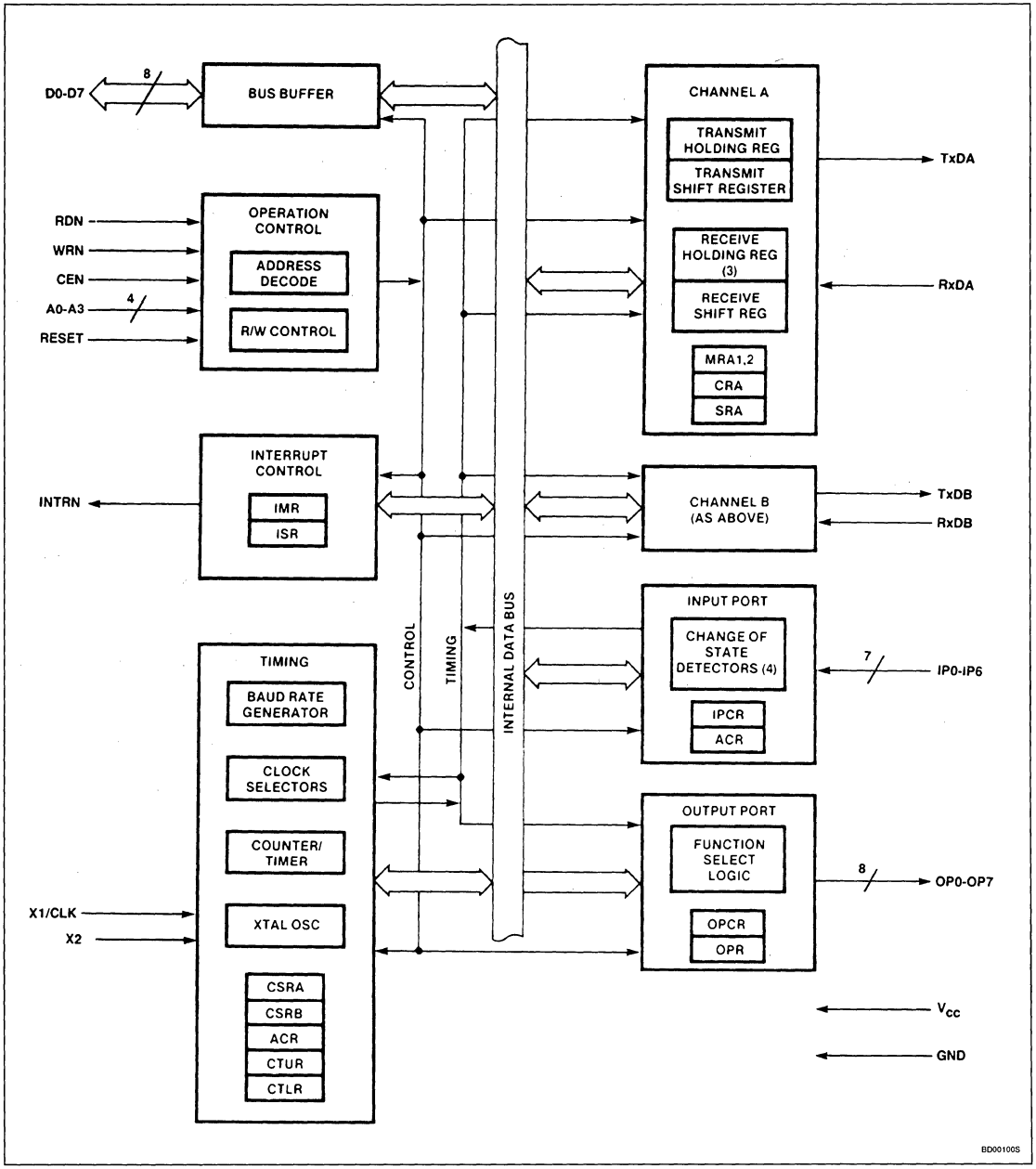
MNEMONIC	APPLICABLE			TYPE	NAME AND FUNCTION
	40	28	24		
D0-D7	X	X	X	I/O	**Data Bus:** Bidirectional 3-state data bus used to transfer commands, data and status between the DUART and the CPU. D0 is the least significant bit.
CEN	X	X	X	I	**Chip Enable:** Active low input signal. When low, data transfers between the CPU and the DUART are enabled on D0-D7 as controlled by the WRN, RDN and A0-A3 inputs. When high, places the D0-D7 lines in the 3-state condition.
WRN	X	X	X	I	**Write Strobe:** When low and CEN is also low, the contents of the data bus is loaded into the addressed register. The transfer occurs on the rising edge of the signal.
RDN	X	X	X	I	**Read Strobe:** When low and CEN is also low, causes the contents of the addressed register to be presented on the data bus. The read cycle begins on the falling edge of RDN.
A0-A3	X	X	X	I	**Address Inputs:** Select the DUART internal registers and ports for read/write operations.
RESET	X	X	X	I	**Reset:** A high level clears internal registers (SRA, SRB, IMR, ISR, OPR, OPCR), puts OP0-OP7 in the high state, stops the counter/timer, and puts channels A and B in the inactive state, with the TxDA and TxDB outputs in the mark (high) state.
INTRN	X	X	X	O	**Interrupt Request:** Active low, open drain, output which signals the CPU that one or more of the eight maskable interrupting conditions are true.
X1/CLK	X	X	X	I	**Crystal 1:** Crystal or external clock input. A crystal or clock of the specified limits must be supplied at all times. When a crystal is used, a capacitor must be connected from this pin to ground (see figure 5).
X2	X	X		I	**Crystal 2:** Connection for other side of the crystal. When a crystal is used, a capacitor must be connected from this pin to ground (see figure 5).
RxDA	X	X	X	I	**Channel A Receiver Serial Data Input:** The least significant bit is received first. 'Mark' is high, 'space' is low.

2

PIN DESCRIPTION (Continued)

MNEMONIC	APPLICABLE			TYPE	NAME AND FUNCTION
	40	28	24		
RxDB	X	X	X	I	**Channel B Receiver Serial Data Input:** The least significant bit is received first. 'Mark' is high, 'space' is low.
TxDA	X	X	X	O	**Channel A Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the 'mark' condition when the transmitter is disabled, idle, or when operating in local loopback mode. 'Mark' is high, 'space' is low.
TxDB	X	X	X	O	**Channel B Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the 'mark' condition when the transmitter is disabled, idle, or when operating in local loopback mode. 'Mark' is high, 'space' is low.
OP0	X	X		O	**Output 0:** General purpose output, or channel A request to send (RTSAN, active low). Can be deactivated automatically on receive or transmit.
OP1	X	X		O	**Output 1:** General purpose output, or channel B request to send (RTSBN, active low). Can be deactivated automatically on receive or transmit.
OP2	X			O	**Output 2:** General purpose output, or channel A transmitter 1X or 16X clock output, or channel A receiver 1X clock output.
OP3	X			O	**Output 3:** General purpose output, or open drain, active low counter/timer output, or channel B transmitter 1X clock output, or channel B receiver 1X clock output.
OP4	X			O	**Output 4:** General purpose output, or channel A open drain, active low, RxRDYA/FFULLA output.
OP5	X			O	**Output 5:** General purpose output, or channel B open drain, active low, RxRDYB/FFULLB output.
OP6	X			O	**Output 6:** General purpose output, or channel A open drain, active low, TxRDYA output.
OP7	X			O	**Output 7:** General purpose output, or channel B open drain, active low, TxRDYB output.
IP0	X			I	**Input 0:** General purpose input, or channel A clear to send active low input (CTSAN).
IP1	X			I	**Input 1:** General purpose input, or channel B clear to send active low input (CTSBN).
IP2	X	X		I	**Input 2:** General purpose input, or counter/timer external clock input.
IP3	X			I	**Input 3:** General purpose input, or channel A transmitter external clock input (TxCA). When the external clock is used by the transmitter, the transmitted data is clocked on the falling edge of the clock.
IP4	X			I	**Input 4:** General purpose input, or channel A receiver external clock input (RxCA). When the external clock is used by the receiver, the received data is sampled on the rising edge of the clock.
IP5	X			I	**Input 5:** General purpose input, or channel B transmitter external clock input (TxCB). When the external clock is used by the transmitter, the transmitted data is clocked on the falling edge of the clock.
IP6	X			I	**Input 6:** General purpose input or channel B receiver external clock input (RxCB). When the external clock is used by the receiver, the received data is sampled on the rising edge of the clock.
V_{CC}	X	X	X	I	**Power Supply:** +5V supply input
GND	X	X	X	I	**Ground**

BLOCK DIAGRAM



BLOCK DIAGRAM

The 2681 DUART consists of the following eight major sections: data bus buffer, operation control, interrupt control, timing, communications channels A and B, input port and output port. Refer to the block diagram.

Data Bus Buffer

The data bus buffer provides the interface between the external and internal data busses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the DUART.

Operation Control

The operation control logic receives operation commands from the CPU and generates appropriate signals to internal sections to control device operation. It contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer.

Interrupt Control

A single active low interrupt output (INTRN) is provided which is activated upon the occurrence of any of eight internal events. Associated with the interrupt system are the interrupt mask register (IMR) and the interrupt status register (ISR). The IMR may be programmed to select only certain conditions to cause INTRN to be asserted. The ISR can be read by the CPU to determine all currently active interrupting conditions.

Outputs OP3-OP7 can be programmed to provide discrete interrupt outputs for the transmitters, receivers, and counter/timer.

Timing Circuits

The timing block consists of a crystal oscillator, a baud rate generator, a programmable 16-bit counter/timer, and four clock selectors. The crystal oscillator operates directly from a 3.6864MHz crystal connected across the X1/CLK and X2 inputs. If an external clock of the appropriate frequency is available, it may be connected to X1/CLK. The clock serves as the basic timing reference for the baud rate generator (BRG), the counter/timer, and other internal circuits. A clock signal within the limits specified in the specifications section of this data sheet must always be supplied to the DUART.

If an external is used instead of a crystal, both X1 and X2 (of the 28/40 pin versions) should be driven using a configuration similar to the one in figure 5. For the 24 pin version in which only X1/CLK is available, the input clock must be capable of attaining a V_{IH} of 4.4 volts.

The baud rate generator operates from the oscillator or external clock input and is capable of generating 18 commonly used data communications baud rates ranging from 50 to 38.4K baud. The clock outputs from the BRG are at 16X the actual baud rate. The counter/timer can be used as a timer to produce a 16X clock for any other baud rate by counting down the crystal clock or an external clock. The four clock selectors allow the independent selection, for each receiver and transmitter, of any of these baud rates or an external timing signal.

The counter/timer (C/T) can be programmed to use one of several timing sources as its input. The output of the C/T is available to the clock selectors and can also be programmed to be output at OP3. In the counter mode, the contents of the C/T can be read by the CPU and it can be stopped and started under program control. In the timer mode, the C/T acts as a programmable divider.

Communications Channels A And B

Each communications channel of the 2681 comprises a full duplex asynchronous receiver/transmitter (UART). The operating frequency for each receiver and transmitter can be selected independently from the baud rate generator, the counter timer, or from an external input.

The transmitter accepts parallel data from the CPU, converts it to a serial bit stream, inserts the appropriate start, stop, and optional parity bits and outputs a composite serial stream of data on the TxD output pin. The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for start bit, stop bit, parity bit (if any), or break condition and sends an assembled character to the CPU.

The input port pulse detection circuitry uses a 38.4KHz sampling clock derived from one of the baud rate generator taps. This results in a sampling period of slightly more than 25μsec (this assumes that the clock input is 3.6864MHz). The detection circuitry, in order to guarantee that a true change in level has occurred, requires two successive samples at the new logic level be observed. As a consequence, the minimum duration of the signal change is 25μsec if the transition occurs "coincident with the first sample pulse." The 50μsec time refers to the situation in which the change of state is "just missed" and the first change of state is not detected until 25μsec later.

Input Port

The inputs to this unlatched 7-bit port can be read by the CPU by performing a read operation at address D_{16}. A high input results in a logic 1 while a low input results in a logic 0. D_7 will always be read as a logic 1. The pins of this port can also serve as auxiliary inputs to certain portions of the DUART logic.

Four change-of-state detectors are provided which are associated with inputs IP3, IP2, IP1, and IP0. A high-to-low or low-to-high transition of these inputs, lasting longer than 25-50μs, will set the corresponding bit in the input port change register. The bits are cleared when the register is read by the CPU. Any change of state can also be programmed to generate an interrupt to the CPU.

Output Port

The 8-bit multi-purpose output port can be used as a general purpose output port, in which case the outputs are the complements of the output port register (OPR). OPR[n] = 1 results in OP[n] = low and vice versa. Bits of the OPR can be individually set and reset. A bit is set by performing a write operation at address E_{16} with the accompanying data specifying the bits to be set (1 = set, 0 = no change). Likewise, a bit is reset by a write at address F_{16} with the accompanying data specifying the bits to be reset (1 = reset, 0 = no change).

Outputs can be also individually assigned specific functions by appropriate programming of the channel A mode registers (MR1A, MR2A), the channel B mode registers (MR1B, MR2B), and the output port configuration register (OPCR).

OPERATION

Transmitter

The 2681 is conditioned to transmit data when the transmitter is enabled through the command register. The 2681 indicates to the CPU that it is ready to accept a character by setting the TxRDY bit in the status register. This condition can be programmed to generate an interrupt request at OP6 or OP7 and INTRN. When a character is loaded into the transmit holding register (THR), the above conditions are negated. Data is transferred from the holding register to transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again which means one full character time of buffering is provided. Characters cannot be loaded into the THR while the transmitter is disabled

The transmitter converts the parallel data from the CPU to a serial bit stream on the TxD output pin. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The least significant bit is sent first. Following the transmission of the stop bits, if a new character is not available in the THR, the TxD output remains high and the TxEMT bit in the status register (SR) will be set to 1. Transmission resumes and the TxEMT bit is cleared when the CPU loads a new character into the THR. If the transmitter is disabled, it continues operating until the character currently being

transmitted is completely sent out. The transmitter can be forced to send a continuous low condition by issuing a send break command.

The transmitter can be reset through a software command. If it is reset, operation ceases immediately and the transmitter must be enabled through the command register before resuming operation. If CTS operation is enabled, the CTSN input must be low in order for the character to be transmitted. If it goes high in the middle of a transmission, the character in the shift register is transmitted and TxDA then remains in the marking state until CTSN goes low. The transmitter can also control the deactivation of the RTSN output. If programmed, the RTSN output will be reset one bit time after the character in the transmit shift register and transmit holding register (if any) are completely transmitted, if the transmitter has been disabled.

Receiver

The 2681 is conditioned to receive data when enabled through the command register. The receiver looks for a high to low (mark to space) transition of the start bit on the RxD input pin. If a transition is detected, the state of the RxD pin is sampled each 16X clock for 7-½ clocks (16X clock mode) or at the next rising edge of the bit time clock (1X clock mode). If RxD is sampled high, the start bit is invalid and the search for a valid start bit begins again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input at one bit time intervals at the theoretical center of the bit, until the proper number of data bits and the parity bit (if any) have been assembled, and one stop bit has been detected. The least significant bit is received first. The data is then transferred to the receive holding register (RHR) and the RxRDY bit in the SR is set to a 1. This condition can be programmed to generate an interrupt at OP4 or OP5 and INTRN. If the character length is less than eight bits, the most significant unused bits in the RHR are set to zero.

After the stop bit is detected, the receiver will immediately look for the next start bit. However, if a non-zero character was received without a stop bit (framing error) and RxD remains low for one half of the bit period after the stop bit was sampled, then the receiver operates as if a new start bit transition had been detected at that point (one-half bit time after the stop bit was sampled).

The parity error, framing error, overrun error and received break state (if any) are strobed into the SR at the received character boundary, before the RxRDY status bit is set. If a break condition is detected (RxD is low for the entire character including the stop bit), a character consisting of all zeros will be loaded into the RHR and the received break bit in the SR is set to 1. The RxD input must return to a high condition for at least one-half bit time before a search for the next start bit begins.

The RHR consists of a first-in-first-out (FIFO) stack with a capacity of three characters. Data is loaded from the receive shift register into the topmost empty position of the FIFO. The RxRDY bit in the status register is set whenever one or more characters are available to be read, and a FFULL status bit is set if all three stack positions are filled with data. Either of these bits can be selected to cause an interrupt. A read of the RHR outputs the data at the top of the FIFO. After the read cycle, the data FIFO and its associated status bits (see below) are 'popped' thus emptying a FIFO position for new data.

In addition to the data word, three status bits (parity error, framing error, and received break) are also appended to each data character in the FIFO (overrun is not). Status can be provided in two ways, as programmed by the error mode control bit in the mode register. In the 'character' mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these three bits is the logical OR of the status for all characters coming to the top of the FIFO since the last 'reset error' command was issued. In either mode reading the SR does not affect the FIFO. The FIFO is 'popped' only when the RHR is read. Therefore the status register should be read prior to reading the FIFO.

If the FIFO is full when a new character is received, that character is held in the receive shift register until a FIFO position is available. If an additional character is received while this state exits, the contents of the FIFO are not affected: the character previously in the shift register is lost and the overrun error status bit (SR[4]) will be set-upon receipt of the start bit of the new (overruning) character.

The receiver can control the deactivation of RTS. If programmed to operate in this mode, the RTSN output will be negated when a valid start bit was received and the FIFO is full. When a FIFO position becomes available, the RTSN output will be re-asserted automatically. This feature can be used to prevent an overrun, in the receiver, by connecting the RTSN output to the CTSN input of the transmitting device.

If the receiver is disabled, the FIFO characters can be read. However, no additional characters can be received until the receiver is enabled again. If the receiver is reset, the FIFO and all of the receiver status, and the corresponding output ports and interrupt are reset. No additional characters can be received until the receiver is enabled again.

Multidrop Mode

The DUART is equipped with a wake up mode used for multidrop applications. This mode is selected by programming bits MR1A[4:3] or MR1B[4:3] to '11' for channels A and B respectively. In this mode of operation, a 'master' station transmits an address character followed by data characters for the addressed 'slave' station. The slave stations, with receivers that are normally disabled, examine the received data stream and 'wake-up' the CPU (by setting RxRDY) only upon receipt of an address character. The CPU compares the received address to its station address and enables the receiver if it wishes to receive the subsequent data characters. Upon receipt of another address character, the CPU may disable the receiver to initiate the process again.

A transmitted character consists of a start bit, the programmed number of data bits, an address/data (A/D) bit, and the programmed number of stop bits. The polarity of the transmitted A/D bit is selected by the CPU by programming bit MR1A[2]/MR1B[2]. MR1A[2]/MR1B[2] = 0 transmits a zero in the A/D bit position, which identifies the corresponding data bits as data, while MR1A[2]/MR1B[2] = 1 transmits a one in the A/D bit position, which identifies the corresponding data bits as an address. The CPU should program the mode register prior to loading the corresponding data bits into the THR.

In this mode, the receiver continuously looks at the received data stream, whether it is enabled or disabled. If disabled, it sets the RxRDY status bit and loads the character into the RHR FIFO if the received A/D bit is a one (address tag), but discards the received character if the received A/D bit is a zero (data tag). If enabled, all received characters are transferred to the CPU via the RHR. In either case, the data bits are loaded into the data FIFO while the A/D bit is loaded into the status FIFO position normally used for parity error (SRA[5] or SRB[5]). Framing error, overrun error, and break detect operate normally whether or not the receiver is enabled.

PROGRAMMING

The operation of the DUART is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The addressing of the registers is described in table 1.

The contents of certain control registers are initialized to zero on RESET. Care should be exercised if the contents of a register are changed during operation, since certain changes may cause operational problems.

Table 1. 2681 REGISTER ADDRESSING

A3	A2	A1	A0	READ (RDN = 0)	WRITE (WRN = 0)
0	0	0	0	Mode Register A (MR1A, MR2A)	Mode Register A (MR1A, MR2A)
0	0	0	1	Status Register A (SRA)	Clock Select Reg. A (CSRA)
0	0	1	0	*Reserved*	Command Register A (CRA)
0	0	1	1	RX Holding Register A (RHRA)	TX Holding Register A (THRA)
0	1	0	0	Input Port Change Reg. (IPCR)	Aux. Control Register (ACR)
0	1	0	1	Interrupt Status Reg. (ISR)	Interrupt Mask Reg. (IMR)
0	1	1	0	Counter/Timer Upper (CTU)	C/T Upper Register (CTUR)
0	1	1	1	Counter/Timer Lower (CTL)	C/T Lower Register (CTLR)
1	0	0	0	Mode Register B (MR1B, MR2B)	Mode Register B (MR1B, MR2B)
1	0	0	1	Status Register B (SRB)	Clock Select Reg. B (CSRB)
1	0	1	0	*Reserved*	Command Register B (CRB)
1	0	1	1	RX Holding Register B (RHRB)	TX Holding Register B (THRB)
1	1	0	0	*Reserved*	*Reserved*
1	1	0	1	Input Port	Output Port Conf. Reg. (OPCR)
1	1	1	0	Start Counter Command	Set Output Port Bits Command
1	1	1	1	Stop Counter Command	Reset Output Port Bits Command

2

For example, changing the number of bits per character while the transmitter is active may cause the transmission of an incorrect character. In general, the contents of the MR, the CSR, and the OPCR should only be changed while the receiver(s) and transmitter(s) are not enabled, and certain changes to the ACR should only be made while the C/T is stopped.

Mode registers 1 and 2 of each channel are accessed via independent auxiliary pointers. The pointer is set to MR1x by RESET or by issuing a 'reset pointer' command via the corresponding command register. Any read or write of the mode register while the pointer is at MR1x, switches the pointer to MR2x. The pointer then remains at MR2x, so that subsequent accesses are always to MR2x unless the pointer is reset to MR1x as described above.

Mode, command, clock select, and status registers are duplicated for each channel to provide total independent operation and control. Refer to table 2 for register bit descriptions. The reserved registers at addresses H'02' and H'0A' should never be read during normal operation since they are reserved for internal diagnostics.

MR1A – channel A Mode Register 1

MR1A is accessed when the channel A MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRA. After reading or writing MR1A, the pointer will point to MR2A.

MR1A[7] – Channel A Receiver Request-to-Send Control

This bit controls the deactivation of the RTSAN output (OP0) by the receiver. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR 1A[7] = 1 causes RTSAN to be negated upon receipt of a valid start bit if the channel A FIFO is full. However, OPR[0] is not reset and RTSAN will be asserted again when an empty FIFO position is available. This feature can be used for flow control to prevent overrun in the receiver by using the RTSAN output signal to control the CTSN input of the transmitting device.

MR1A[6] – Channel A Receiver Interrupt Select

This bit selects either the channel A receiver ready status (RXRDY) or the channel A FIFO full status (FFULL) to be used for CPU interrupts. It also causes the selected bit to be output on OP4 if it is programmed as an interrupt output via the OPCR.

MR1A[5] – Channel A Error Mode Select

This bit selects the operating mode of the three FIFOed status bits (FE, PE, received break) for channel A. In the 'character' mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these bits is the accumulation (logical OR) of the status for all characters coming to the top of the FIFO since the last 'reset error' command for channel A was issued.

Table 2. REGISTER BIT FORMATS

	BIT7	BIT6	BIT5	BIT4 BIT3	BIT2	BIT1 BIT0
	RX RTS CONTROL	RX INT SELECT	ERROR MODE	PARITY MODE	PARITY TYPE	BITS PER CHAR.
MR1A MR1B	0 = no 1 = yes	0 = RXRDY 1 = FFULL	0 = char 1 = block	00 = with parity 01 = force parity 10 = no parity 11 = multi-drop mode	0 = even 1 = odd	00 = 5 01 = 6 10 = 7 11 = 8

Table 2. REGISTER BIT FORMATS (Continued)

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	CHANNEL MODE		Tx RTS CONTROL	CTS ENABLE Tx	STOP BIT LENGTH*			
MR2A MR2B	00 = Normal 01 = Auto echo 10 = Local loop 11 = Remote loop		0 = no 1 = yes	0 = no 1 = yes	0 = 0.563 1 = 0.625 2 = 0.688 3 = 0.750	4 = 0.813 5 = 0.875 6 = 0.938 7 = 1.000	8 = 1.563 9 = 1.625 A = 1.688 B = 1.750	C = 1.813 D = 1.875 E = 1.938 F = 2.000

*Add 0.5 to values shown for 0–7 if channel is programmed for 5 bits/char.

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RECEIVER CLOCK SELECT				TRANSMITTER CLOCK SELECT			
CSRA CSRB	See Text				See Text			

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
		MISCELLANEOUS COMMANDS			DISABLE Tx	ENABLE Tx	DISABLE Rx	ENABLE Rx
CRA CRB	not used – must be 0	See Text			0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RECEIVED BREAK	FRAMING ERROR	PARITY ERROR	OVERRUN ERROR	TxEMT	TxRDY	FFULL	RxRDY
SRA SRB	0 = no 1 = yes *	0 = no 1 = yes *	0 = no 1 = yes *	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes

*These status bits are appended to the corresponding data character in the receive FIFO. A read of the status register provides these bits (7:5) from the top of the FIFO together with bits (4:0). These bits are cleared by a 'reset error status' command. In character mode they are discarded when the corresponding data character is read from the FIFO.

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	OP7	OP6	OP5	OP4	OP3		OP2	
OPCR	0 = OPR[7] 1 = TxRDYB	0 = OPR[6] 1 = TxRDYA	0 = OPR[5] 1 = RxRDY/ FFULLB	0 = OPR[4] 1 = RxRDY/ FFULLA	00 = OPR[3] 01 = C/T OUTPUT 10 = TxCB(1X) 11 = RxCB(1X)		00 = OPR[2] 01 = TxCA(16X) 10 = TxCA(1X) 11 = RxCA(1X)	

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	BRG SET SELECT	COUNTER/TIMER MODE AND SOURCE			DELTA IP3 INT	DELTA IP2 INT	DELTA IP1 INT	DELTA IP0 INT
ACR	0 = set1 1 = set2	See table 4			0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	DELTA IP3	DELTA IP2	DELTA IP1	DELTA IP0	IP3	IP2	IP1	IP0
IPCR	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = low 1 = high	0 = low 1 = high	0 = low 1 = high	0 = low 1 = high

Table 2. REGISTER BIT FORMATS (Continued)

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ISR	INPUT PORT CHANGE	DELTA BREAK B	RxRDY/ FFULLB	TxRDYB	COUNTER READY	DLETA BREAK A	RxRDY/ FFULLA	TxRDYA
	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
IMR	IN. PORT CHANGE INT	DELTA BREAK B INT	RxRDY/ FFULLB INT	TxRDYB INT	COUNTER READY INT	DELTA BREAK A INT	RxRDY/ FFULLA INT	TxRDYA INT
	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CTUR	C/T[15]	C/T[14]	C/T[13]	C/T[12]	C/T[11]	C/T[10]	C/T[9]	C/T[8]

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CTLR	C/T[7]	C/T[6]	C/T[5]	C/T[4]	C/T[3]	C/T[2]	C/T[1]	C/T[0]

MR1A[4:3] – Channel A Parity Mode Select
If 'with parity' or 'force parity' is selected, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR1A[4:3] = 11 selects channel A to operate in the special multidrop mode described in the Operation section.

MR1A[2] – Channel A Parity Type Select
This bit selects the parity type (odd or even) if the 'with parity' mode is programmed by MR1A[4:3], and the polarity of the forced parity bit if the 'force parity' mode is programmed. It has no effect if the 'no parity' mode is programmed. In the special multidrop mode it selects the polarity of the A/D bit.

MR1A[1:0] – Channel A Bits Per Character Select
This field selects the number of data bits per character to be transmitted and received. The character length does not include the start, parity, and stop bits.

MR2A – Channel A Mode Register 2

MR2A is accessed when the channel A MR pointer points to MR2, which occurs after any access to MR1A. Accesses to MR2A do not change the pointer.

MR2A[7:6] – Channel A Mode Select
Each channel of the DUART can operate in one of four modes. MR2A[7:6] = 00 is the normal mode, with the transmitter and receiver operating independently. MR2A[7:6] = 01 places the channel in the automatic echo mode, which automatically retransmits the received data. The following conditions are true while in automatic echo mode:

1. Received data is reclocked and retransmitted on the TxDA output.
2. The receive clock is used for the transmitter.
3. The receiver must be enabled, but the transmitter need not be enabled.
4. The channel A TxRDY and TxEMT status bits are inactive.
5. The received parity is checked, but is not regenerated for transmission, i.e., transmitted parity bit is as received.
6. Character framing is checked, but the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.
8. CPU to receiver communication continues normally, but the CPU to transmitter link is disabled.

Two diagnostic modes can also be configured. MR2A[7:6] = 10 selects local loopback mode. In this mode:

1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver.
3. The TxDA output is held high.
4. The RxDA input is ignored.
5. The transmitter must be enabled, but the receiver need not be enabled.
6. CPU to transmitter and receiver communications continue normally.

The second diagnostic mode is the remote loopback mode, selected by MR2A[7:6] = 11. In this mode:

1. Received data is relocked and retransmitted on the TxDA output.
2. The receive clock is used for the transmitter.
3. Received data is not sent to the local CPU, and the error status conditions are inactive.
4. The received parity is not checked and is not regenerated for transmission, i.e., transmitted parity bit is as received.
5. The receiver must be enabled.
6. Character framing is not checked, and the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.

The user must exercise care when switching into and out of the various modes. The selected mode will be activated immediately upon mode selection, even if this occurs in the middle of a received or transmitted char-

acter. Likewise, if a mode is deselected, the device will switch out of the mode immediately. An exception to this is switching out of autoecho or remote loopback modes: if the de-selection occurs just after the receiver has sampled the stop bit (indicated in autoecho by assertion of RxRDY), and the transmitter is enabled, the transmitter will remain in autoecho mode until the entire stop bit has been retransmitted.

MR2A[5] – Channel A Transmitter Request-to-Send Control

This bit controls the deactivation of the RTSAN output (OP0) by the transmitter. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR 2A[5] = 1 causes OPR[0] to be reset automatically one bit time after the characters in the channel A transmit shift register and in the THR, if any, are completely transmitted, including the programmed number of stop bits, if the transmitter is not enabled. This feature can be used to automatically terminate the transmission of a message as follows:

1. Program auto-reset mode: MR2A[5] = 1.
2. Enable transmitter.
3. Assert RTSAN: OPR[0] = 1.
4. Send message.
5. Verify the message is sent by waiting until the transmit ready status (TxRDY) is asserted. Disable transmitter after the last character is loaded into the channel A THR.
6. The last character will be transmitted and OPR[0] will be reset one bit time after the last stop bit, causing RTSAN to be negated.

MR2A[4] – Channel A Clear-to-send Control

If this bit is 0, CTSAN has no effect on the transmitter. If this bit is a 1, the transmitter checks the state of CTSAN (IP0) each time it is ready to send a character. If IP0 is asserted (low), the character is transmitted. If it is negated (high), the TxDA output remains in the marking state and the transmission is delayed until CTSAN goes low. Changes in CTSAN while a character is being transmitted do not affect the transmission of that character.

MR2A[3:0] – Channel A Stop Bit Length Select

This field programs the length of the stop bit appended to the transmitted character. Stop bit lengths of $^{9}/_{16}$ to 1 and $1^{9}/_{16}$ to 2 bits, in increments of $^{1}/_{16}$ bit, can be programmed for character lengths of 6, 7, and 8 bits. For a character length of 5 bits, $1^{1}/_{16}$ to 2 stop bits can be programmed in increments of $^{1}/_{16}$ bit. The receiver only checks for a 'mark' condition at the center of the first stop bit position (one bit time after the last data bit, or after the parity bit if parity is enabled) in all cases.

If an external 1X clock is used for the transmitter, MR2A[3] = 0 selects one stop bit and MR2A[3] = 1 selects two stop bits to be transmitted.

MR1B – Channel B Mode Register 1

MR1B is accessed when the channel B MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRB. After reading or writing MR1B, the pointer will point to MR2B.

The bit definitions for this register are identical to the bit definitions for MR1A, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

MR2B – Channel B Mode Register 2

MR2B is accessed when the channel B MR pointer points to MR2, which occurs after any access to MR1B. Accesses to MR2B do not change the pointer.

The bit definitions for this register are identical to the bit definitions for MR2A, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

CSRA – Channel A Clock Select Register

CSRA[7:4] – Channel A Receiver Clock Select

This field selects the baud rate clock for the channel A receiver as follows:

CSRA[7:4]	Baud Rate ACR[7] = 0	ACR[7] = 1
0 0 0 0	50	75
0 0 0 1	110	110
0 0 1 0	134.5	134.5
0 0 1 1	200	150
0 1 0 0	300	300
0 1 0 1	600	600
0 1 1 0	1,200	1,200
0 1 1 1	1,050	2,000
1 0 0 0	2,400	2,400
1 0 0 1	4,800	4,800
1 0 1 0	7,200	1,800
1 0 1 1	9,600	9,600
1 1 0 0	38.4K	19.2K
1 1 0 1	Timer	Timer
1 1 1 0	IP4-16X	IP4-16X
1 1 1 1	IP4-1X	IP4-1X

The receiver clock is always a 16X clock except for CSRA[7:4] = 1111.

CSRA[3:0] – Channel A Transmitter Clock Select

This field selects the baud rate clock for the channel A transmitter. The field definition is as per CSRA[7:4] except as follows:

CSRA[3:0]	Baud Rate ACR[7] = 0	ACR[7] = 1
1 1 1 0	IP3–16X	IP3–16X
0 1 1 1	IP3–1X	IP3–1X

The transmitter clock is always a 16X clock except for CSRA[3:0] = 1111.

CSRB – Channel B Clock Select Register

CSRB[7:4] – Channel B Receiver Clock Select

This field selects the baud rate clock for the channel B receiver. The field definition is as per CSRA[7:4] except as follows:

CSRB[7:4]	Baud Rate ACR[7] = 0	ACR[7] = 1
1 1 1 0	IP6–16X	IP6–16X
0 1 1 1	IP6–1X	IP6–1X

The receiver clock is always a 16X clock except for CSRB[7:4] = 1111.

CSRB[3:0] – Channel B Transmitter Clock Select

This field selects the baud rate clock for the channel B transmitter. The field definition is as per CSRA[7:4] except as follows:

CSRB[3:4]	Baud Rate ACR[7] = 0	ACR[7] = 1
1 1 1 0	IP5–16X	IP5–16X
1 1 1 1	IP5–1X	IP5–1X

The transmitter clock is always a 16X clock except for CSRB[3:0] = 1111.

CRA – Channel A Command Register

CRA is a register used to supply commands to channel A. Multiple commands can be specified in a single write to CRA as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

CRA[6:4] – Channel A Miscellaneous Commands

The encoded value of this field may be used to specify a single command as follows:

CRA[6:4]	COMMAND
0 0 0	No command.
0 0 1	Reset MR pointer. Causes the channel A MR pointer to point to MR1.
0 1 0	Reset receiver. Resets the channel A receiver as if a hardware reset had been applied. The receiver is disabled and the FIFO is flushed.
0 1 1	Reset transmitter. Resets the channel A transmitter as if a hardware reset had been applied.

CRA[6:4]	COMMAND
1 0 0	Reset error status. Clears the channel A Received Break, Parity Error, Framing Error, and Overrun Error bits in the status register (SRA[7:4]). Used in character mode to clear OE status (although RB, PE, and FE bits will also be cleared) and in block mode to clear all error status after a block of data has been received.
1 0 1	Reset channel A break change interrupt. Causes the channel A break detect change bit in the interrupt status register (ISR[2]) to be cleared to zero.
1 1 0	Start break. Forces the TXDA output low (spacing). If the transmitter is empty the start of the break condition will be delayed up to two bit times. If the transmitter is active the break begins when transmission of the character is completed. If a character is in the THR, the start of the break will be delayed until that character, or any others loaded subsequently are transmitted. The transmitter must be enabled for this command to be accepted.
1 1 1	Stop Break. The TXDA line will go high (marking) within two bit times. TXDA will remain high for one bit time before the next character, if any, is transmitted.

CRA[3] – Disable Channel A Transmitter
This command terminates transmitter operation and resets the TxRDY and TxEMT status bits. However, if a character is being transmitted or if a character is in the THR when the transmitter is disabled, the transmission of the character(s) is completed before assuming the inactive state.

CRA[2] – Enable Channel A Transmitter
Enables operation of the channel A transmitter. The TxRDY status bit will be asserted.

CRA[1] – Disable Channel A Receiver
This command terminates operation of the receiver immediately – a character being received will be lost. The command has no effect on the receiver status bits or any other control registers. If the special multidrop mode is programmed, the receiver operates even if it is disabled. See Operation section.

CRA[0] – Enable Channel A Receiver
Enables operation of the channel A receiver. If not in the special wakeup mode, this also forces the receiver into the search for start-bit state.

CRB – Channel B Command Register

CRB is a register used to supply commands to channel B. Multiple commands can be specified in a single write to CRB as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

The bit definitions for this register are identical to the bit definitions for CRA, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

SRA – Channel A Status Register

SRA[7] – Channel A Received Break
This bit indicates that an all zero character of the programmed length has been received without a stop bit. Only a single FIFO position is occupied when a break is received: further entries to the FIFO are inhibited until the RxDA line returns to the marking state for at least one-half a bit time (two successive edges of the internal or external 1x clock).

When this bit is set, the channel A 'change in break' bit in the ISR (ISR[2]) is set. ISR[2] is also set when the end of the break condition, as defined above, is detected.

The break detect circuitry can detect breaks that originate in the middle of a received character. However, if a break begins in the middle of a character, it must persist until at least the end of the next character time in order for it to be detected.

SRA[6] – Channel A Framing Error
This bit, when set, indicates that a stop bit was not detected when the corresponding data character in the FIFO was received. The stop bit check is made in the middle of the first stop bit position.

SRA[5] – Channel A Parity Error
This bit is set when the 'with parity' or 'force parity' mode is programmed and the corresponding character in the FIFO was received with incorrect parity.

In the special multidrop mode the parity error bit stores the received A/D bit.

SRA[4] – Channel A Overrun Error
This bit, when set, indicates that one or more characters in the received data stream have been lost. It is set-upon receipt of a new character when the FIFO is full and a character is already in the receive shift register waiting for an empty FIFO position. When this occurs, the character in the receive shift register (and its break detect, parity error and framing error status, if any) is lost.

This bit is cleared by a 'reset error status' command.

SRA[3] – Channel A Transmitter Empty (TxEMTA)
This bit will be set when the channel A transmitter underruns, i.e., both the transmit holding register (THR) and the transmit shift register are empty. It is set after transmission of the last stop bit of a character if no character is in the THR awaiting transmission. It is reset when the THR is loaded by the CPU or when the transmitter is disabled.

SRA[2] – Channel A Transmitter Ready (TxRDYA)
This bit, when set, indicates that the THR is empty and ready to be loaded with a character. This bit is cleared when the THR is loaded by the CPU and is set when the character is transferred to the transmit shift register. TxRDY is reset when the transmitter is disabled and is set when the transmitter is first enabled, viz., characters loaded into the THR while the transmitter is disabled will not be transmitted.

SRA[1] – Channel A FIFO Full (FFULLA)
This bit is set when a character is transferred from the receive shift register to the receive FIFO and the transfer causes the FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, FFULL will not be reset when the CPU reads the RHR.

SRA[0] – Channel A Receiver Ready (RxRDYA)
This bit indicates that a character has been received and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR, if after this read there are no more characters still in the FIFO.

SRB – Channel B Status Register

The bit definitions for this register are identical to the bit definitions for SRA, except that all status applies to the channel B receiver and transmitter and the corresponding inputs and outputs.

OPCR – Output Port Configuration Register

OPCR[7] – OP7 Output Select
This bit programs the OP7 output to provide one of the following:

–The complement of OPR[7]

–The channel B transmitter interrupt output, which is the complement of TxRDYB. When in this mode OP7 acts as an open collector

output. Note that this output is not masked by the contents of the IMR.

OPCR[6] – OP6 Output Select
This bit programs the OP6 output to provide one of the following:

–The complement of OPR[6]

–The channel A transmitter interrupt output, which is the complement of TxRDYA. When in this mode OP6 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[5] – OP5 Output Select
This bit programs the OP5 output to provide one of the following:

–The complement of OPR[5]

–The channel B receiver interrupt output, which is the complement of ISR[5]. When in this mode OP5 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[4] – OP4 Output Select
This bit programs the OP4 output to provide one of the following:

–The complement of OPR[4]

–The channel A receiver interrupt output, which is the complement of ISR[1]. When in this mode OP4 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[3:2] – OP3 Output Select
This field programs the OP3 output to provide one of the following:

–The complement of OPR[3]

–The counter/timer output, in which case OP3 acts as an open collector output. In the timer mode, this output is a square wave at the programmed frequency. In the counter mode, the output remains high until terminal count is reached, at which time it goes low. The output returns to the high state when the counter is stopped by a stop counter command. Note that this output is not masked by the contents of the IMR.

–The 1X clock for the channel B transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a free running 1X clock is output.

–The 1X clock for the channel B receiver, which is the clock that samples the received data. If data is not being received, a free running 1X clock is output.

OPCR[1:0] – OP2 Output Select
This field programs the OP2 output to provide one of the following:

–The complement of OPR[2]

–The 16X clock for the channel A transmitter. This is the clock selected by CSRA[3:0], and will be a 1X clock if CSRA[3:0] = 1111.

–The 1X clock for the channel A transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a free running 1X clock is output.

–The 1X clock for the channel A receiver, which is the clock that samples the received data. If data is not being received, a free running 1X clock is output.

ACR – Auxiliary Control Register

ACR[7] – Baud Rate Generator Set Select
This bit selects one of two sets of baud rates to be generated by the BRG:

Set 1: 50, 110, 134.5, 200, 300, 600, 1.05K, 1.2K, 2.4K, 4.8K, 7.2K, 9.6K, and 38.4K baud.

Set 2: 75, 110, 134.5, 150, 300, 600, 1.2K, 1.8K, 2.0K, 2.4K, 4.8K, 9.6K, and 19.2K baud.

The selected set of rates is available for use by the channel A and B receivers and transmitters as described in CSRA and CSRB. Baud rate generator characteristics are given in table 3.

ACR[6:4] – Counter/Timer Mode And Clock Source Select
This field selects the operating mode of the counter/timer and its clock source as shown in table 4.

ACR[3:0] – IP3, IP2, IP1, IP0 Change Of State Interrupt Enable
This field selects which bits of the input port change register (IPCR) cause the input change bit in the interrupt status register (ISR[7]) to be set. If a bit is in the 'on' state, the setting of the corresponding bit in the IPCR will also result in the setting of ISR[7], which results in the generation of an interrupt output if IMR[7] = 1. If a bit is in the 'off' state, the setting of that bit in the IPCR has no effect on ISR[7].

IPCR – Input Port Change Register

IPCR[7:4] – IP3, IP2, IP1, IP0 Change Of State
These bits are set when a change of state, as defined in the input port section of this data sheet, occurs at the respective input pins. They are cleared when the IPCR is read by the CPU. A read of the IPCR also clears ISR[7], the input change bit in the interrupt status register.

The setting of these bits can be programmed to generate an interrupt to the CPU.

IPCR[3:0] – IP3, IP2, IP1, IP0 Current State
These bits provide the current state of the respective inputs. The information is unlatched and reflects the state of the input pins at the time the IPCR is read.

ISR – Interrupt Status Register

This register provides the status of all potential interrupt sources. The contents of this register are masked by the interrupt mask register (IMR). If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the reading of the ISR – the true status will be provided regardless of the contents of the IMR. The contents of this register are initialized to 00_{16} when the DUART is reset.

ISR[7] – Input Port Change Status
This bit is a '1' when a change of state has occurred at the IP0, IP1, IP2, or IP3 inputs and that event has been selected to cause an interrupt by the programming of ACR[3:0]. The bit is cleared when the CPU reads the IPCR.

ISR[6] – Channel B Change In Break
This bit, when set, indicates that the channel B receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a channel B 'reset break change interrupt' command.

ISR[5] – Channel B Receiver Ready Or FIFO Full
The function of this bit is programmed by MR1B[6]. If programmed as receiver ready, it indicates that a character has been received in channel B and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the channel B FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

ISR[4] – Channel B Transmitter Ready
This bit is a duplicate of TxRDYB (SRB[2]).

ISR[3] – Counter Ready
In the counter mode, this bit is set when the counter reaches terminal count and is reset when the counter is stopped by a stop counter command.

Table 3. BAUD RATE GENERATOR CHARACTERISTICS CRYSTAL OR CLOCK = 3.6864MHz

NOMINAL RATE (BAUD)	ACTUAL 16X CLOCK (KHz)	ERROR (PERCENT)
50	0.8	0
75	1.2	0
110	1.759	-0.069
134.5	2.153	0.059
150	2.4	0
200	3.2	0
300	4.8	0
600	9.6	0
1050	16.756	-0.260
1200	19.2	0
1800	28.8	0
2000	32.056	0.175
2400	38.4	0
4800	76.8	0
7200	115.2	0
9600	153.6	0
19.2K	307.2	0
38.4K	614.4	0

NOTE:
Duty cycle of 16X clock is 50% ±1%

Table 4. ACR 6:4 FIELD DEFINITION

ACR 6:4	MODE	CLOCK SOURCE
0 0 0	Counter	External (IP2)
0 0 1	Counter	TXCA - 1X clock of channel A transmitter
0 1 0	Counter	TXCB - 1X clock of channel B transmitter
0 1 1	Counter	Crystal or external clock (X1/CLK) divided by 16
1 0 0	Timer	External (IP2)
1 0 1	Timer	External (IP2) divided by 16
1 1 0	Timer	Crystal or external clock (X1/CLK)
1 1 1	Timer	Crystal or external clock (X1/CLK) divided by 16

In the timer mode, this bit is set once each cycle of the generated square wave (every other time that the counter/timer reaches zero count). The bit is reset by a stop counter command. The command, however, does not stop the counter/timer.

ISR[2] - Channel A Change In Break
This bit, when set, indicates that the channel A receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a channel A 'reset break change interrupt' command.

ISR[1] - Channel A Receiver Ready Or FIFO Full
The function of this bit is programmed by MR1A[6]. If programmed as receiver ready, it indicates that a character has been received in channel A and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the channel A FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

ISR[0] - Channel A Transmitter Ready
This bit is a duplicate of TxRDYA (SRA[2]).

IMR - Interrupt Mask Register

The programming of this register selects which bits in the ISR cause an interrupt output. If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the programmable interrupt outputs OP3-OP7 or the reading of the ISR.

CTUR And CTLR - Counter/ Timer Registers

The CTUR and CTLR hold the eight MSBs and eight LSBs respectively of the value to be used by the counter/timer in either the counter or timer modes of operation. The minimum value which may be loaded into the CTUR/CTLR registers is 0002_{16}. Note that these registers are write-only and cannot be read by the CPU.

In the timer (programmable divider) mode, the C/T generates a square wave with a period of twice the value (in clock periods) of the CTUR and CTLR. If the value in CTUR or CTLR is changed, the current half-period will not be affected, but subsequent half periods will be. In this mode the C/T runs continuously. Receipt of a start counter command (read with A3-A0 = 1110) causes the counter to terminate the current timing cycle and to begin a new cycle using the values in CTUR and CTLR.

The counter ready status bit (ISR[3]) is set once each cycle of the square wave. The bit is reset by a stop counter command (read with A3-A0 = 1111). The command, however, does not stop the C/T. The generated square wave is output on OP3 if it is programmed to be the C/T output.

On power up and after reset, the timer/ counter runs in timer mode and can only be restarted. Because it cannot be shut off or stopped, and runs continuously in timer mode, it is recommended that at initialization, the output port (OP3) should be masked off through the OPCR[3:2] = 00 until the T/C is programmed to the desired operational state.

In the counter mode, the C/T counts down the number of pulses loaded into CTUR and CTLR by the CPU. Counting begins upon receipt of a start counter command. Upon reaching terminal count (0000_{16}), the counter ready interrupt bit (ISR[3]) is set. The counter continues counting past the terminal count until stopped by the CPU. If OP3 is programmed to be the output of the C/T, the output remains high until terminal count is reached, at which time it goes low. The output returns to the high state and ISR[3] is cleared when the counter is stopped by a stop counter command. The CPU may change the values of CTUR and CTLR at any time, but the new count becomes effective only on the next start counter command. If new values have not been loaded, the previous count values are preserved and used for the next count cycle.

In the counter mode, the current value of the upper and lower 8 bits of the counter (CTU, CTL) may be read by the CPU. It is recommended that the counter be stopped when reading to prevent potential problems which may occur if a carry from the lower 8-bits to the upper 8-bits occurs between the times that both halves of the counter are read. However, note that a subsequent start counter command will cause the counter to begin a new count cycle using the values in CTUR and CTLR.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4, 5, 6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage (except X1/CLK)		2.0			V
V_{IH}	Input high voltage (X1/CLK)		4.0			V
V_{OL}	Output low voltage	$I_{OL} = 2.4mA$			0.4	V
V_{OH}	Output high voltage (except o.c.outputs)	$I_{OH} = -400\mu A$	2.4			V
I_{IL}	Input leakage current	$V_{IN} = 0$ to V_{CC}	−10		10	μA
I_{LL}	Data bus 3-state leakage current	$V_O = 0.4$ to V_{CC}	−10		10	μA
I_{X1L}	X1/CLK low input current	$V_{IN} = 0$, X2 grounded	−4.0	−2.0	0.0	mA
		$V_{IN} = 0$, X2 floated[7]	−3.0	−1.5	0.0	mA
I_{X1H}	X1/CLK high input current	$V_{IN} = V_{CC}$, X2 grounded	−1.0	0.2	1.0	mA
		$V_{IN} = V_{CC}$, X2 floated[7]	0.0	3.5	10.0	mA
I_{X2L}	X2 low input current	$V_{IN} = 0$, X1/CLK floated	−100	−30	0.0	μA
I_{X2H}	X2 high input current	$V_{IN} = V_{cc}$, X1/CLK floated	0.0	+30	100	μA
I_{OC}	Open collector output leakage current	$V_O = 0.4$ to V_{CC}	−10		10	μA
I_{CC}	Power supply current				150	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all inputs except X1/CLK swing between 0.4V and 2.4V with a transition time of 20ns maximum. For X1/CLK this swing is between 0.4V and 4.4V. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical supply voltages, and typical processing parameters.
7. X2 is left internally floating in the 24 pin version.

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4, 5, 6, 7]

PARAMETER		LIMITS			UNIT
		Min	Typ	Max	
Reset Timing (figure 1)					
t_{RES}	RESET pulse width	1.0			μs
Bus Timing (figure 2)[8]					
t_{AS}	A0-A3 set-up time to RDN, WRN low	10			ns
t_{AH}	A0-A3 hold time from RDN, WRN high	0			ns
t_{CS}	CEN set-up time to RDN, WRN low	0			ns
t_{CH}	CEN hold time from RDN, WRN high	0			ns
t_{RW}	WRN, RDN pulse width	225			ns
t_{DD}	Data valid after RDN low			175	ns
t_{DF}	Data bus floating after RDN high			100	ns
t_{DS}	Data setsup time before WRN high	100			ns
t_{DH}	Data hold time after WRN high	20			ns
t_{RWD}	High time between READs and/or WRITEs[9,10]	200			ns

2

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C, V_{CC} = 5.0\ V \pm 5\%$[4, 5, 6, 7]

PARAMETER		LIMITS			UNIT
		Min	Typ	Max	
Port Timing (figure 3)[8]					
t_{PS}	Port input set-up time before RDN low	0			ns
t_{PH}	Port input hold time after RDN high	0			ns
t_{PD}	Port output valid after WRN high			400	ns
Interrupt Timing (figure 4)					
t_{IR}	INTRN (or OP3-OP7 when used as interrupts) negated from:				
	Read RHR (RXRDY/FFULL interrupt)			300	ns
	Write THR (TXRDY interrupt)			300	ns
	Reset command (delta break interrupt)			300	ns
	Stop C/T command (counter interrupt)			300	ns
	Read IPCR (input port change interrupt)			300	ns
	Write IMR (clear of interrupt mask bit)			300	ns
Clock Timing (figure 5)					
t_{CLK}	X1/CLK high or low time	100			ns
f_{CLK}	X1/CLK frequency	2.0	3.6864	4.0	MHz
t_{CTC}	CTCLK (IP2) high or low time	100			ns
f_{CTC}	CTCLK (IP2) frequency	0		4.0	MHz
t_{RX}	RxC high or low time	220			ns
f_{RX}	RxC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
t_{TX}	TxC high or low time	220			ns
f_{TX}	TxC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
Transmitter Timing (figure 6)					
t_{TXD}	TxD output delay from TxC low			350	ns
t_{TCS}	Output delay from TxC low to TxD data output	0		150	ns
Receiver Timing (figure 7)					
t_{RXS}	RxD data set-up time to RXC high	240			ns
t_{RXH}	RxD data hold time from RXC high	200			ns

NOTES:

4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all inputs except X1/CLK swing between 0.4V and 2.4V with a transition time of 20ns maximum. For X1/CLK this swing is between 0.4V and 4.4V. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical supply voltages, and typical processing parameters.
7. Test condition for outputs: $C_L = 150pF$, except interrupt outputs. Test condition for interrupt outputs: $C_L = 50pF$, $R_L = 2.7K$ ohm to V_{CC}.
8. Timing is illustrated and referenced to the WRN and RDN inputs. The device may also be operated with CEN as the 'strobing' input. In this case, all timing specifications apply referenced to the falling and rising edges of CEN. CEN and RDN (also CEN and WRN) are AND'ed internally. As a consequence, the signal asserted last initiates the cycle and the signal negated first terminates the cycle.
9. If CEN is used as the 'strobing' input, the parameter defines the minimum high times between one CEN and the next. The RDN signal must be negated for t_{RWD} to guarantee that any status register changes are valid.
10. Consecutive write operations to the same command register require at least three edges of the X1 clock between writes.

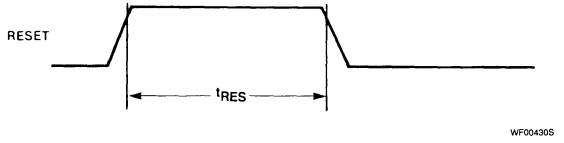


Figure 1. Reset Timing

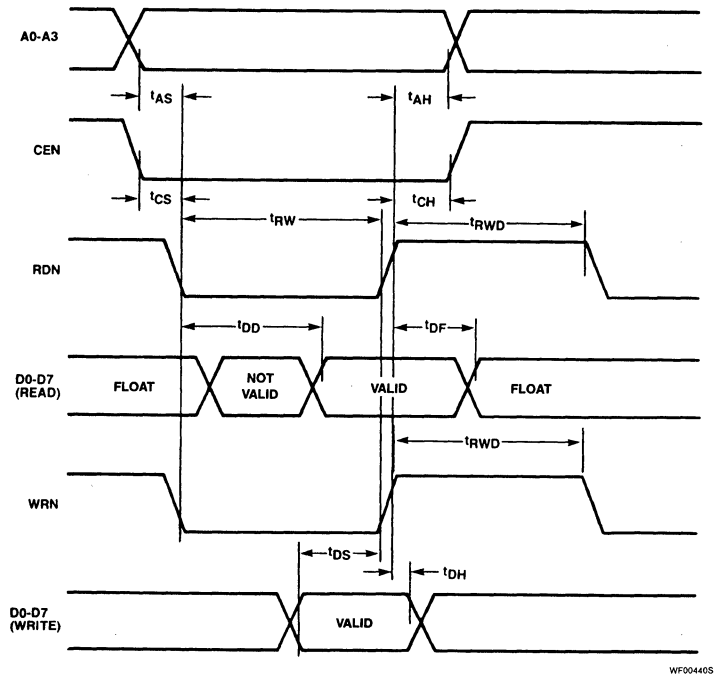


Figure 2. Bus Timing

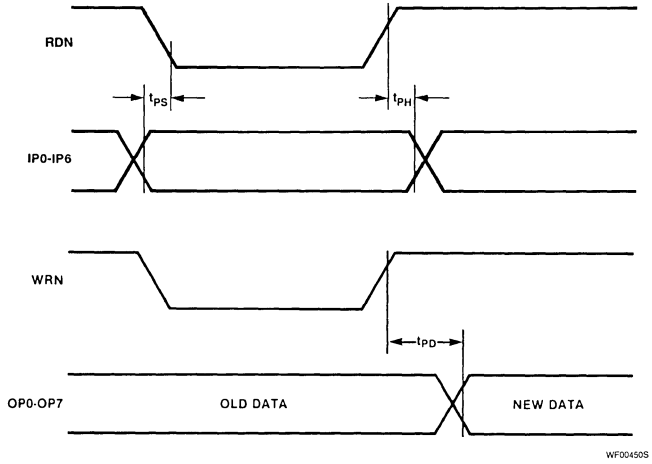
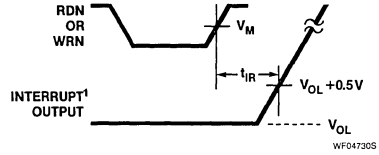


Figure 3. Port Timing



NOTES:
1. INTRN or OP3 – OP7 when used as interrupt outputs.
2. The test for open drain outputs is intended to guarantee switching of the output transistor. Measurement of this response is referenced from the midpoint of the switching signal, V_M, to a point 0.5 volts above V_{OL}. This point represents noise margin that assures true switching has occurred. Beyond this level, the effects of external circuitry and test environment are pronounced and can greatly affect the resultant measurement.

Figure 4. Interrupt Timing

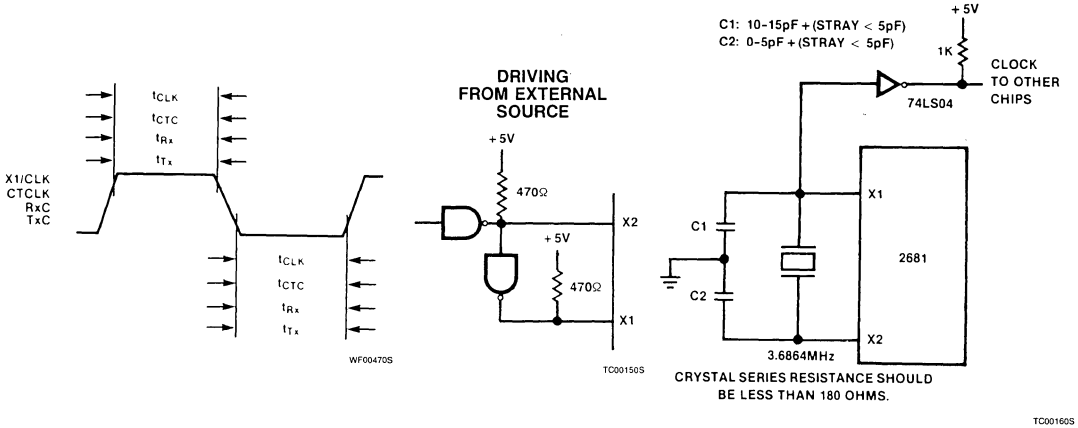


Figure 5. Clock Timing

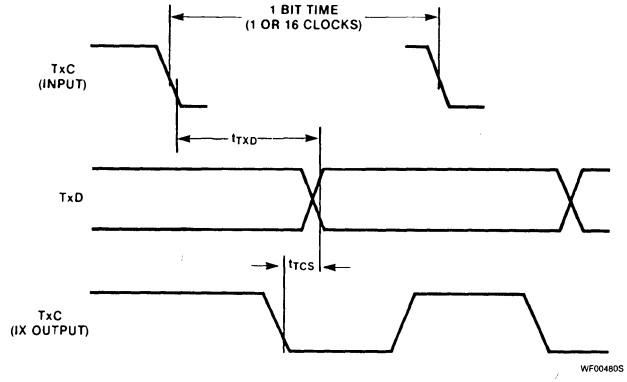


Figure 6. Transmit

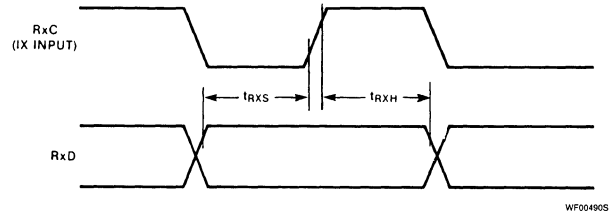
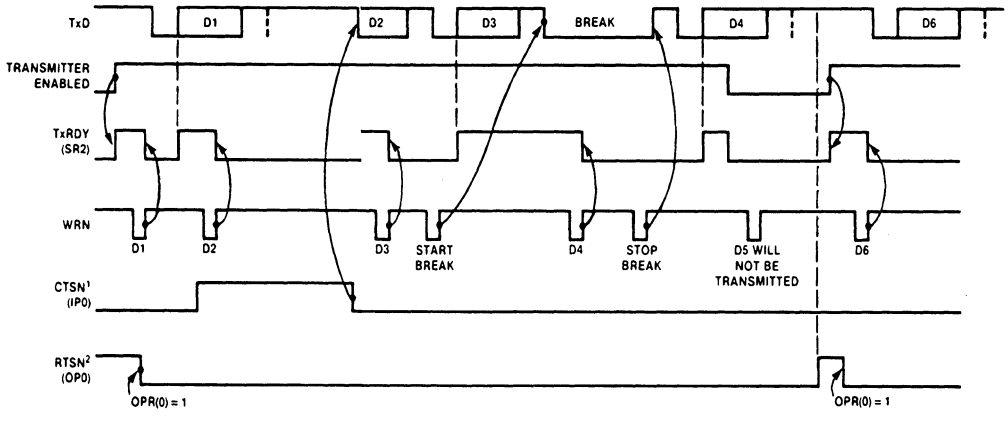
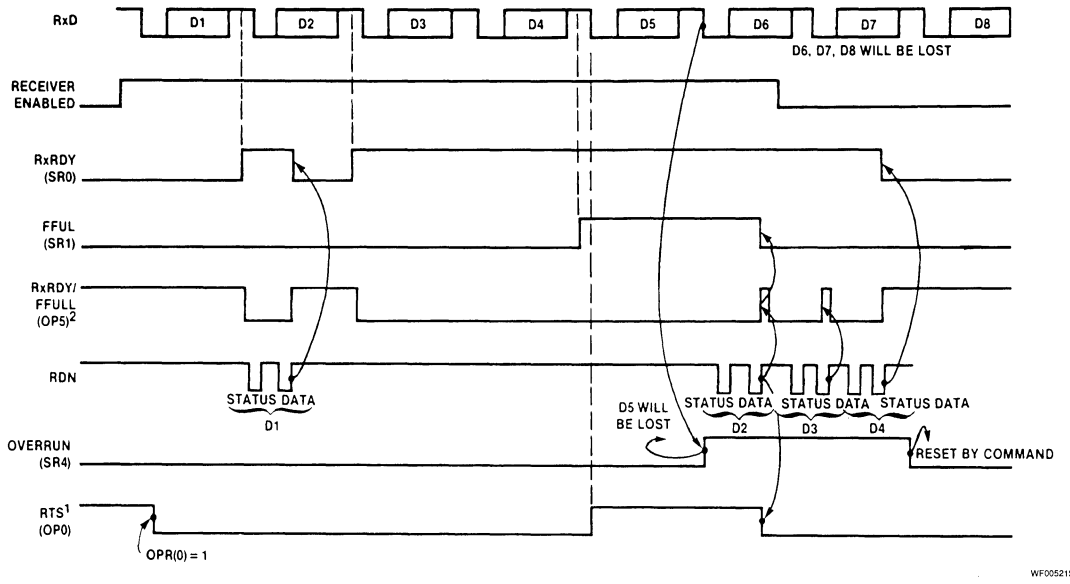


Figure 7. Receive



NOTES:
1. Timing shown for MR2(4) = 1.
2. Timing shown for MR2(5) = 1.

Figure 8. Transmitter Timing



NOTES:
1. Timing shown for MR1(7) = 1.
2. Shown for OPCR(4) = 1 and MR(6) = 0.

Figure 9. Receiver Timing

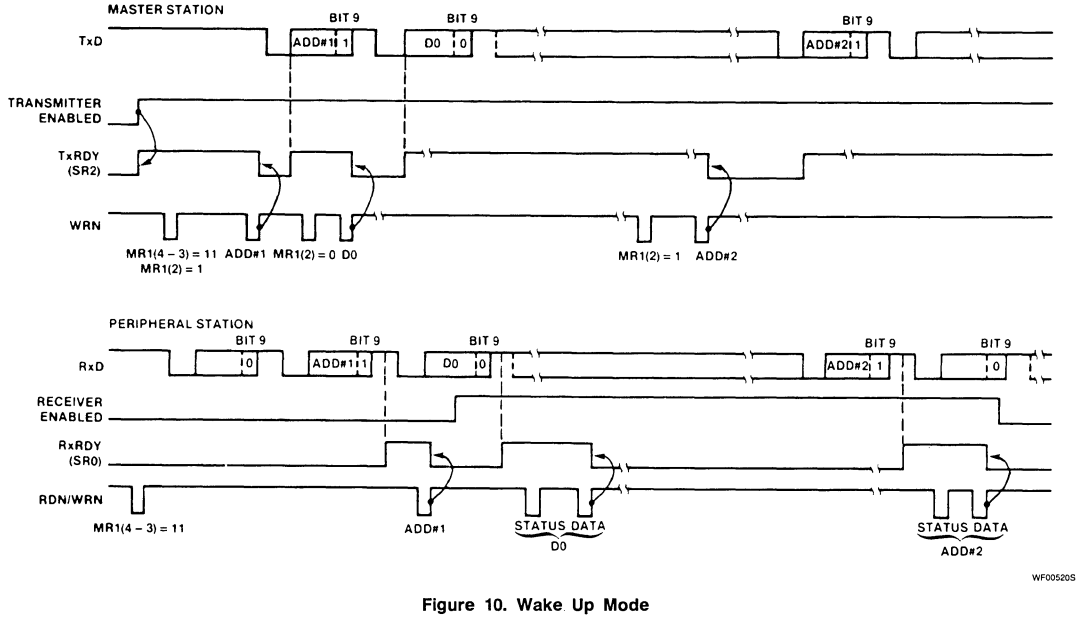


Figure 10. Wake Up Mode

Signetics

SCC2691
Universal Asynchronous Receiver/Transmitter (UART)

Objective Specification

Microprocessor Products

DESCRIPTION

The Signetics SCC2691 Universal Asynchronous Receiver/Transmitter (UART) is a single chip CMOS-LSI communications device that provides a full-duplex asynchronous receiver/transmitter in a single 24 pin DIP. It is fabricated with Signetics CMOS technology which combines the benefits of high density and low power consumption.

The operating speed of the receiver and transmitter can be selected independently as one of eighteen fixed baud rates, a 16x clock derived from a programmable counter/timer, or an external 1x or 16x clock. The baud rate generator and counter/timer can operate directly from a crystal or from external clock inputs. The ability to independently program the operating speed of the receiver and transmitter make the UART particularly attractive for dual-speed channel applications such as clustered terminal systems.

The receiver is quadruple buffered to minimize the potential of receiver overrun or to reduce interrupt overhead in interrupt driven systems. In addition, a handshaking capability is provided to disable a remote UART transmitter when the receiver buffer is full.

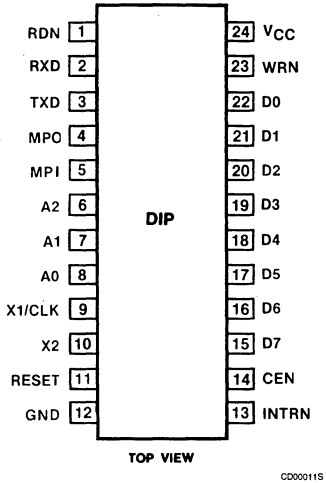
The UART provides a power down mode in which the oscillator is frozen but the register contents are stored. This results in reduced power consumption on the order of several magnitudes.

The UART is fully TTL compatible and operates from a single +5V power supply.

FEATURES

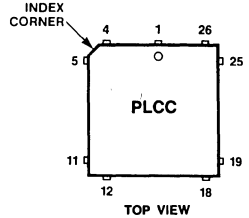
- **Full-duplex asynchronous receiver/transmitter**
- **Quadruple buffered receiver data register**
- **Programmable data format:**
  - **5 to 8 data bits plus parity**
  - **Odd, even, no parity or force parity**
  - **I, 1.5 or 2 stop bits programmable in 1/16 bit increments**
- **Baud rate for the receiver and transmitter selectable from:**
  - **18 fixed rates: 50 to 38.4K baud**
  - **One user defined rate derived from programmable timer/counter**
  - **External 1x or 16x clock**
- **Parity, framing, and overrun error detection**
- **False start bit detection**
- **Line break detection and generation**
- **Programmable channel mode**
  - **Normal (full-duplex)**
  - **Automatic echo**
  - **Local loopback**
  - **Remote loopback**
- **Multi-function programmable 16-bit counter/timer**
- **Single interrupt output with seven maskable interrupting conditions**
- **On-chip crystal oscillator**
- **Low power mode**
- **TTL compatible**
- **Single +5V power supply**

PIN CONFIGURATION



TOP VIEW

CD00011S



TOP VIEW

CD0028PS

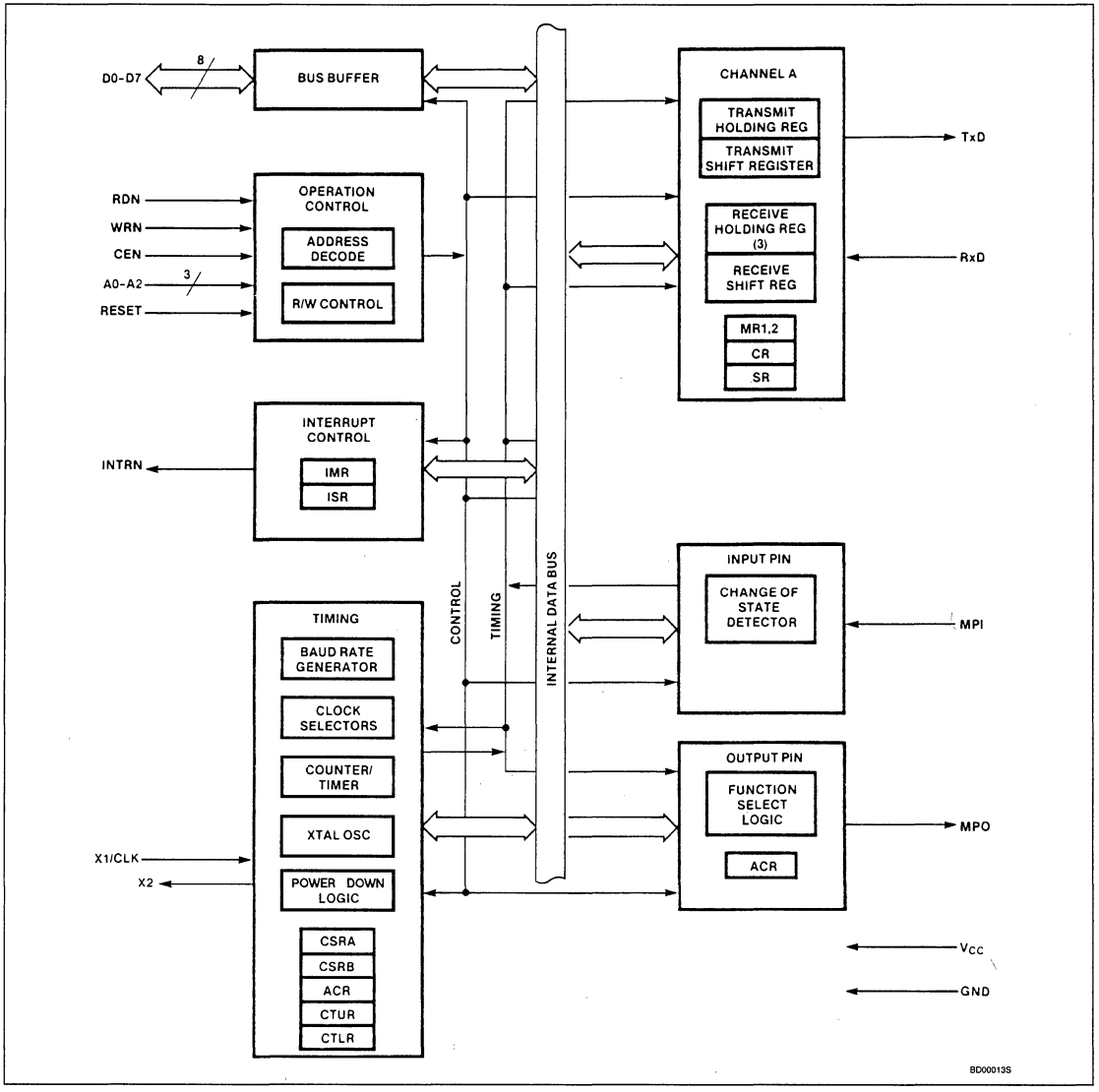
Pin	Function	Pin	Function
1	NC	15	NC
2	RDN	16	INTRN
3	RXD	17	CEN
4	TXD	18	D7
5	MPO	19	D6
6	MPI	20	D5
7	NC	21	D4
8	A2	22	D3
9	A1	23	NC
10	A0	24	D2
11	X1/CLK	25	D1
12	X2	26	D0
13	RESET	27	WRN
14	GND	28	V_{CC}

Universal Asynchronous Receiver/Transmitter (UART) SCC2691

ORDERING CODE

PACKAGES	V_{CC}=5V± 10%, T_A=0°C to 70°C
Plastic DIP	SCC2691AC1N24
Plastic LCC	SCC2691AC1A28

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
D0 – D7	22 – 15	26 – 24 22 – 18	I/O	**Data Bus:** Active high 8-bit bidirectional three-state data bus. Bit 0 is the LSB and bit 7 is the MSB. All data, command, and status transfers between the CPU and the UART take place over this bus. The direction of the transfer is controlled by the WRN and RDN inputs when the CEN input is low. When the CEN input is high, the data bus is in the three-state condition.
CEN	14	17	I	**Chip Enable:** Active low input. When low, data transfers between the CPU and the UART are enabled on D0 – D7 as controlled by the WRN, RDN, and A0 – A2 inputs. When CEN is high, the UART is effectively isolated from the data bus and D0 – D7 are placed in the three-state condition.
WRN	23	27	I	**Write Strobe:** Active low input. A low on this pin while CEN is low causes the contents of the data bus to be transferred to the register selected by A0 – A2. The transfer occurs on the trailing (rising) edge of the signal.
RDN	1	2	I	**Read Strobe:** Active low input. A low on this pin while CEN is low causes the contents of the register selected by A0 – A2 to be placed on the data bus. The read cycle begins on the leading (falling) edge of RDN.
A0 – A2	8 – 6	10 – 8	I	**Address Inputs:** Active high address inputs to select the UART registers for read/write operations.
RESET	11	13	I	**Reset:** Master reset. A high on this pin clears the status register (SR), clears the interrupt mask register (IMR), clears the auxiliary control register (ACR), and places the receiver and transmitter in the inactive state causing the TXD output to go to the marking (high) state.
INTRN	13	16	O	**Interrupt Request:** This active low output is asserted upon occurrence of one or more of seven maskable interrupting conditions. The CPU can read the interrupt status register to determine the interrupting condition(s).
X1/CLK	9	11	I	**Crystal 1:** Crystal or external clock input. When using the crystal oscillator, this pin serves as the connection for one side of the crystal. If a crystal is not used, an external clock is supplied at this input. An external clock (or crystal) is required even if the internal baud rate generator is not utilized. This clock is used to drive the internal baud rate generator, as an optional input to the timer/counter, and to provide other clocking signals required by the chip.
X2	10	12	O	**Crystal 2:** Connection for other side of crystal. If an external source is used instead of a crystal, this connection should be open.
RXD	2	3	I	**Receiver Serial Data Input:** The least significant bit is received first. If external receiver clock is specified, this input is sampled on the rising edge of the clock.
TXD	3	4	O	**Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the marking (high) condition when the transmitter is idle or disabled and when the UART is operating in local loopback mode. If external transmitter clock is specified, the data is shifted on the falling edge of the transmitter clock.
MPO	4	5	O	**Multi-Purpose Output:** One of the following functions can be selected for this output pin by programming the auxiliary control register: **RTSN** – Request to send active low output. This output is asserted and negated via the command register. By appropriate programming of the mode registers, RTSN can be programmed to be automatically reset after the character in the transmitter is completely shifted or when the receiver FIFO and shift register are full. **C/TO** – The counter/timer output. **TXC1X** – The 1X clock for the transmitter. **TXC16X** – The 16X clock for the transmitter. **RXC1X** – The 1X clock for the receiver. **RXC16X** – The 16X clock for the receiver. **TXRDY** – The transmitter holding register empty signal. Active low interrupt. **RXRDY/FFULL** – The receiver FIFO not empty/full signal. Active low interrupt.
MPI	5	6	I	**Multi-Purpose Input:** This pin can be programmed to serve as an input for one of the following functions: **GPI** – General purpose input. The current state of the pin can be determined by reading the ISR. **CTSN** – Clear-to-Send active low input. **CTCLK** – Counter/timer external clock input. **RTCLK** – Receiver and/or transmitter external clock input. This may be a 1X or 16X clock as programmed by CSR[3:0] or CSR[7:4].
V_{CC}	24	28	I	**Power Supply:** +5V supply input
GND	12	14	I	**Ground**

BLOCK DIAGRAM

As shown on the block diagram, the UART consists of: data bus buffer, interrupt control, operation control, timing, receiver and transmitter.

Data Bus Buffer

The data bus buffer provides the interface between the external and internal data buses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the UART.

Interrupt Control

A single interrupt output (INTRN) is provided which is asserted upon the occurrence of any of the following internal events:

- Transmit holding register ready
- Transmit shift register empty
- Receive holding register ready or FIFO full
- Change in break received status
- Counter reached terminal count
- Change in MPI input
- High level at the MPI input

Associated with the interrupt system are the interrupt mask register (IMR) and the interrupt status register (ISR). The IMR can be programmed to select only certain of the above conditions to cause INTRN to be asserted. The ISR can be read by the CPU to determine all currently active interrupting conditions. However, the bits of the ISR are not masked by the IMR.

Operation Control

The operation control logic receives operation commands from the CPU and generates appropriate signals to internal sections to control device operation. It contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer. The functions performed by the CPU read and write operations are shown in table 1.

Table 1. REGISTER ADDRESSING

A2	A1	A0	READ (RDN=0)	WRITE (WRN=0)
0	0	0	MR1,MR2	MR1, MR2
0	0	1	SR	CSR
0	1	0	Reserved*	CR
0	1	1	RHR	THR
1	0	0	Reserved*	ACR
1	0	1	ISR	IMR
1	1	0	CTU	CTUR
1	1	1	(CTL)	CTLR

*Reserved registers should never be read during normal operation since they are reserved for internal diagnostics.

ACR = Auxiliary control register
CR = Command register
CSR = Clock select register
CTL = Counter/timer lower
CTLR = Counter/timer lower register
CTU = Counter/timer upper
CTUR = Counter/timer upper register
MR = Mode register A
SR = Status register
THR = TX holding register

Mode registers 1 and 2 are accessed via an auxiliary pointer. The pointer is set to MR1 by RESET or by issuing a reset pointer command via the command register. Any read or write of the mode register while the pointer is at MR1 switches the pointer to MR2. The pointer then remains at MR2 so that subsequent accesses are to MR2, unless the pointer is reset to MR1 as described above.

Timing Circuits

The timing block consists of a crystal oscillator, a baud rate generator, a programmable 16-bit counter/timer, and two clock selectors.

The crystal oscillator operates directly from a 3.6864MHz crystal connected across the X1/CLK and X2 inputs with a minimum of external components. If an external clock of the appropriate frequency is available, it may be connected to X1/CLK. If an external clock is used instead of a crystal, X1/CLK is driven using a configuration similar to the one in figure 5. However, the input high voltage must be capable of attaining 4.4V. The clock serves as the basic timing reference for the baud rate generator (BRG), the counter/timer, and other internal circuits. A clock frequency, within the limits specified in the electrical specifications, must be supplied even if the internal BRG is not used.

The baud rate generator operates from the oscillator or external clock input and is capable of generating 18 commonly used data communications baud rates ranging from 50 to 38.4K baud. Thirteen of these are available simultaneously for use by the receiver and transmitter. Eight are fixed, and one of two sets of five can be selected by programming ACR[7]. The clock outputs from the BRG are at 16X the actual baud rate. The counter/timer can be used as a timer to produce a 16X clock for any other baud rate by counting down the crystal clock or an external clock. The clock selectors allow the independent selection by the receiver and transmitter of any of these baud rates or an external timing signal.

The C/T operation is programmed by ACR[6:4]. One of eight timing sources can be used as the input to the C/T. The output of the C/T is available to the clock selectors and can also be programmed by ACR[2:0], to be output on the MPO pin.

In the timer mode, the C/T generates a square wave whose period is twice the number of clock periods loaded into the C/T upper and lower registers. The counter ready bit in the ISR is set once each cycle of the square wave. If the value in CTUR or CTLR is changed, the current half period will not be affected, but subsequent half periods will be affected. In this mode the C/T runs continuously and does not recognize the stop counter command (the command only resets the counter ready bit in the ISR). Receipt of a start C/T command causes the counter to terminate the current timing cycle and to begin a new cycle using the values in CTUR and CTLR.

In the counter mode, the C/T counts down the number of pulses loaded into CTUR and CTLR. Counting begins upon receipt of a start C/T command. Upon reaching terminal count, the counter ready bit in the ISR is set. The counter continues counting past the terminal count until stopped by the CPU. If MPO is programmed to be the output of the C/T, the output remains high until terminal count is reached, at which time it goes low. The output returns to the high state and the counter ready bit is cleared when the counter is stopped by a stop counter command. The CPU may change the values of CTUR and CTLR at any time, but the new count becomes effective only on the next start counter command following a stop counter command. If new values have not been loaded, the previous count values are preserved and used for the next count cycle.

In the counter mode, the current value of the upper and lower 8 bits of the counter may be read by the CPU. It is recommended that the counter be stopped when reading to prevent potential problems which may occur if a carry from the lower 8 bits to the upper 8 bits occurs between the times that both halves of the counter are read. However a subsequent start counter command causes the counter to begin a new count cycle using the values in CTUR and CTLR.

Receiver and Transmitter

The UART is a full duplex asynchronous receiver/transmitter. The operating frequency for the receiver and transmitter can be selected independently from the baud rate generator, the counter/timer, or from an external input. Registers associated with the communications channel are the mode registers (MR1 and MR2), the clock select register (CSR), the command register (CR), the status register (SR), the transmit holding register (THR), and the receive holding register (RHR).

Transmitter

The transmitter accepts parallel data from the CPU and converts it to a serial bit stream on

the TXD output pin. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The least significant bit is sent first. Following the transmission of the stop bits, if a new character is not available in the THR, the TXD output remains high and the TXEMT bit in the SR will be set to 1. Transmission resumes and the TXEMT bit is cleared when the CPU loads a new character into the THR. In the 16X clock mode, this also resynchronizes the internal 1X transmitter clock so that transmission of the new character begins with minimum delay.

The transmitter can be forced to send a break (continuous low condition) by issuing a start break command via the CR. The break is terminated by a stop break command.

If the transmitter is disabled, it continues operating until the character currently being transmitted and the character in the THR, if any, are completely sent out. Characters cannot be loaded into the THR while the transmitter is disabled.

Receiver

The receiver accepts serial data on the RxD pin, converts the serial input to parallel format, checks for start bit, stop bit, parity bit (if any), or break condition, and presents the assembled character to the CPU. The receiver looks for a high to low (mark to space) transition of the start bit on the RXD input pin. If a transition is detected, the state of the RXD pin is sampled again each 16X clock for 7-1/2 clocks (16X clock mode) or at the next rising edge of the bit time clock (1X clock mode). If RXD is sampled high, the start bit is invalid and the search for a valid start bit begins again. If RXD is still low, a valid start bit is assumed and the receiver continues to sample the input at one bit time intervals at the theoretical center of the bit, until the proper number of data bits and the parity bit (if any) have been assembled, and one stop bit has been detected. The least significant bit is received first. The data is then transferred to the RHR and the RXRDY bit in the SR is set to a 1. If the character length is less than eight bits, the most significant unused bits in the RHR are set to zero.

After the stop bit is detected, the receiver will immediately look for the next start bit. However, if a non-zero character was received without a stop bit (i.e. framing error) and RXD remains low for one half of the bit period after the stop bit was sampled, then the receiver operates as if a new start bit transition had been detected at that point (one half bit time after the stop bit was sampled). The parity error, framing error and overrun error (if any) are strobed into the SR at the received character boundary, before the RXRDY status bit is set.

If a break condition is detected (RxD is low for the entire character including the stop bit) only one character consisting of all zeros will be loaded into the FIFO and the received break bit in the SR is set to 1. The RXD input must return to a high condition for two successive clock edges of the 1X clock (internal or external) before a search for the next start bit begins.

RECEIVER FIFO

The RHR consists of a first-in-first-out (FIFO) queue with a capacity of three characters. Data is loaded from the receive shift register into the top-most empty position of the FIFO. The RXRDY bit in the status register (SR) is set whenever one or more characters are available to be read, and a FFULL status bit is set if all three queue positions are filled with data. Either of these bits can be selected to cause an interrupt. A read of the RHR outputs the data at the top of the FIFO. After the read cycle, the data FIFO and its associated status bits are 'popped' thus emptying a FIFO position for new data.

In addition to the data word, three status bits (parity error, framing error, and received break) are appended to each data character in the FIFO. Status can be provided in two ways, as programmed by the error mode control bit in the mode register. In the character mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the block mode, the status provided in the SR for these three bits is the logical OR of the status for all characters coming to the top of the FIFO since the last reset error command was issued. In either mode, reading the SR does not affect the FIFO. The FIFO is 'popped' only when the RHR is read. Therefore, the SR should be read prior to reading the corresponding data character.

If the FIFO is full when a new character is received, that character is held in the receive shift register until a FIFO position is available. If an additional character is received while this state exists, the contents of the FIFO are not affected: the character previously in the shift register is lost and the overrun error status bit (SR[4]) will be set upon receipt of the start bit of the new (overrunning) character.

WAKE UP MODE

In addition to the normal transmitter and receiver operation described above, the UART incorporates a special mode which provides automatic wake-up of the receiver through address frame recognition for multiprocessor communications. This mode is selected by programming bits MR1[4:3] to '11'.

In this mode of operation, a 'master' station transmits an address character followed by data characters for the addressed 'slave' station. The slave stations, whose receivers are normally disabled, examine the received data stream and 'wake-up' the CPU (by setting RXRDY) only upon receipt of an address character. The CPU compares the received address to its station address and enables the receiver if it wishes to receive the subsequent data characters. Upon receipt of another address character, the CPU may disable the receiver to initiate the process again.

A transmitted character consists of a start bit, the programmed number of data bits, an address/data (A/D) bit, and the programmed number of stop bits. The polarity of the transmitted A/D bit is selected by the CPU by programming bit MR1[2]: MR1[2] = 0 transmits a zero in the A/D bit position which identifies the corresponding data bits as data, while MR1[2] = 1 transmits a one in the A/D bit position which identifies the corresponding data bits as an address. The CPU should program the mode register prior to loading the corresponding data bits into the THR.

While in this mode, the receiver continuously looks at the received data stream, whether it is enabled or disabled. If disabled, it sets the RXRDY status bit and loads the character into the RHR FIFO if the received A/D bit is a one, but discards the received character if the received A/D bit is a zero. If enabled, all received characters are transferred to the CPU via the RHR. In either case, the data bits are loaded into the data FIFO while the A/D bit is loaded into the status FIFO position normally used for parity error (SR[5]). Framing error, overrun error, and break detect operate normally whether or not the receiver is enabled.

MULTI-PURPOSE INPUT PIN

The MPI pin can be programmed as an input to one of several UART circuits. The function of the pin is selected by programming the appropriate control register (MR2[4], ACR[6:4], CSR[7:4, 3:0]). Only one of the functions may be selected at any given time. If CTS or GPI is selected, a change of state detector provided with the pin is activated. A high-to-low or low-to-high transition of the inputs lasting longer than 25 – 50μsec sets the MPI change-of-state bit in the interrupt status register. The bit is cleared via a command. The change of state can be programmed to generate an interrupt to the CPU by setting the corresponding bit in the interrupt mask register.

The input port pulse detection circuitry uses a 38.4KHz sampling clock derived from one of the baud rate generator taps. This produces a sampling period of slightly more than 25μsec (assuming a 3.6864MHz oscillator input). The detection circuitry, in order to guarantee that a true change in level has occurred, requires two successive samples at the new logic level be observed. As a consequence, the minimum duration of the signal change is 25μsec if the transition occurs coincident with the first sample pulse. The 50μsec time refers to the condition where the change of state is just missed and the first change of state is not detected until after an additional 25μsec.

MULTI-PURPOSE OUTPUT PIN

This pin can be programmed to serve as a request-to-send output, the counter/timer output, the output for the 1X or 16X transmitter or receiver clocks, the TXRDY output or the RXRDY/FFULL output (see ACR [2:0] – MPO Output Select).

REGISTERS

The operation of the UART is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. Addressing of the registers is as described in table 1.

The contents of certain control registers are initialized to zero on RESET (see Reset pin description). Care should be exercised if the contents of a register are changed during operation, since certain changes may cause operational problems – e.g., changing the number of bits per character while the transmitter is active may cause the transmission of an incorrect character. The contents of the MR, the CSR, and the ACR should only be changed while the receiver(s) and transmitter(s) are disabled, and certain changes to the ACR should only be made while the C/T is stopped.

The bit formats of the UART registers are depicted in table 2.

MR1 – Mode Register 1

MR1 is accessed when the MR pointer points to MR1. The pointer is set to MR1 by RESET or by a set pointer command applied via CR. After reading or writing MR1, the pointers are set at MR2.

MR1[7] – Receiver Request-to-Send Control

This bit controls the deactivation of the RTSN output (MPO) by the receiver. This output is manually asserted and negated by commands applied via the command register. MR1[7] = 1 causes RTSN to be automatically negated upon receipt of a valid start bit if the receiver FIFO is full. RTSN is reasserted when an empty FIFO position is available. This feature can be used to prevent overrun in the receiver by using the RTSN output signal to control the CTS input of the transmitting device.

MR1[6] – Receiver Interrupt Select

This bit selects either the receiver ready status (RXRDY) or the FIFO full status (FFULL) to be used for CPU interrupts.

MR1[5] – Error Mode Select

This bit selects the operating mode of the three FIFOed status bits (FE, PE, received break). In the character mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the block mode, the status provided in the SR for these bits is the accumulation (logical OR) of the status for all characters coming to the top of the FIFO since the last reset error command was issued.

MR1[4:3] – Parity Mode Select

If with parity or force parity is selected , a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR1[4:3] = 11 selects the channel to operate in the special wake up mode.

MR1[2] – Parity Type Select

This bit selects the parity type (odd or even) if the with parity mode is programmed by MR1[4:3], and the polarity of the forced parity bit if the force parity mode is programmed. It has no effect if the no parity mode is programmed. In the special wake up mode, it selects the polarity of the A/D bit.

MR1[1:0] – Bits per Character Select

This field selects the number of data bits per character to be transmitted and received. The character length does not include the start, parity, and stop bits.

MR2 Mode Register 2

MR2 is accessed when the channel MR pointer points to MR2, which occurs after any access to MR1. Accesses to MR2 do not change the pointer.

MR2[7:6] – Mode Select

The UART can operate in one of four modes: MR2[7:6] = 00 is the normal mode, with the transmitter and receiver operating independently. MR2[7:6] = 01 places the channel in the automatic echo mode, which automatically retransmits the received data. The following conditions are true while in automatic echo mode:

1. Received data is reclocked and retransmitted on the TXD output.
2. The receive clock is used for the transmitter.
3. The receiver must be enabled, but the transmitter need not be enabled.
4. The TXRDY and TXEMT status bits are inactive.
5. The received parity is checked, but is not regenerated for transmission, i.e., transmitted parity bit is as received.
6. Character framing is checked, but the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.
8. CPU to receiver communication continues normally, but the CPU to transmitter link is disabled.

Two diagnostic modes can also be selected. MR2[7:6] = 10 selects local loopback mode. In this mode:

1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver.
3. The TXD output is held high.
4. The RXD input is ignored.
5. The transmitter must be enabled, but the receiver need not be enabled.
6. CPU to transmitter and receiver communications continue normally.

The second diagnostic mode is the remote loopback mode, selected by MR2[7:6] = 11. In this mode:

1. Received data is reclocked and retransmitted on the TXD output.
2. The receive clock is used for the transmitter.
3. Received data is not sent to the local CPU, and the error status conditions are inactive.
4. The received parity is not checked and is not regenerated for transmission, i.e., the transmitted parity bit is as received.
5. The receiver must be enabled, but the transmitter need not be enabled.
6. Character framing is not checked, and the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.

When switching in and out of the various modes, the selected mode is activated immediately upon mode selection, even if this occurs in the middle of a received or transmitted character. Likewise, if a mode is deselected, the device will switch out of the mode immediately. An exception to this is switching out of auto echo or remote loopback modes; if the deselection occurs just after the receiver has sampled the stop bit (indicated to be in autoecho by assertion of RXRDY), and the transmitter is enabled, the transmitter will remain in auto echo mode until one full stop bit has been retransmitted.

MR2[5] – Transmitter Request-to-Send Control

This bit controls the deactivation of the RTSN output (MPO) by the transmitter. This output

is manually asserted and negated by appropriate commands issued via the command register. MR2[5] = 1 causes RTSN to be reset automatically one bit time after the characters in the transmit shift register and in the THR (if any) are completely transmitted; includes the programmed number of stop bits if the transmitter is not enabled. This feature can be used to automatically terminate the transmission of a message as follows:

1. Program auto-reset mode: MR2[5] = 1.
2. Enable transmitter.
3. Assert RTSN via command.
4. Send message.
5. Verify the next to last character of the message is being sent by waiting until transmitter ready is asserted. Disable transmitter after the last character is loaded into the THR.
6. The last character will be transmitted and RTSN will be reset one bit time after the last stop bit.

MR2[4] – Clear-to-Send Control
The state of this bit determines if the CTSN input (MPI) controls the operation of the transmitter. If this bit is 0, CTSN has no effect on the transmitter. If this bit is a 1, the transmitter checks the state of CTSN each time it is ready to send a character. If it is asserted (low), the character is transmitted. If it is negated (high), the TXD output remains in the marking state and the transmission is delayed until CTSN goes low. Changes in CTSN while a character is being transmitted do not affect the transmission of that character. This feature can be used to prevent overrun of a remote receiver.

MR2[3:0] – Stop Bit Length Select
This field programs the length of the stop bit appended to the transmitted character. Stop bit lengths of 9/16 to 1 and 1-9/16 to 2 bits, in increments of 1/16 bit, can be programmed for character lengths of 6, 7, and 8 bits. For a character length of 5 bits, 1-1/16 to 2 stop bits can be programmed in increments of 1/16 bit. In all cases, the receiver only checks for a mark condition at the center of the first stop bit position (one bit time after the last data bit, or after the parity bit if parity is enabled). If an external 1X clock is used for the transmitter, MR2[3] = 0 selects one stop bit and MR2[3] = 1 selects two stop bits to be transmitted.

CSR – Clock Select Register

CSR[7:4] – Receiver Clock Select
When using a 3.6864MHz crystal or external clock input, this field selects the baud rate clock for the receiver as shown in table 3.

Table 2. REGISTER BIT FORMATS

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RX RTS CONTROL	RX INT SELECT	ERROR MODE	PARITY MODE		PARITY TYPE	BITS PER CHAR	
MR1	0=no 1=yes	0=RXRDY 1=FFULL	0=char 1=block	00=with parity 01=force parity 10=no parity 11=special mode		0=even 1=odd	00=5 01=6 10=7 11=8	

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	CHANNEL MODE		Tx RTS CONTROL	CTS ENABLE Tx	STOP BIT LENGTH*			
MR2	00=Normal 01=Auto echo 10=Local loop 11=Remote loop		0=no 1=yes	0=no 1=yes	0=0.563 1=0.625 2=0.688 3=0.750	4=0.813 5=0.875 6=0.938 7=1.000	8=1.563 9=1.625 A=1.688 B=1.750	C=1.813 D=1.875 E=1.938 F=2.000

*Add 0.5 to values shown for 0 – 7, if channel is programmed for 5 bits/char.

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RECEIVER CLOCK SELECT				TRANSMITTER CLOCK SELECT			
CSR	See Text				See Text			

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	MISCELLANEOUS COMMANDS				DISABLE Tx	ENABLE Tx	DISABLE Rx	ENABLE Rx
CR	See Text				0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes

Table 2. REGISTER BIT FORMATS (Continued)

SR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
RECEIVED BREAK	FRAMING ERROR	PARITY ERROR	OVERRUN ERROR	TXEMT	TXRDY	FFULL	RXRDY
0=no 1=yes *	0=no 1=yes *	0=no 1=yes *	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes

*These status bits are appended to the corresponding data character in the receive FIFO. A read of the status register provides these bits [7:5] from the top of the FIFO together with bits [4:0]. These bits are cleared by a reset error status command. In character mode they are reset when the corresponding data character is read from the FIFO.

ACR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
BRG SET SELECT	COUNTER/TIMER MODE AND SOURCE			POWER DOWN MODE	MPO PIN FUNCTION SELECT		
0=set1 1=set2	See Text			0 = on 1 = off	000=RTSN 001=C/TO 010=TXC(1X) 011=TXC(16X)	100=RXC(1X)) 101=RXC(16X) 110=TXRDY 111=RXRDY/FFULL	

ISR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
MPI PIN CHANGE	MPI PIN CURRENT STATE		COUNTER READY	DELTA BREAK	RXRDY/ FFULL	TXEMT	TXRDY
0=no 1=yes	0=low 1=high	not used	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes

IMR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
MPI CHANGE INT	MPI LEVEL INT		COUNTER READY INT	DELTA BREAK INT	RXRDY/ FFULL INT	TXEMT INT	TXRDY INT
0=off 1=on	0=off 1=on	not used	0=off 1=on	0=off 1=on	0=off 1=on	0=off 1=on	0=off 1=on

CTUR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
C/T[15]	C/T[14]	C/T[13]	C/T[12]	C/T[11]	C/T[10]	C/T[9]	C/T[8]

CTLR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
C/T[7]	C/T[6]	C/T[5]	C/T[4]	C/T[3]	C/T[2]	C/T[1]	C/T[0]

2

Table 3. BAUD RATE

CSR[3:0]/ [7:4]	ACR[7]=0	ACR[7]=1
0 0 0 0	50	75
0 0 0 1	110	110
0 0 1 0	134.5	134.5
0 0 1 1	200	150
0 1 0 0	300	300
0 1 0 1	600	600
0 1 1 0	1,200	1,200
0 1 1 1	1,050	2,000
1 0 0 0	2,400	2,400
1 0 0 1	4,800	4,800
1 0 1 0	7,200	1,800
1 0 1 1	9,600	9,600
1 1 0 0	38.4K	19.2K
1 1 0 1	Timer	Timer
1 1 1 0	MPI – 16X	MPI – 16X
1 1 1 1	MPI – 1X	MPI – 1X

The receiver clock is always a 16X clock, except for CSR[7:4]=1111.

CSR[3:0] – Transmitter Clock Select
This field selects the baud rate clock for the transmitter. The field definition is as shown in table 3.

CR – Command Register

CR is used to write commands to the UART. Multiple commands can be specified in a single write to CR as long as the commands are non-conflicting, e.g., the enable transmitter and reset transmitter commands cannot be specified in a single command word.

CR[7:4] – Miscellaneous Commands
The encoded value of this field may be used to specify a single command as follows:

0000 No command.

0001 Reset MR pointer. Causes the MR pointer to point to MR1.

0010 Reset receiver. Resets the receiver as if a hardware reset had been applied. The receiver is disabled and the FIFO is flushed.

0011 Reset transmitter. Resets the transmitter as if a hardware reset had been applied.

0100 Reset error status. Clears the received break, parity error, framing error, and overrun error bits in the status register (SR[7:4]). Used in character mode to clear OE status (although RB, PE, and FE bits will also be cleared), and in block mode to clear all error status after a block of data has been received.

0101 Reset break change interrupt. Causes the break detect change bit in the interrupt status register (ISR[3]) to be cleared to zero.

0110 Start break. Forces the TXD output low (spacing). If the transmitter is empty, the start of the break condition will be delayed up to two bit times. If the transmitter is active, the break begins when transmission of the character is completed. If a character is in the THR, the start of break is delayed until that character or any others loaded after it has been transmitted (TXEMT must be true before break begins). The transmitter must be enabled to start a break.

0111 Stop break. The TXD line will go high (marking) within two bit times. TXD will remain high for one bit time before the next character, if any, is transmitted.

1000 Start C/T. In counter or timer modes, causes the contents of CTUR/ CTLR to be preset into the counter/timer and starts the counting cycle. In timer mode, any counting cycle in progress when the command is issued is terminated. In counter mode, has no effect unless a stop C/T command was issued previously.

1001 Stop counter. In counter mode, stops operation of the counter/timer, resets the counter ready bit in the ISR, and forces the MPO output high if it is programmed to be the output of the C/T. In timer mode, resets the counter ready bit in the ISR but has no affect on the counter/timer itself or on the MPO output.

1010 Assert RTSN. Causes the RTSN output to be asserted (low).

1011 Negate RTSN. Causes the RTSN output to be negated (high).

1100 Reset MPI change interrupt. Causes the MPI change bit in the interrupt status register (ISR[7]) to be cleared to zero.

1101 Reserved.

111x Reserved.

CR[3] – Disable Transmitter
This command terminates transmitter operation and resets the TXRDY and TXEMT status bits. However, if a character is being transmitted or if a character is in the THR when the transmitter is disabled, the transmission of the character(s) is completed before assuming the inactive state.

CR[2] – Enable Transmitter
Enables operation of the channel A transmitter. The TXRDY status bit will be asserted.

CR[1] – Disable Receiver
This command terminates operation of the receiver immediately – a character being received will be lost. The command has no effect on the receiver status bits or any other control registers. If the special wakeup mode is programmed, the receiver operates even if it is disabled (see Wakeup Mode).

CR[0] – Enable Receiver
Enables operation of the receiver. If not in the special wakeup mode, this also forces the receiver into the search for start bit state.

SR – Channel Status Register

SR[7] – Received Break
This bit indicates that an all zero character of the programmed length has been received without a stop bit. Only a single FIFO position is occupied when a break is received; further entries to the FIFO are inhibited until the RXD line returns to the marking state for at least one half bit time (two successive edges of the internal or external 1x clock).

When this bit is set, the change in break bit in the ISR (ISR[3]) is set. ISR[3] is also set when the end of the break condition, as defined above, is detected.

The break detect circuitry is capable of detecting breaks that originate in the middle of a received character. However, if a break begins in the middle of a character, it must last until the end of the next character time in order for it to be detected.

SR[6] – Framing Error (FE)
This bit, when set, indicates that a stop bit was not detected when the corresponding data character in the FIFO was received. The stop bit check is made in the middle of the first stop bit position.

SR[5] – Parity Error (PE)
This bit is set when the with parity or force parity mode is programmed and the corresponding character in the FIFO was received with incorrect parity.

In the special wakeup mode, the parity error bit stores the received A/D bit.

SR[4] – Overrun Error (OE)
This bit, when set, indicates that one or more characters in the received data stream have been lost. It is set upon receipt of a new character when the FIFO is full and a character is already in the receive shift register waiting for an empty FIFO position. When this occurs, the character in the receive shift register (and its break detect, parity error and framing error status, if any) is lost.

This bit is cleared by a reset error status command.

SR[3] – Transmitter Empty (TXEMT)
This bit will be set when the transmitter underruns, i.e., both the transmit holding register (THR) and the transmit shift register are empty. However, this bit is not set until one character has been transmitted. It is set after transmission of the last stop bit of a

character, if no character is in the THR awaiting transmission. It is reset when the THR is loaded by the CPU, or when the transmitter is disabled.

SR[2] – Transmitter Ready (TXRDY)
This bit, when set, indicates that the THR is empty and ready to be loaded with a character. This bit is cleared when the THR is loaded by the CPU and is set when the character is transferred to the transmit shift register. TXRDY is reset when the transmitter is disabled and is set when the transmitter is first enabled, e.g., characters loaded in the THR while the transmitter is disabled will not be transmitted.

SR[1] – FIFO Full (FFULL)
This bit is set when a character is transferred from the receive shift register to the receive FIFO and the transfer causes the FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the FIFO and there is no character in the receive shift register. If a character is waiting in the receive shift register because the FIFO is full, FFULL will be reset by the CPU read and then set by the transfer of the character to the FIFO, which causes all three FIFO positions to be occupied.

SR[0] – Receiver Ready (RXRDY)
This bit indicates that a character has been received and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR, and no more characters are in the FIFO.

ACR – Auxiliary Control Register

ACR[7] – Baud Rate Generator Set Select
This bit selects one of two sets of baud rates generated by the BRG.

Set 1: 50, 110, 134.5, 200, 300, 600, 1.05K, 1.2K, 2.4K, 4.8K, 7.2K, 9.6K, and 38.4K baud.

Set 2: 75, 110, 134.5, 150, 300, 600, 1.2K, 1.8K, 2.0K, 2.4K, 4.8K, 9.6K, and 19.2K baud.

The selected set of rates is available for use by the receiver and transmitter.

ACR[6:4] – Counter/Timer Mode and Clock Source Select
This field selects the operating mode of the counter/timer and its clock source as follows:

ACR[6:4]	Mode	Clock Source
0 0 0	Counter	MPI pin
0 0 1	Counter	MPI pin divided by 16
0 1 0	Counter	TXC – 1X clock of the transmitter
0 1 1	Counter	Crystal or external clock (X1/CLK) divided by 16
1 0 0	Timer	MPI pin
1 0 1	Timer	MPI pin divided by 16
1 1 0	Timer	Crystal or external clock (X1/CLK)
1 1 1	Timer	Crystal or external clock (X1/CLK) divided by 16

ACR[3] – Power Down Mode Select
This bit, when set to zero, selects the power down mode. In this mode, the 2691 oscillator is stopped and all functions requiring this clock are suspended. The contents of all registers are saved. It is recommended that the transmitter and receiver be disabled prior to placing the 2691 in this mode. Note that this bit must be set to a logic 1 after power up.

ACR[2:0] – MPO Output Select
This field programs the MPO output pin to provide one of the following:

000 Request to send active low output (RTSN). This output is asserted and negated via the command register. Mode RTSN can be programmed to be automatically reset after the character in the transmitter is completely shifted out or when the receiver FIFO and receiver shift register are full using MR2[5] and MR1[7], respectively.

001 The counter/timer output. In the timer mode, this output is a square wave with a period of twice the value (in clock periods) of the contents of the CTUR and CTLR. In the counter mode, the output remains high until the terminal count is reached, at which time it goes low. The output returns to the high state when the counter is stopped by a stop counter command.

010 The 1X clock for the transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a non-synchronized 1X clock is output.

011 The 16X clock for the transmitter. This is the clock selected by CSR[3:0], and is a 1X clock if CSR[3:0] = 1111.

100 The 1X clock for the receiver, which is the clock that samples the received data. If data is not being received, a non-synchronized 1X clock is output.

101 The 16X clock for the receiver. This is the clock selected by CSR[7:4], and is a 1X clock if CSR[7:4] = 1111.

110 The transmitter register empty signal, which is the complement of SR[2]. Active low output.

111 The receiver ready or FIFO full signal (complement of ISR[2]). Active low output.

ISR – Interrupt Status Register

This register provides the status of all potential interrupt sources. The contents of this register are masked by the interrupt mask register (IMR). If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output is asserted (low). If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no affect on the INTRN output. Note that the IMR does not mask the reading of the ISR – the true status is provided regardless of the contents of the IMR.

ISR[7] – MPI Change of State
This bit is set when a change of state occurs at the MPI input pin. It is reset by a reset MPI change interrupt command.

ISR[6] – MPI Current State
This bit provides the current state of the MPI pin. The information is unlatched and reflects the state of the pin at the time the ISR is read.

ISR[4] – Counter Ready
In the counter mode of operation, this bit is set when the counter reaches terminal count and is reset when the counter is stopped by a stop counter command. It is initialized to '0' when the chip is reset.

In the timer mode, this bit is set once each cycle of the generated square wave (every other time the C/T reaches zero count). The bit is reset by a stop counter command. The command, however, does not stop the C/T.

ISR[3] – Change in Break
This bit, when set, indicates that the receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a reset break change interrupt command.

ISR[2] – Receiver Ready or FIFO Full
The function of this bit is programmed by MR1[6]. If programmed as receiver ready, it indicates that a character has been received and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the receiver FIFO. If the FIFO contains more characters, the bit will be set again after the FIFO is read. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when FIFO is read and there is no character in the receiver shift register. If there

is a character waiting in the receive shift register because the FIFO is full, the bit is set again when the waiting character is transferred into the FIFO.

ISR[1] – Transmitter Empty
This bit is a duplicate of TXEMT (SR[3]).

ISR[0] – Transmitter Ready
This bit is a duplicate of TXRDY (SR[2]).

IMR – Interrupt Mask Register

The programming of this register selects which bits in the ISR cause an interrupt output. If a bit in the ISR is a '1' and the corresponding bit in the IMR is a '1', the INTRN output is asserted (low). If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask reading of the ISR.

CTUR AND CTLR – Counter/ Timer Registers

The CTUR and CTLR hold the eight MSB's and eight LSB's, respectively, the value to be used by the counter/timer in either the counter or timer modes of operation. The minimum value which may be loaded is 0002_{16}.

In the timer (programmable divider) mode, the C/T generates a square wave whose period is twice the value (in clock periods) of the CTUR and CTLR. If the value in CTUR or CTLR is changed, the current half-period will not be affected, but subsequent half-periods will be.

The counter ready status bit (ISR[4]) is set once each cycle of the square wave. The bit is reset by a stop counter command. The command, however, does not stop the C/T. The generated square wave is output on MPO if it is programmed to be the C/T output.

In the counter mode, the C/T counts down the number of pulses loaded into CTUR and CTLR. Counting begins upon receipt of a start C/T command. Upon reaching the terminal count, the counter ready interrupt bit (ISR[4]) is set. The counter continues counting past the terminal count until stopped by the CPU. If MPO is programmed to be the output of the C/T, the output remains high until the terminal count is reached, at which time it goes low.

The output returns to the high state and ISR[4] is cleared when the counter is stopped by a stop counter command. The CPU may change the values of CTUR and CTLR at any time, but the new count becomes effective only on the next start counter command. If new values have not been loaded, the previous count values are preserved and used for the next count cycle.

2

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
Voltage from V_{CC} to GND[3]	−0.5 to +7.0	V
Voltage from any pin to ground[3]	−0.5 to V_{CC} ±10%	V
Power dissipation	0.2	W

DC ELECTRICAL CHARACTERISTICS T_A=0°C to +70°C, V_{CC}=5.0V ±10%[4,5,6]

	PARAMETER	TEST CONDITIONS	LIMITS Min	LIMITS Typ	LIMITS Max	UNIT
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage					
	All except X1/CLK		2.0			V
	X1/CLK		$0.9V_{CC}$		V_{CC}	V
V_{OL}	Output low voltage	I_{OL} = 2.4mA			0.4	V
V_{OH}[7]	Output high voltage					
	(except open drain outputs)	I_{OH} = −400µA	2.4			V
I_{IL}	Input leakage current	V_{IN} = 0 to V_{CC}	−10		10	µA
I_{LL}	Data bus 3-state leakage current	V_O = 0.4 to V_{CC}	−10		10	µA
I_{OD}	Open drain output leakage current	V_O = 0.4 to V_{CC}	−10		10	µA
I_{X1L}	X1/CLK low input current	V_{IN} = 0, X2 floated	−100	−30	0.0	µA
I_{X1H}	X1/CLK high input current	V_{IN} = V_{CC}, X2 floated	0.0	+30	100	µA
	When oscillator is in power down mode:					
I_{X1H}	X1/CLK high input current	V_{IN} = V_{CC}, X2 floated	2	6	10	mA
I_{X2L}	X2 low output current	V_{OUT} = 0, X1/CLK = V_{CC}			100	µA
I_{X2H}	X2 high output current	V_{OUT} = V_{CC}, X1/CLK = 0V			100	µA
I_{CC}	Power supply current				20	mA
	Standby				500	µA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. For X1/CLK, this swing is between 0.4V and 4.4V. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical supply voltages, and typical processing parameters.
7. Test condition for outputs: C_L=150pF, except interrupt outputs. Test conditions for interrupt outputs: C_L=50pF, R_L=2.7Kohm to V_{CC}.
8. Timing is illustrated and referenced to the WRN and RDN inputs. The device may also be operated with CEN as the 'strobing' input. In this case, all timing specifications apply referenced to the falling and rising edges of CEN. CEN and RDN (also CEN and WRN) are OR'ed internally. As a consequence, the signal asserted last initiates the cycle and the signal negated first terminates the cycle.
9. If CEN is used as the 'strobing' input, this parameter defines the minimum high time between one CEN and the next. The RDN signal must be negated for t_{RWD} to guarantee that any status register changes are valid.
10. Consecutive write operations to the same command require at least three edges of the X1 clock between writes.

AC ELECTRICAL CHARACTERISTICS T_A=0°C to +70°C, V_{CC}=5.0V ±10%[4,5,6,7]

PARAMETER		TENTATIVE LIMITS			UNIT
		Min	Typ	Max	
Reset timing (figure 1)					
t_{RES}	RESET pulse width	1.0			µs
Bus timing (figure 2)[8]					
t_{AS}	A0 – A2 set-up time to RDN, WRN low	10			ns
t_{AH}	A0 – A2 hold time from RDN, WRN high	0			ns
t_{CS}	CEN set-up time to RDN, WRN low	0			ns
t_{CH}	CEN hold time from RDN, WRN high	0			ns
t_{RW}	WRN, RDN pulse width	225			ns
t_{DD}	Data valid after RDN low			175	ns
t_{DF}	Data bus floating after RDN high			100	ns
t_{DS}	Data set-up time before WRN high	100			ns
t_{DH}	Data hold time after WRN high	10			ns
t_{RWD}[10]	Time between READs and/or WRITEs	200			ns
MPI and MPO timing (figure 3)[8]					
t_{PS}	MPI input set-up time before RDN low	0			ns
t_{PH}	MPI input hold time after RDN high	0			ns
t_{PD}	MPO output valid after WRN high			370	ns
Interrupt timing (figure 4)					
t_{IR}	INTRN negated:				
	Read RHR (RXRDY/FFULL interrupt)			370	ns
	Write THR (TXRDY, TXEMT interrupt)			370	ns
	Reset command (Break change interrupt)			370	ns
	Reset command (MPI change interrupt)			370	ns
	Stop C/T command (counter interrupt)			370	ns
	Write IMR (clear of interrupt mask bit)			270	ns
Clock timing (figure 5)					
t_{CLK}	X1/CLK high or low time	100			ns
f_{CLK}	X1/CLK frequency	2.0	3.6864	4.0	MHz
t_{CTC}	Counter/timer clock high or low time	100			ns
f_{CTC}	Counter/timer clock frequency	0		4.0	MHz
t_{RX}	RXC high or low time	220			ns
f_{RX}	RXC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
t_{TX}	TXC high or low time	220			ns
f_{TX}	TXC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
Transmitter timing (figure 6)					
t_{TXD}	TXD output delay from TxC low			350	ns
t_{TCS}	TXC output delay from TxD output data	0		150	ns
Receiver timing (figure 7)					
t_{RXS}	RXD data set-up time to RXC high	240			ns
t_{RXH}	RXD data hold time from RXC high	200			ns

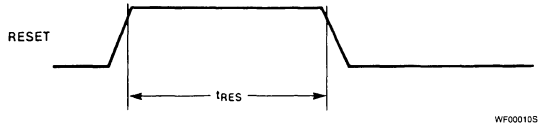


Figure 1. Reset Timing

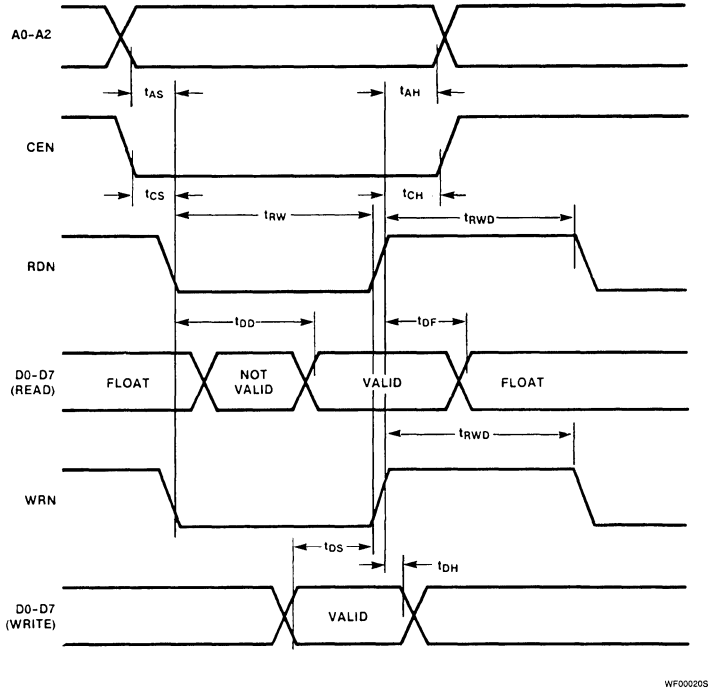


Figure 2. Bus Timing

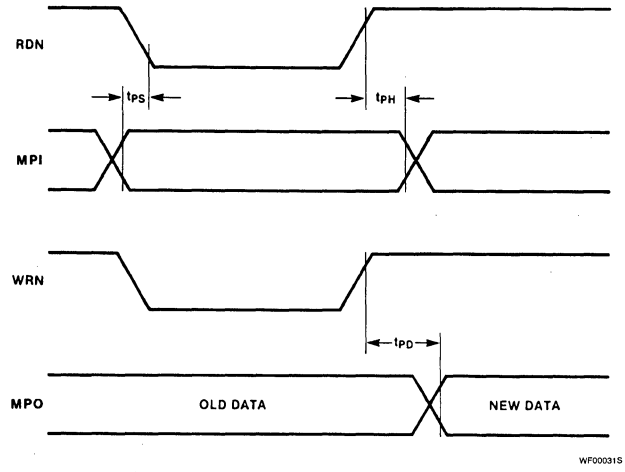
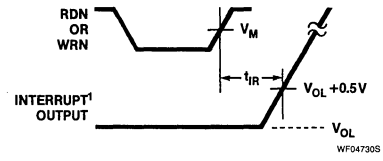


Figure 3. I/O Timing



NOTES:
1. INTRN or MPO when used as interrupt outputs.
2. The test for open drain outputs is intended to guarantee switching of the output transistor. Measurement of this response is referenced from the midpoint of the switching signal, V_M, to a point 0.5V above V_{OL}. This point represents a noise margin that assures true switching has occurred. Beyond this level, the effects of external circuitry and test environment are pronounced and can greatly affect the resultant measurement.

Figure 4. Interrupt Timing

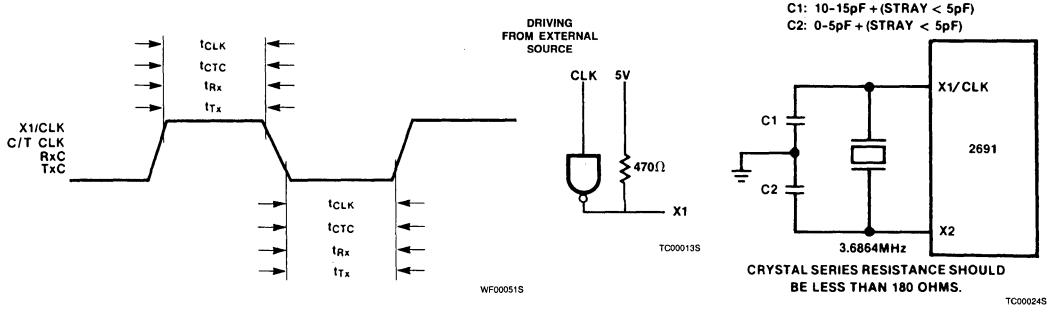


Figure 5. Clock Timing

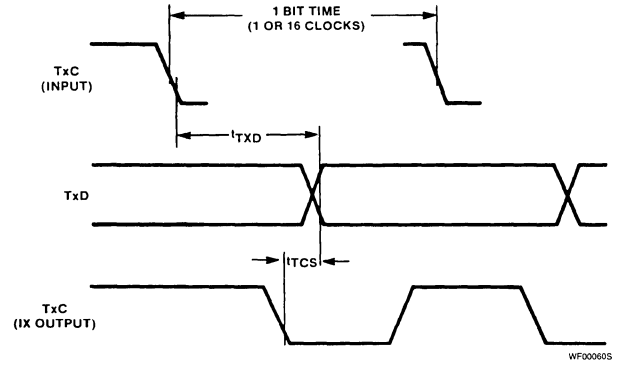


Figure 6. Transmit Timing

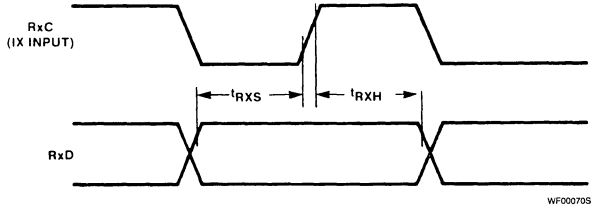
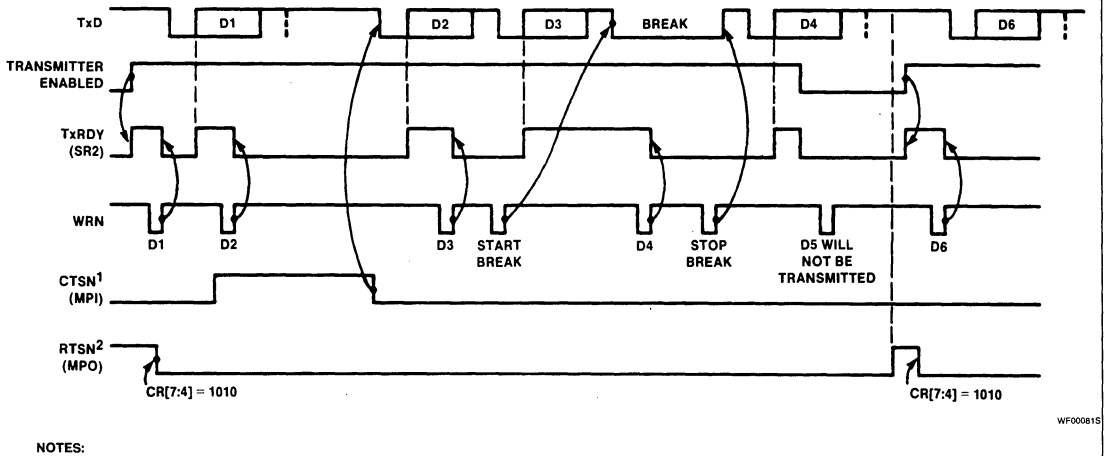
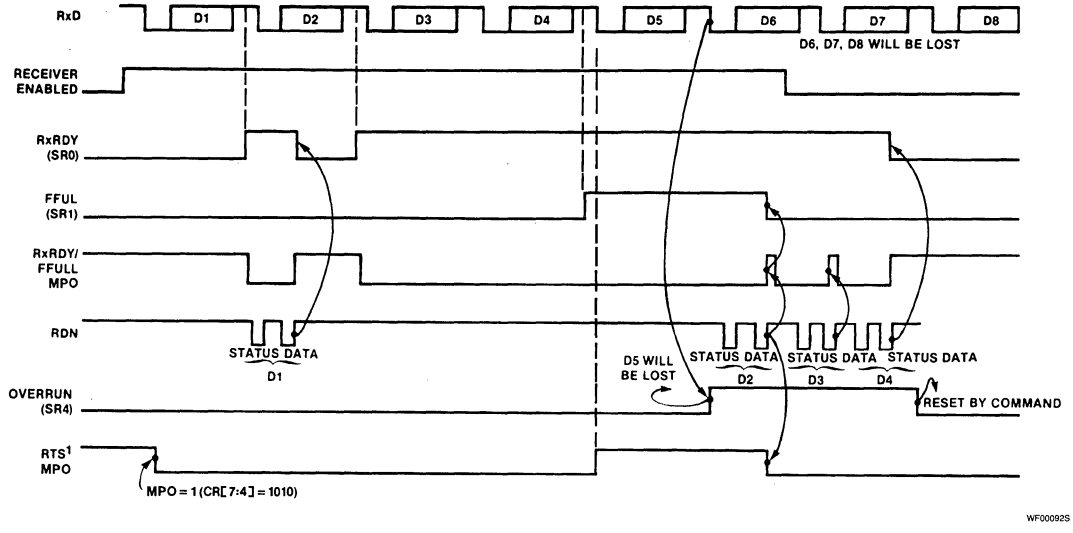


Figure 7. Receive Timing



NOTES:
1. Timing shown for MR2[4] = 1.
2. Timing shown for MR2[5] = 1.

Figure 8. Transmitter Timing



NOTES:
1. Timing shown for MR1[7] = 1.
2. Shown for ACR[2:0] = 111 and MR1[6] = 0.

Figure 9. Receiver Timing

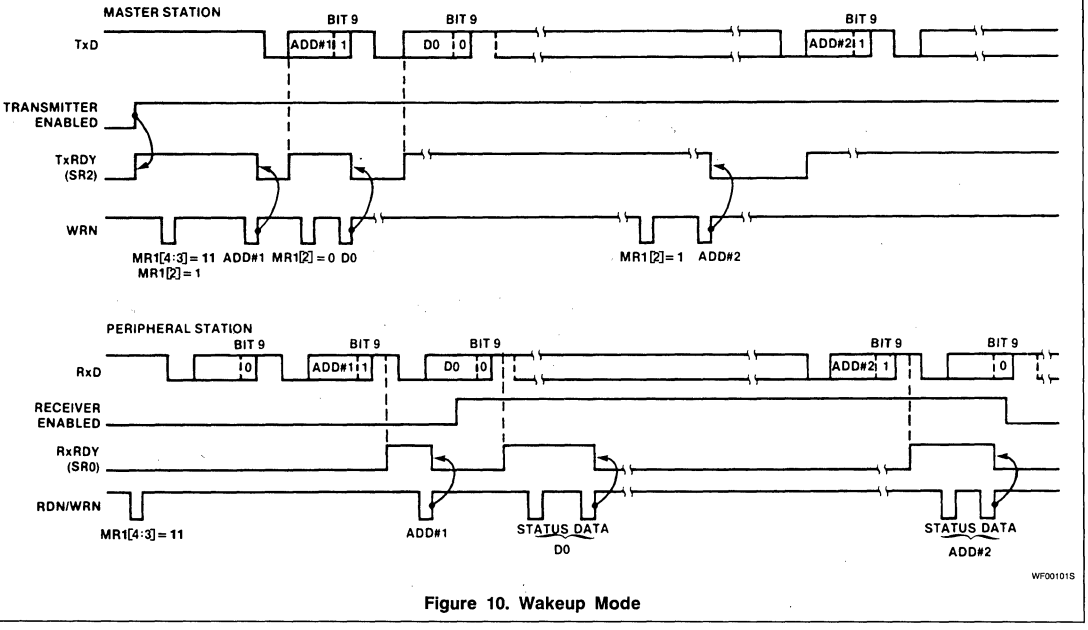


Figure 10. Wakeup Mode

SCC2698
Octal Universal Asynchronous Receiver/Transmitter (Octal UART)

Microprocessor Products

Product Specification (Brief)

DESCRIPTION

The Signetics SCC2698 Octal Universal Asynchronous Receiver/Transmitter (Octal-UART) is a single chip MOS-LSI communications device that provides an eight channel full-duplex asynchronous receiver/transmitter in a single package. It is fabricated with Signetics CMOS technology which combines the benefits of high density and low power consumption.

The operating speed of each receiver and transmitter can be selected independently as one of eighteen fixed baud rates, a 16x clock derived from a programmable counter/timer, or an external 1x or 16x clock. The baud rate generator and counter/timer can operate directly from a crystal or from external clock inputs. The ability to independently program the operating speed of the receiver and transmitter make the Octal-UART particularly functional for dual-speed channel applications such as clustered terminal systems.

The receiver is quadruply buffered to minimize the potential of receiver overrun, or to reduce interrupt overhead in interrupt driven systems. In addition, a handshaking capability is provided to disable a remote UART transmitter when the receiver buffer is full.

The Octal-UART provides a power down mode in which the oscillator is frozen but the register contents are stored. This results in reduced power consumption on the order of several magnitudes.

The Octal-UART is fully TTL compatible and operates from a single +5V power supply.

FEATURES

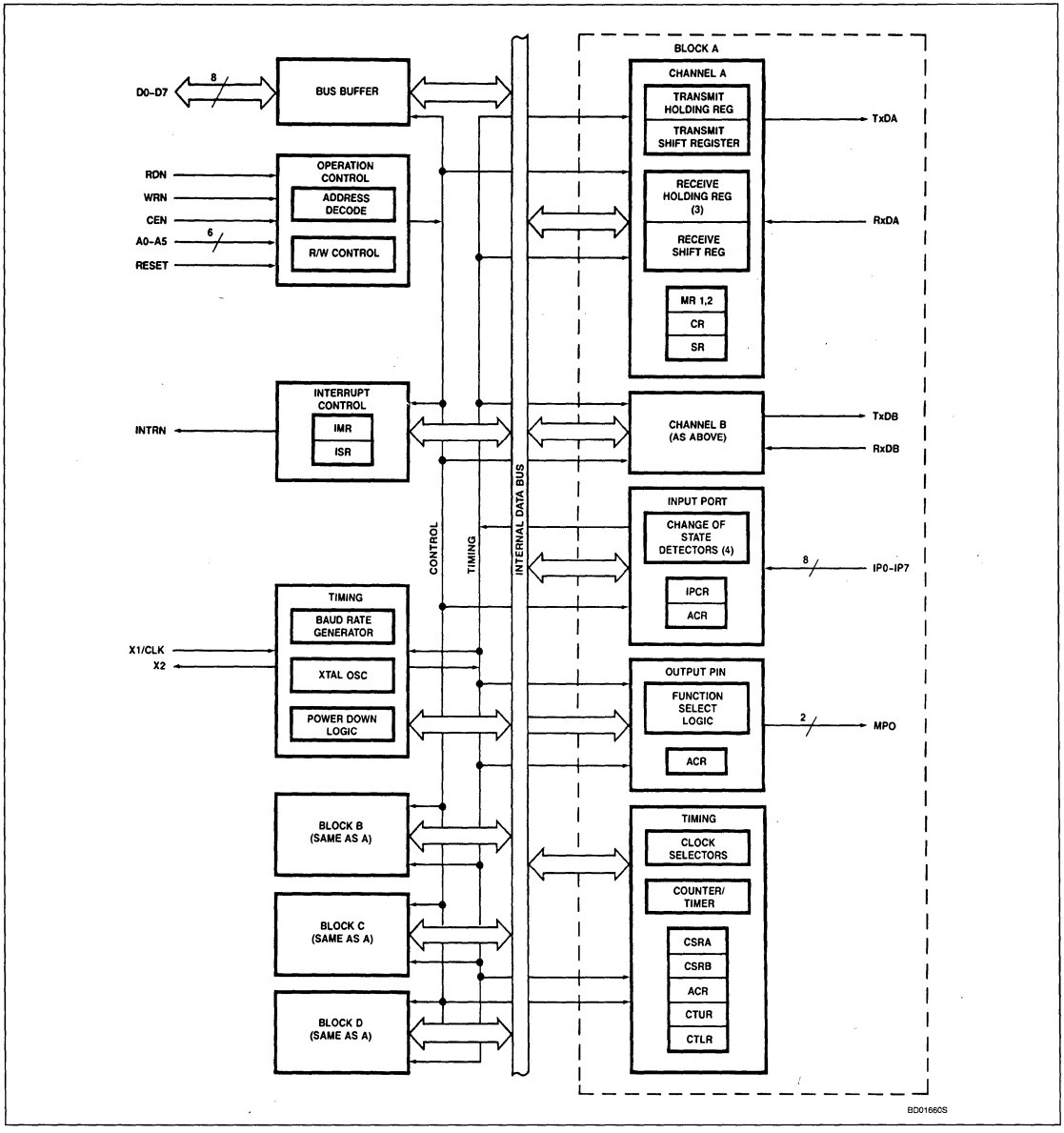
- **Eight full-duplex asynchronous receiver/transmitters**
- **Quadruple buffered receiver data register**
- **Programmable data format:**
  - **5 to 8 data bits plus parity**
  - **Odd, even, no parity or force parity**
  - **1, 1.5 or 2 stop bits programmable in 1/16 bit increments**
- **Baud rate for the receiver and transmitter selectable from:**
  - **18 fixed rates: 50 to 38.4K baud**
  - **One user defined rate derived from programmable timer/counter**
  - **External 1x or 16x clock**
- **Parity, framing, and overrun error detection**
- **False start bit detection**
- **Line break detection and generation**
- **Programmable channel mode**
  - **Normal (full duplex), automatic echo, local loop back, remote loop back**
- **Four multi-function programmable 16-bit counter/timers**
- **Single interrupt output with eight maskable interrupting conditions**
- **On-chip crystal oscillator**
- **TTL compatible**
- **Single +5V power supply with low power mode**
- **48-pin DIP and 84-pin PLCC packages**

Octal Universal Asynchronous Receiver/Transmitter — SCC2698

ORDERING CODE

PACKAGES	$V_{CC} = 5V \pm 5\%$, $T_A = 0$ to 70°C
Plastic DIP	SCC2698AC1N64
Plastic LCC	SCC2698AC1A84

BLOCK DIAGRAM



SCN68000
16-/32-Bit Microprocessor

Product Specification

Microprocessor Products

DESCRIPTION

The SCN68000 is the first implementation of the S68000 16/32 bit microprocessor architecture. The SCN68000 has a 16-bit data bus and 24-bit address bus, while the full architecture provides for 32-bit address and data buses. It is completely code-compatible with the SCN68008 8-bit data bus implementation of the S68000 and is downward code-compatible with the SCN68010 virtual extension and the SCN68020 32-bit implementation of the architecture. Any user-mode programs written using the SCN68000 instruction set will run unchanged on the SCN68008, SCN68010, and 68020. This is possible because the user programming model is identical for all four processors and the instruction sets are proper sub-sets of the complete architecture.

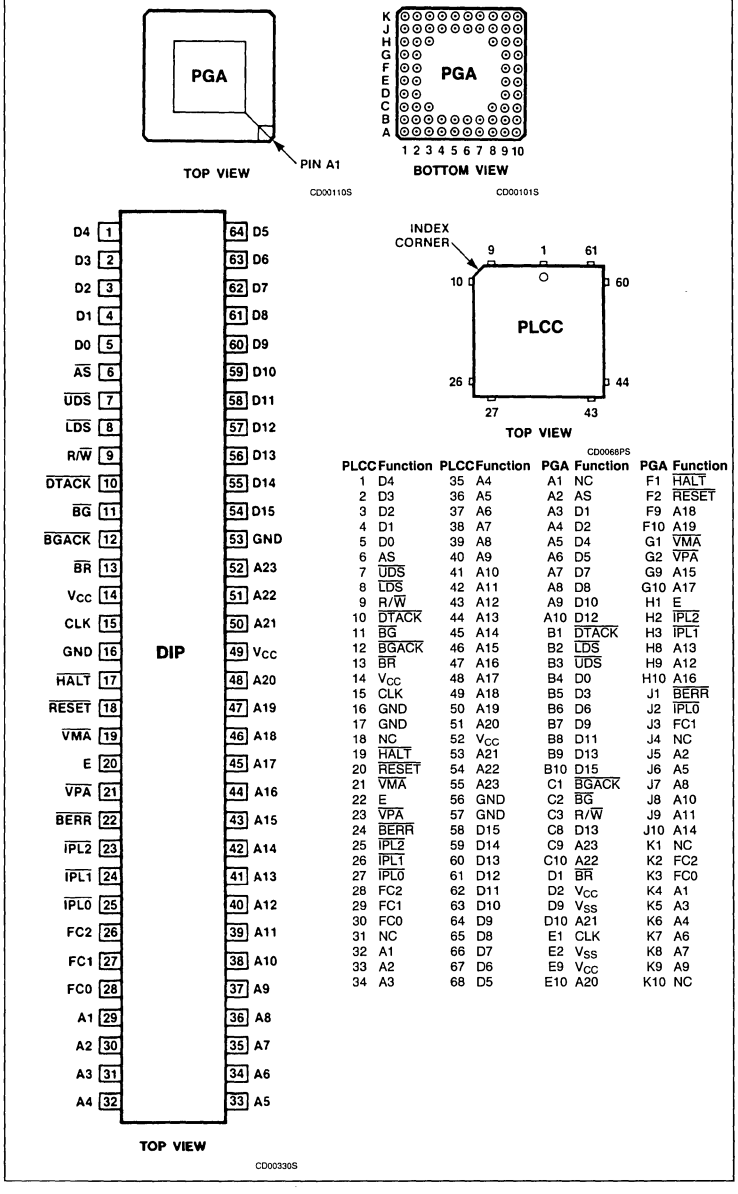
The SCN68000 possesses an asynchronous bus structure with a 24-bit address bus and a 16-bit data bus.

The resources available to the SCN68000 user consist of the following:

- 17 32-bit data and address registers
- 16 megabyte direct addressing range
- 56 powerful instruction types
- Operations on five main data types
- Memory-mapped I/O
- 14 addressing modes

As shown in the programming model (figure 1), the SCN68000 offers sixteen 32-bit registers and a 32-bit program counter. The first eight registers (D0 – D7) are used as data registers for byte (8-bit), word (16-bit), and long word (32-bit) operations. The second set of seven registers (A0 – A6) and the user stack pointer (USP) may be used as software stack pointers and base address registers. In addition, the registers may be used for word and long word operations. All of the 16 registers may be used as index registers.

PIN CONFIGURATION



PLCC	Function	PLCC	Function	PGA	Function	PGA	Function
1	D4	35	A4	A1	NC	F1	HALT
2	D3	36	A5	A2	AS	F2	RESET
3	D2	37	A6	A3	D1	F9	A18
4	D1	38	A7	A4	D2	F10	A19
5	D0	39	A8	A5	D4	G1	VMA
6	AS	40	A9	A6	D5	G2	VPA
7	UDS	41	A10	A7	D7	G9	A15
8	LDS	42	A11	A8	D8	G10	A17
9	R/W	43	A12	A9	D10	H1	E
10	DTACK	44	A13	A10	D12	H2	IPL2
11	BG	45	A14	B1	DTACK	H3	IPL1
12	BGACK	46	A15	B2	LDS	H8	A13
13	BR	47	A16	B3	UDS	H9	A12
14	VCC	48	A17	B4	D0	H10	A16
15	CLK	49	A18	B5	D3	J1	BERR
16	GND	50	A19	B6	D6	J2	IPL0
17	GND	51	A20	B7	D9	J3	FC1
18	NC	52	VCC	B8	D11	J4	NC
19	HALT	53	A21	B9	D13	J5	A2
20	RESET	54	A22	B10	D15	J6	A5
21	VMA	55	A23	C1	BGACK	J7	A8
22	E	56	GND	C2	BG	J8	A10
23	VPA	57	GND	C3	R/W	J9	A11
24	BERR	58	D15	C8	D13	J10	A14
25	IPL2	59	D14	C9	A23	K1	NC
26	IPL1	60	D13	C10	A22	K2	FC2
27	IPL0	61	D12	D1	BR	K3	FC0
28	FC2	62	D11	D2	VCC	K4	A1
29	FC1	63	D10	D9	VSS	K5	A3
30	FC0	64	D9	D10	A21	K6	A4
31	NC	65	D8	E1	CLK	K7	A6
32	A1	66	D7	E2	VSS	K8	A7
33	A2	67	D6	E9	VCC	K9	A9
34	A3	68	D5	E10	A20	K10	NC

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C			
	6MHz	8MHz	10MHz	12.5MHz
Ceramic DIP	SCN68000C6I64	SCN68000C8I64	SCN68000CAI64	SCN68000CBI64
Plastic DIP	SCN68000C6N64	SCN68000C8N64	SCN68000CAN64	SCN68000CBN64
Plastic LCC	SCN68000C6A68	SCN68000C8A68	SCN68000CAA68	SCN68000CBA68
Pin Grid Array	SCN68000C6AP68	SCN68000C8AP68	SCN68000CAP68	SCN68000CBAP68

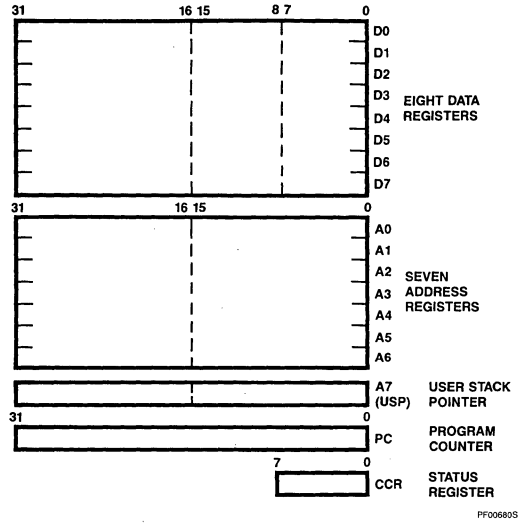


Figure 1. User Programming Model

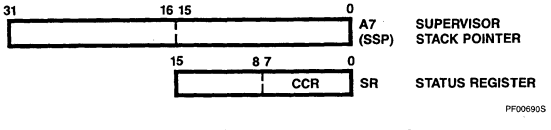


Figure 2. Supervisor Programming Model Supplement

In supervisor mode, the upper byte of the status register and the supervisor stack pointer (SSP) are also available to the programmer. These registers are shown in figure 2.

The status register (figure 3) contains the interrupt mask (eight levels available) as well as the condition codes: extend (X), negative (N), zero (Z), overflow (V), and carry (C). Additional status bits indicate that the processor is in a trace (T) mode and in a supervisor (S) or user state.

Data Types and Addressing Modes

Five basic data types are supported. These data types are:

- Bits
- BCD digits (4 bits)
- Bytes (8 bits)
- Words (16 bits)
- Long words (32 bits)

In addition, operations on other data types such as memory addresses, status word data, etc., are provided in the instruction set.

The 14 address modes, shown in table 1, include six basic types:

- Register direct
- Register indirect
- Absolute
- Program counter relative
- Immediate
- Implied

Included in the register indirect addressing modes is the capability to do postincrementing, predecrementing, offsetting, and indexing. The program counter relative mode can also be modified via indexing and offsetting.

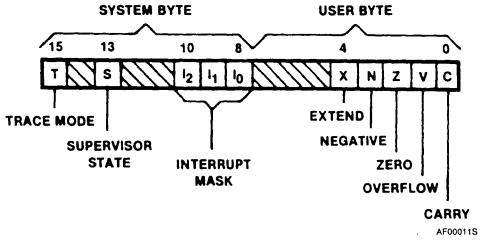


Figure 3. Status Register

Table 1. ADDRESSING MODES

ADDRESSING MODES	SYNTAX
Register direct addressing	
Data register direct	Dn
Address register direct	An
Absolute data addressing	
Absolute short	xxx.W
Absolute long	xxx.L
Program counter relative addressing	
Relative with offset	d_{16}(PC)
Relative with index and offset	d_8(PC,Xn)
Register indirect addressing	
Register indirect	(An)
Postincrement register indirect	(An) +
Predecrement register indirect	–(An)
Register indirect with offset	d_{16}(An)
Indexed register indirect with offset	d_8(An,Xn)
Immediate data addressing	
Immediate	#xxx
Quick immediate	#1 – #8
Implied addressing	
Implied register	SR/USP/SP/PC

NOTES:
Dn = Data register
An = Address register
Xn = Address or data register used as index register
SR = Status register
PC = Program counter
SP = Stack pointer
USP = User stack pointer
( ) = Effective Address
d_8 = 8-bit offset (displacement)
d_{16} = 16-bit offset (displacement)
#xxx = Immediate data

Instruction Set Overview

The SCN68000 instruction set is shown in table 2. Some additional instructions are variations, or subsets, of these and they appear in table 3. Special emphasis has been given to the instruction set's support of structured high-level languages to facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words and most instructions can use any of the 14 addressing modes. Combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned, multiply and divide, "quick" arithmetic operations, BCD arithmetic, and expanded operations (through traps).

Table 2. INSTRUCTION SET SUMMARY

MNEMONIC	DESCRIPTION
ABCD	Add decimal with extend
ADD	Add
AND	Logical AND
ASL	Arithmetic shift left
ASR	Arithmetic shift right
B_{CC}	Branch conditionally
BCHG	Bit test and change
BCLR	Bit test and clear
BRA	Branch always
BSET	Bit test and set
BSR	Branch to subroutine
BTST	Bit test
CHK	Check register against bounds
CLR	Clear operand
CMP	Compare
DB_{CC}	Test condition, decrement and branch
DIVS	Signed divide
DIVU	Unsigned divide
EOR	Exclusive OR
EXG	Exchange registers
EXT	Sign extend
JMP	Jump
JSR	Jump to subroutine
LEA	Load effective address
LINK	Link stack
LSL	Logical shift left
LSR	Logical shift right
MOVE	Move source to destination
MULS	Signed multiply
MULU	Unsigned multiply
NBCD	Negate decimal with extend
NEG	Negate
NOP	No operation
NOT	One's complement
OR	Logical OR
PEA	Push effective address
RESET	Reset external devices
ROL	Rotate left without extend
ROR	Rotate right without extend
ROXL	Rotate left with extend
ROXR	Rotate right with extend
RTE	Return from exception
RTR	Return and restore
RTS	Return from subroutine
SBCD	Subtract decimal with extend
S_{CC}	Set conditional
STOP	Stop
SUB	Subtract
SWAP	Swap data register halves
TAS	Test and set operand
TRAP	Trap
TRAPV	Trap on overflow
TST	Test
UNLK	Unlink

2

Table 3. VARIATIONS OF INSTRUCTION TYPES

INSTRUCTION TYPE	VARIATION	DESCRIPTION
ADD	ADD	Add
	ADDA	Add address
	ADDQ	Add quick
	ADDI	Add immediate
	ADDX	Add with extend
AND	AND	Logical AND
	ANDI	AND immediate
	ANDI to CCR	AND immediate to condition codes
	ANDI to SR	AND immediate to status register
CMP	CMP	Compare
	CMPA	Compare address
	CMPM	Compare memory
	CMPI	Compare immediate
EOR	EOR	Exclusive OR
	EORI	Exclusive OR immediate
	EORI to CCR	Exclusive OR immediate to condition codes
	EORI to SR	Exclusive OR immediate to status register
MOVE	MOVE	Move source to destination
	MOVEA	Move address
	MOVEM	Move multiple registers
	MOVEP	Move peripheral data
	MOVEQ	Move quick
	MOVE from SR	Move from status register
	MOVE to SR	Move to status register
	MOVE to CCR	Move to condition codes
	MOVE USP	Move user stack pointer
NEG	NEG	Negate
	NEGX	Negate with extend
OR	OR	Logical OR
	ORI	OR immediate
	ORI to CCR	OR immediate to condition codes
	ORI to SR	OR immediate to status register
SUB	SUB	Subtract
	SUBA	Subtract address
	SUBI	Subtract immediate
	SUBQ	Subtract quick
	SUBX	Subtract with extend

DATA ORGANIZATION AND ADDRESSING CAPABILITIES

This section contains a description of the registers and the data organization of the SCN68000.

Operand Size

Operand sizes are defined as follows: a byte equals 8 bits, a word equals 16 bits, and a long word equals 32 bits. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation. Implicit instructions support some subset of all three sizes.

Data Organization in Registers

The eight data registers support data operands of 1, 8, 16, or 32 bits. The seven address registers together with the stack pointers support address operands of 32 bits.

Data Registers

Each data register is 32 bits wide. Byte operands occupy the low order 8 bits, word operands the low order 16 bits, and long word operands the entire 32 bits. The least significant bit is addressed as bit zero; the most significant bit is addressed as 31.

When a data register is used as either a source or destination operand, only the appropriate low order portion is changed; the remaining high order portion is neither used nor changed.

Address Registers

Each address register and the stack pointer is 32 bits wide and holds a full 32-bit address. Address registers do not support the sized operands. Therefore, when an address register is used as a source operand, either the low order word or the entire long word operand is used depending on the operation size. When an address register is used as the destination operand, the entire register is affected regardless of the operation size. If the operation size is word, any other operands are sign extended to 32 bits before the operation is performed.

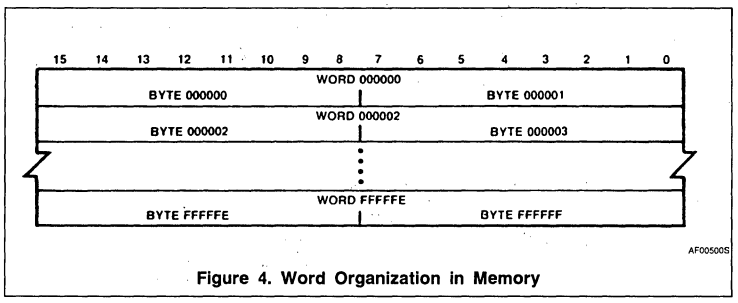


Figure 4. Word Organization in Memory

Data Organization in Memory

Bytes are individually addressable with the high order byte having an even address the same as the word, as shown in figure 4. The low order byte has an odd address that is one count higher than the word address. Instructions and multibyte data are accessed only on word (even byte) boundaries. If a long word datum is located at address n (n even), then the second word of that datum is located at address n + 2.

The data types supported by the SCN68000 are: bit data, integer data of 8, 16, or 32 bits, 32-bit addresses and binary coded decimal data. Each of these data types is put in memory, as shown in figure 5. The numbers indicate the order in which the data would be accessed from the processor.

Addressing

Instructions for the SCN68000 contain two kinds of information: the type of function to be performed and the location of the operand(s) on which to perform that function. The methods used to locate (address) the operand(s) are explained in the following paragraphs.

Instructions specify an operand location in one of three ways:

Register specification – the number of the register is given in the register field of their instruction.

Effective address – use of the different effective addressing modes.

Implicit reference – the definition of certain instructions implies the use of specific registers.

Instruction Format

Instructions are from one to five words in length as shown in figure 6. The length of the instruction and the operation to be performed is specified by the first word of the instruction which is called the operation word. The remaining words further specify the operands. These words are either immediate operands or extensions to the effective address mode specified in the operation word.

Program/Data References

The SCN68000 separates memory references into two classes: program references and data references. Program references, as the name implies, are references to that section of memory that contains the program being executed. Data references refer to that section of memory that contains data. Operand reads are from the data space except in the case of the program counter relative addressing mode. All operand writes are to the data space.

Register Specification

The register field within an instruction specifies the register to be used. Other fields within the instruction specify whether the register selected is an address or data register and how the register is to be used.

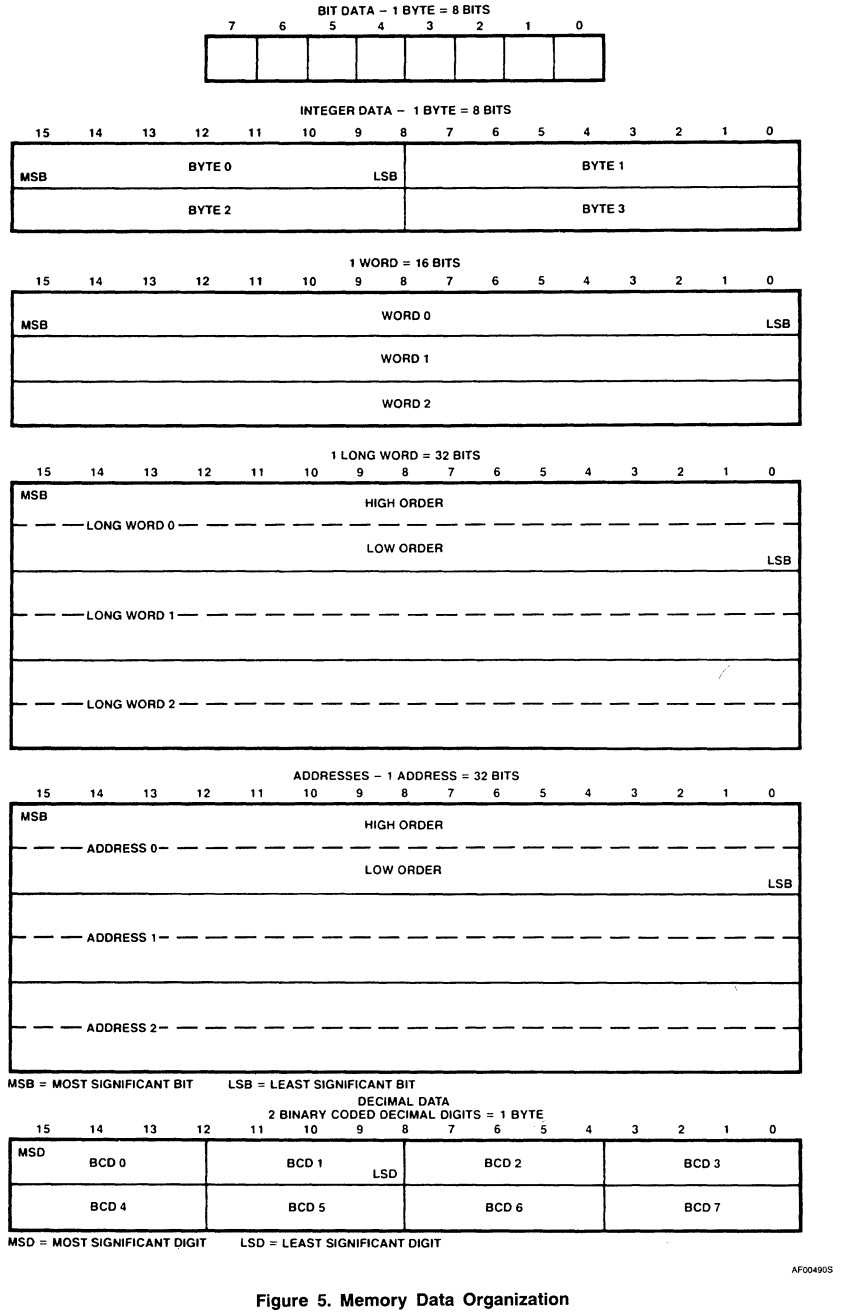


Figure 5. Memory Data Organization

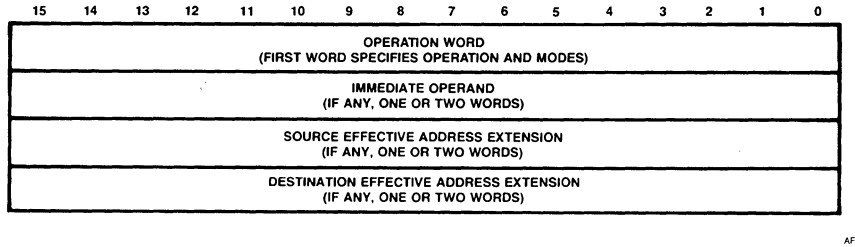


Figure 6. Instruction Operation Word General Format

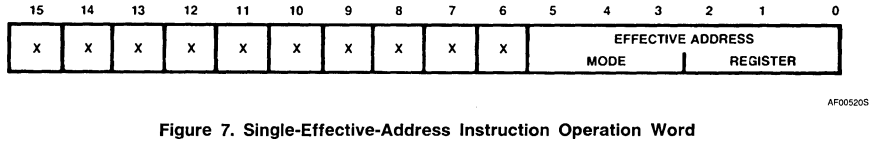


Figure 7. Single-Effective-Address Instruction Operation Word

Effective Address

Most instructions specify the location of an operand by using the effective address field in the operation word. For example, figure 7 shows the general format of the single-effective-address instruction operation word. The effective address is composed of two 3-bit fields: the mode field and the register field. The value in the mode field selects the different address modes. The register field contains the number of a register.

The effective address field may require additional information to fully specify the operand. This additional information, called the effective address extension, is contained in the following word or words and is considered part of the instruction, as shown in figure 6. The effective address modes are grouped into three categories: register direct, memory addressing, and special.

Register Direct Modes

These effective addressing modes specify that the operand is in one of 16 multifunction registers.

Data Register Direct — The operand is in the data register specified by the effective address register field.

Address Register Direct — The operand is in the address register specified by the effective address register field.

Memory Address Modes

These effective addressing modes specify that the operand is in memory and provide the specific address of the operand.

Address Register Indirect — The address of the operand is in the address register specified by the register field. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Address Register Indirect with Postincrement — The address of the operand is in the address register specified by the register field. After the operand address is used, it is incremented by one, two, or four depending on whether the size of the operand is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is incremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.

Address Register Indirect with Predecrement — The address of the operand is in the address register specified by the register field. Before the operand address is used, it is decremented by one, two, or four depending upon whether the operand size is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is decremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.

Address Register Indirect with Displacement — This addressing mode requires one word of extension. The address of the operand is the sum of the address in the address register and the sign-extended 16-bit displacement integer in the extension word. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Address Register Indirect with Index — This addressing mode requires one word of extension. The address of the operand is the sum of the address in the address register, the sign-extended displacement integer in the low order eight bits of the extension word, and the contents of the index register. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Special Address Modes

The special address modes use the effective address register field to specify the special addressing mode instead of a register number.

Absolute Short Address — This addressing mode requires one word of extension. The address of the operand is the extension word. The 16-bit address is sign extended before it is used. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Absolute Long Address — This addressing mode requires two words of extension. The address of the operand is developed by the concatenation of the extension words. The high order part of the address is the first extension word; the low order part of the address is the second extension word. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Program Counter with Displacement — This addressing mode requires one word of extension. The address of the operand is the sum of the address in the program counter and the sign-extended 16-bit displacement integer in the word. The value in the program counter is the address of the extension word. The reference is classified as a program reference.

Program Counter with Index — This addressing mode requires one word of extension. The address is the sum of the address in the program counter, the sign-extended displacement integer in the lower eight bits of the extension word, and the contents of the index register. The value in the program counter is the address of the extension word. This reference is classified as a program reference.

Immediate Data — This addressing mode requires either one or two words of extension depending on the size of the operation.

Byte operation – operand is low order byte of extension word

Word operation – operand is extension word

Long word operation – operand is in the two extension words, high order 16 bits are in the first extension word, low order 16 bits are in the second extension word.

Implicit Reference — Some instructions make implicit reference to the program counter (PC), the system stack pointer (SP), the supervisor stack pointer (SSP), the user stack pointer (USP), or the status register (SR). A selected set of instructions may reference the status register by means of the effective address field. These are:

ANDI to CCR	ORI to SR
ANDI to SR	MOVE to CCR
EORI to CCR	MOVE to SR
EORI to SR	MOVE from SR
ORI to CCR	

Effective Address Encoding Summary

Table 4 is a summary of the effective addressing modes discussed in the previous paragraphs.

System Stack

The system stack is used implicitly by many instructions; user stacks and queues may be created and maintained through the addressing modes. Address register seven (A7) is the system stack pointer (SP). The system stack pointer is either the supervisor stack pointer (SSP) or the user stack pointer (USP), depending on the state of the S bit in the status register. If the S bit indicates supervisor state, SSP is the active system stack pointer and the USP cannot be referenced as an address register. If the S bit indicates user state, the USP is the active system stack pointer and the SSP cannot be referenced. Each system stack fills from high memory to low memory.

Table 4. EFFECTIVE ADDRESS ENCODING SUMMARY

ADDRESSING MODE	MODE	REGISTER
Data register direct	000	Register number
Address register direct	001	Register number
Address register indirect	010	Register number
Address register indirect with postincrement	011	Register number
Address register indirect with predecrement	100	Register number
Address register indirect with displacement	101	Register number
Address register indirect with index	110	Register number
Absolute short	111	000
Absolute long	111	001
Program counter with displacement	111	010
Program counter with index	111	011
Immediate	111	100

Table 5. DATA MOVEMENT OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
EXG	32	Rx ↔ Ry
LEA	32	EA → An
LINK	–	AN → –(SP) SP → An SP + displacement → SP
MOVE	8, 16, 32	s → (EA)d
MOVEM	16, 32	(EA) → An, Dn An, Dn → EA
MOVEP	16, 32	(EA) → Dn Dn → (EA)
MOVEQ	8	#xxx → Dn
PEA	32	EA → –(SP)
SWAP	32	Dn[31:16] ↔ Dn[15:0]
UNLK	–	An → SP (SP) + → An

NOTES:
s = source; d = destination; [ ] = bit number; –( ) = indirect with predecrement; ( ) + = indirect with postdecrement; # = immediate data

INSTRUCTION SET SUMMARY

This section contains an overview of the form and structure of the SCN68000 instruction set. The instructions form a set of tools that include all the machine functions to perform the following operations:

Data movement	Bit manipulation
Integer arithmetic	Binary coded decimal
Logical	Program control
Shift and rotate	System control

The complete range of instruction capabilities combined with the flexible addressing modes described previously provide a very flexible base for program development.

Data Movement Operations

The basic method of data acquisition (transfer and storage) is provided by the move (MOVE) instruction. The move instruction and the effective addressing modes allow both address and data manipulation. Data move instructions allow byte, word, and long word operands to be transferred from memory to memory, memory to register, register to memory, and register to register. Address move instructions allow word and long word operand transfers and ensure that only legal address manipulations are executed. In addition to the general move instruction there are several special data movement instructions: move multiple registers (MOVEM), move peripheral data (MOVEP), exchange registers (EXG), load effective address (LEA), push effective address (PEA), link stack (LINK), unlink stack (UNLK), and move quick (MOVEQ). Table 5 is a summary of the data movement operations.

Integer Arithmetic Operations

The arithmetic operations include the four basic operations of add (ADD), subtract (SUB), multiply (MUL), and divide (DIV), as well as arithmetic compare (CMP), clear (CLR), and negate (NEG). The add and subtract instructions are available for both address and data operations, with data operations accepting all operand sizes. Address operations are limited to legal address size operands (16 or 32 bits). Data, address, and memory compare operations are also available. The clear and negate instructions may be used on all sizes of data operands.

The multiply and divide operations are available for signed and unsigned operands using word multiply to produce a long word product, and a long word dividend with word divisor to produce a word quotient with a word remainder.

Multiprecision and mixed size arithmetic can be accomplished using a set of extended instructions. These instructions are: add extended (ADDX), subtract extended (SUBX), sign extend (EXT), and negate binary with extend (NEGX).

A test operand (TST) instruction that will set the condition codes as a result of a compare of the operand with zero is also available. Test and set (TAS) is a synchronization instruction useful in multiprocessor systems. Table 6 is a summary of the integer arithmetic operations.

Table 6. INTEGER ARITHMETIC OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
ADD	8, 16, 32 16, 32	Dn + (EA) → Dn (EA) + Dn → (EA) (EA) + #xxx → (EA) An + (EA) → An
ADDX	8, 16, 32 16, 32	Dx + Dy + X → Dx –(Ax) + –(Ay) + X → (Ax)
CLR	8, 16, 32	0 → EA
CMP	8, 16, 32 16, 32	Dn – (EA) (EA) – #xxx (Ax) + –(Ay)– An – (EA)
DIVS	32 ÷ 16	Dn ÷ (EA) → Dn
DIVU	32 ÷ 16	Dn ÷ (EA) → Dn
EXT	8 → 16 16 → 32	$(Dn)_8$ → Dn_{16} $(Dn)_{16}$ → Dn_{32}
MULS	16 x 16 → 32	Dn x (EA) → Dn
MULU	16 x 16 → 32	Dn x (EA) → Dn
NEG	8, 16, 32	0 – (EA) → (EA)
NEGX	8, 16, 32	0 – (EA) – X → (EA)
SUB	8, 16, 32 16, 32	Dn – (EA) → Dn (EA) – Dn → (EA) (EA) – #xxx → (EA) An – (EA) → An
SUBX	8, 16, 32	Dx – Dy – X → Dx –(Ax) – –(Ay) – X → (Ax)
TAS	8	(EA) – 0, 1 → EA[7]
TST	8, 16, 32	(EA) – 0

NOTES:
[ ] = bit number
= immediate data
–( ) = indirect with predecrement
( )+ = indirect with postdecrement

Logical Operations

Logical operation instructions AND, OR, EOR, and NOT are available for all sizes of integer data operands. A similar set of immediate instructions (ANDI, ORI, and EORI) provide these logical operations with all sizes of immediate data. Table 7 is a summary of the logical operations.

Table 7. LOGICAL OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
AND	8, 16, 32	Dn ∧ (EA) → Dn (EA) ∧ Dn → (EA) (EA) ∧ #xxx → (EA)
OR	8, 16, 32	Dn ∨ (EA) → Dn (EA) ∨ Dn → (EA) (EA) ∨ #xxx → (EA)
EOR	8, 16, 32	(EA) ⊕ Dy → (EA) (EA) ⊕ #xxx → (EA)
NOT	8, 16, 32	~(EA) → (EA)

NOTES:
= immediate data
~ = invert
∧ = logical AND
∨ = logical OR
⊕ = logical exclusive OR

2

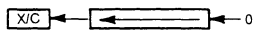
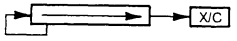
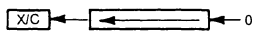
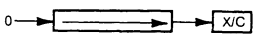
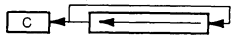
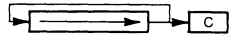
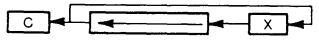
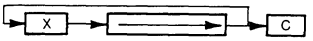
Shift and Rotate Operations

Shift operations in both directions are provided by the arithmetic instructions ASR and ASL and logical shift instructions LSR and LSL. The rotate instructions (with and without extend) available are ROXR, ROXL, ROR, and ROL. All shift and rotate operations can be performed in either registers or memory. Register shifts and rotates support all operand sizes and allow a shift count specified in a data register.

Memory shifts and rotates are for word operands only and allow only single-bit shifts or rotates.

Table 8 is a summary of the shift and rotate operations.

Table 8. SHIFT AND ROTATE OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
ASL	8, 16, 32	
ASR	8, 16, 32	
LSL	8, 16, 32	
LSR	8, 16, 32	
ROL	8, 16, 32	
ROR	8, 16, 32	
ROXL	8, 16, 32	
ROXR	8, 16, 32	

Bit Manipulation Operations

Bit manipulation operations are accomplished using the following instructions: bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). Table 9 is a summary of the bit manipulation operations. (Z is bit 2 of the status register.)

Table 9. BIT MANIPULATION OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
BTST	8, 32	~bit of (EA) → Z
BSET	8, 32	~bit of (EA) → Z 1 → bit of EA
BCLR	8, 32	~bit of (EA) → Z 0 → bit of EA
BCHG	8, 32	~bit of (EA) → Z ~bit of (EA) → bit of EA

NOTE: ~ = invert

Binary Coded Decimal Operations

Multiprecision arithmetic operations on binary coded decimal numbers are accomplished using the following instructions: add decimal with extend (ABCD), subtract decimal with extend (SBCD), and negate decimal with extend (NBCD). Table 10 is a summary of the binary coded decimal operations.

Table 10. BINARY CODED DECIMAL OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
ABCD	8	$Dx_{10} + Dy_{10} + X \rightarrow Dx$ $-(Ax)_{10} + -(Ay)_{10} + X \rightarrow (Ax)$
SBCD	8	$Dx_{10} - Dy_{10} - X \rightarrow Dx$ $-(Ax)_{10} - -(Ay)_{10} - X \rightarrow (Ax)$
NBCD	8	$0 - (EA)_{10} - X \rightarrow (EA)$

NOTE: -( ) = indirect with predecrement.

Program Control Operations

Program control operations are accomplished using a series of conditional and unconditional branch instructions and return instructions. These instructions are summarized in table 11.

The conditional instructions provide setting and branching for the following conditions:

CC – carry clear	LS – low or same
CS – carry set	LT – less than
EQ – equal	MI – minus
F – never true	NE – not equal
GE – greater or equal	PL – plus
GT – greater than	T – always true
HI – high	VC – no overflow
LE – less or equal	VS – overflow

Table 11. PROGRAM CONTROL OPERATIONS

INSTRUCTION	OPERATION
Conditional	
B_{CC}	Branch conditionally (14 conditions) 8- and 16-bit displacement
DB_{CC}	Test condition, decrement, and branch 16-bit displacement
S_{CC}	Set byte conditionally (16 conditions)
Unconditional	
BRA	Branch always 8- and 16-bit displacement
BSR	Branch to subroutine 8- and 16-bit displacement
JMP	Jump
JSR	Jump to subroutine
Returns	
RTR	Return and restore condition codes
RTS	Return from subroutine

System Control Operations

System control operations are accomplished by using privileged instructions, trap generating instructions, and instructions that use or modify the status register. These instructions are summarized in table 12.

Table 12. SYSTEM CONTROL OPERATIONS

INSTRUCTION	OPERATION
Privileged	
ANDI to SR	Logical AND to status register
EORI to SR	Logical EOR to status register
MOVE EA to SR	Load new status register
MOVE USP	Move user stack pointer
ORI to SR	Logical OR to status register
RESET	Reset external devices
RTE	Return from exception
STOP	Stop program execution
Trap generating	
CHK	Check data register against upper bounds
TRAP	Trap
TRAPV	Trap on overflow
Status register	
ANDI to CCR	Logical AND to condition codes
EORI to CCR	Logical EOR to condition codes
MOVE EA to CCR	Load new condition codes
MOVE SR to EA	Store status register
ORI to CCR	Logical OR to condition codes

2

SIGNAL AND BUS OPERATION DESCRIPTION

This section contains a brief description of the input and output signals. A discussion of bus operation during the various machine cycles and operations is also given.

NOTE

The terms **assertion** and **negation** will be used extensively. This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true, independent of whether that level is represented by a high or low voltage. The term negate or negation is used to indicate that a signal is inactive or false.

Signal Description

The input and output signals can be functionally organized into the groups shown in figure 8. The following paragraphs provide a brief description of the signals and a reference (if applicable) to other paragraphs that contain more detail about the function being performed.

Address Bus (A1 through A23)

This 23-bit, unidirectional, three-state bus is capable of addressing 8 megawords of data. It provides the address for bus operation during all cycles except interrupt cycles. During interrupt cycles, address lines A1, A2, and A3 provide information about what level interrupt is being serviced while address lines A4 through A23 are all set to a logic high.

Data Bus (D0 through D15)

This 16-bit bidirectional, three-state bus is the general purpose data path. It can transfer and accept data in either word or byte length. During an interrupt acknowledge cycle, the external device supplies the vector number on data lines D0 – D7.

Asynchronous Bus Control

Asynchronous data transfers are handled using the following control signals: address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are explained in the following paragraphs.

Address Strobe ($\overline{AS}$) — This signal indicates that there is a valid address on the address bus.

Read/Write (R/$\overline{W}$) — This signal defines the data bus transfer as a read or write cycle. The R/$\overline{W}$ signal also works in conjunction with the data strobes as explained in the following paragraph.

Upper and Lower Data Strobe ($\overline{UDS}$, $\overline{LDS}$) — These signals control the flow of data on the data bus, as shown in table 13. When the R/$\overline{W}$ line is high, the processor will read from the data bus as indicated. When the R/$\overline{W}$ line is low, the processor will write to the data bus as shown.

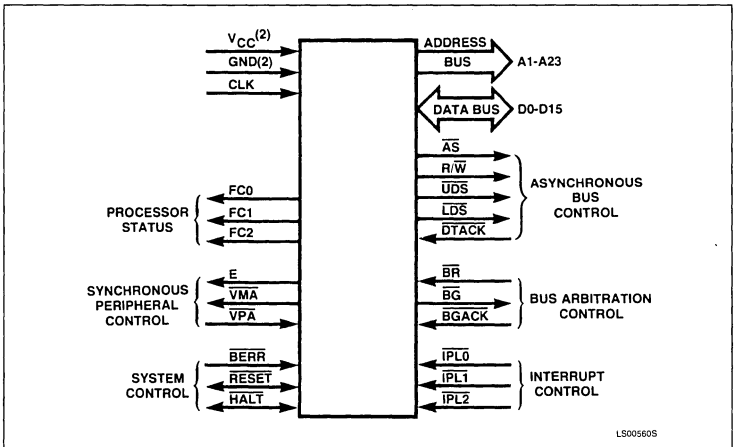


Figure 8. Input and Output Signals

Table 13. DATA STROBE CONTROL OF DATA BUS

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8 – D15	D0 – D7
High	High	–	No valid data	No valid data
Low	Low	High	Valid data bits 8 – 15	Valid data bits 0 – 7
High	Low	High	No valid data	Valid data bits 0 – 7
Low	High	High	Valid data bits 8 – 15	No valid data
Low	Low	Low	Valid data bits 8 – 15	Valid data bits 0 – 7
High	Low	Low	Valid data bits 0 – 7*	Valid data bits 0 – 7
Low	High	Low	Valid data bits 8 – 15	Valid data bits 8 – 15*

*These conditions are a result of current implementation and may not appear on future devices.

Data Transfer Acknowledge ($\overline{DTACK}$) — This input indicates that the data transfer is completed. When the processor recognizes $\overline{DTACK}$ during a read cycle, data is latched and the bus cycle terminated. When $\overline{DTACK}$ is recognized during a write cycle, the bus cycle is terminated. (Refer to **Asynchronous Versus Synchronous Operation**).

Bus Arbitration Control

The three signals, bus request, bus grant, and bus grant acknowledge, form a bus arbitration circuit to determine which device will be the bus master device.

Bus Request ($\overline{BR}$) — This input is wire ORed with all other devices that could be bus masters. This input indicates to the processor that some other device desires to become the bus master.

Bus Grant ($\overline{BG}$) — This output indicates to all other potential bus master devices that the processor will release bus control at the end of the current bus cycle.

Bus Grant Acknowledge ($\overline{BGACK}$) — This input indicates that some other device has become the bus master. This signal should not be asserted until the following four conditions are met:

1. a bus grant has been received,
2. address strobe is inactive which indicates that the microprocessor is not using the bus,
3. data transfer acknowledge is inactive which indicates that neither memory nor peripherals are using the bus, and

4. bus grant acknowledge is inactive which indicates that no other device is still claiming bus mastership.

Interrupt Control ($\overline{IPL0}$ $\overline{IPL1}$, $\overline{IPL2}$)
These input pins indicate the encoded priority level of the device requesting an interrupt. Level seven is the highest priority while level zero indicates that no interrupts are requested. Level seven cannot be masked. The least significant bit is given in $\overline{IPL0}$ and the most significant bit is contained in $\overline{IPL2}$. These lines must remain stable until the processor signals interrupt acknowledge (FC0 – FC2 are all high) to insure that the interrupt is recognized.

System Control
The system control inputs are used to either reset or halt the processor and to indicate to the processor that bus errors have occurred. The three system control inputs are explained in the following paragraphs.

Bus Error ($\overline{BERR}$) — This input informs the processor that there is a problem with the cycle currently being executed. Problems may be a result of:
1. nonresponding devices,
2. interrupt vector number acquisition failure,
3. illegal access request as determined by a memory management unit, or
4. other application dependent errors.

The bus error signal interacts with the halt signal to determine if the current bus cycle should be re-executed or if exception processing should be performed.

Refer to **Bus Error and Halt Operation** for additional information about the interaction of the bus error and halt signals.

Reset ($\overline{RESET}$) — This bidirectional signal line acts to reset (start a system initialization sequence) the processor in response to an external reset signal. An internally generated reset (result of a $\overline{RESET}$ instruction) causes all external devices to be reset and the internal state of the processor is not affected. A total system reset (processor and external devices) is the result of external $\overline{HALT}$ and $\overline{RESET}$ signals applied at the same time. Refer to **Reset Operation** for further information.

Halt ($\overline{HALT}$) — When this bidirectional line is driven by an external device, it will cause the processor to stop at the completion of the current bus cycle. When the processor has been halted using this input, all control signals are inactive and all three-state lines are put in their high-impedance state (refer to table 15). Refer to **Bus Error and Halt Operation** for additional information about the interaction between the $\overline{HALT}$ and bus error signals.

When the processor has stopped executing instructions, such as in a double bus fault condition (refer to **Double Bus Faults**), the $\overline{HALT}$ line is driven by the processor to indicate to external devices that the processor has stopped.

Peripheral Control
These control signals are used to allow the interfacing of synchronous peripheral devices with the asynchronous SCN68000. These signals are explained in the following paragraphs.

Enable (E) — This signal is the standard enable signal common to all synchronous type peripheral devices. The period for this output is ten SCN68000 clock periods (six clocks low, four clocks high). Enable is generated by an internal ring counter which may come up in any state (i.e., at power on, it is impossible to guarantee phase relationship of E to CLK). E is a free-running clock and runs regardless of the state of the bus on the MPU.

Valid Peripheral Address ($\overline{VPA}$) — This input indicates that the device or region addressed is a synchronous family device and that data transfer should be synchronized with the enable (E) signal. This input also indicates that the processor should use automatic vectoring for an interrupt. Refer to **Interface with Synchronous Peripherals.**

Valid Memory Address ($\overline{VMA}$) — This output is used to indicate to synchronous peripheral devices that there is a valid address on the address bus and the processor is synchronized to enable. This signal only responds to a valid peripheral address ($\overline{VPA}$) input which indicates that the peripheral is a synchronous family device.

Processor Status (FC0, FC1, FC2)
These function code outputs indicate the state (user or supervisor) and the cycle type currently being executed, as shown in table 14. The information indicated by the function code outputs is valid whenever address strobe ($\overline{AS}$) is active.

Table 14. FUNCTION CODE OUTPUTS

FUNCTION CODE OUTPUT			CYCLE TYPE
FC2	FC1	FC0	
Low	Low	Low	(Undefined, reserved)
Low	Low	High	User data
Low	High	Low	User program
Low	High	High	(Undefined, reserved)
High	Low	Low	(Undefined, reserved)
High	Low	High	Supervisor data
High	High	Low	Supervisor program
High	High	High	Interrupt acknowledge

Clock (CLK)
The clock input is a TTL-compatible signal that is internally buffered for development of the internal clocks needed by the processor. The clock input should not be gated off at any time and the clock signal must conform to minimum and maximum pulse width times.

Signal Summary
Table 15 is a summary of all the signals discussed in the previous paragraphs.

Bus Operation

The following paragraphs explain control signal and bus operation during data transfer operations, bus arbitration, bus error and halt conditions, and reset operation.

Data Transfer Operations
Transfer of data between devices involves the following leads:
1. address bus A1 through A23,
2. data bus D0 through D15, and
3. control signals.

The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cycles, the bus master assumes responsibility for deskewing all signals it issues at both the start and end of a cycle. In addition, the bus master is responsible for deskewing the acknowledge and data signals from the slave device.

The following paragraphs explain the read, write, and read-modify-write cycles. The indivisible read-modify-write cycle is the method used by the SCN68000 for interlocked multiprocessor communications.

Read Cycle — During a read cycle, the processor receives data from the memory or a peripheral device. The processor reads bytes of data in all cases. If the instruction specifies a word (or double word) operation, the processor reads both upper and lower bytes simultaneously by asserting both upper

and lower data strobes. When the instruction specifies byte operation, the processor uses an internal A0 bit to determine which byte to read and then issues the data strobe required for that byte. For byte operations, when the A0 bit equals zero, the upper data strobe is issued. When the A0 bit equals one, the lower data strobe is issued. When the data is received, the processor correctly positions it internally.

A word read cycle flowchart is given in figure 9. A byte read cycle flowchart is give in figure 10. Read cycle timing is given in figure 11. Figure 12 details word and byte read cycle operations.

Table 15. SIGNAL SUMMARY

SIGNAL NAME	MNEMONIC	INPUT/OUTPUT	ACTIVE STATE	Hi-Z	
				on $\overline{HALT}$	on $\overline{BGACK}$
Address bus	A1 – A23	Output	High	Yes	Yes
Data bus	D0 – D15	Input/output	High	Yes	Yes
Address strobe	$\overline{AS}$	Output	Low	No	Yes
Read/write	R/$\overline{W}$	Output	Read-high Write-low	No	Yes
Upper and lower data strobes	$\overline{UDS}$, $\overline{LDS}$	Output	Low	No	Yes
Data transfer acknowledge	$\overline{DTACK}$	Input	Low	No	No
Bus request	$\overline{BR}$	Input	Low	No	No
Bus grant	$\overline{BG}$	Output	Low	No	No
Bus grant acknowledge	$\overline{BGACK}$	Input	Low	No	No
Interrupt priority level	$\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$	Input	Low	No	No
Bus error	$\overline{BERR}$	Input	Low	No	No
Reset	$\overline{RESET}$	Input/output	Low	No[1]	No[1]
Halt	$\overline{HALT}$	Input/output	Low	No[1]	No[1]
Enable	E	Output	High	No	No
Valid memory address	$\overline{VMA}$	Output	Low	No	Yes
Valid peripheral address	$\overline{VPA}$	Input	Low	No	No
Function code output	FC0, FC1, FC2	Output	High	No[2]	Yes
Clock	CLK	Input	High	No	No
Power input	V_{CC}	Input	–	–	–
Ground	GND	Input	–	–	–

NOTES:
1. Open Drain
2. Function codes are placed in high-impedance state during HALT.

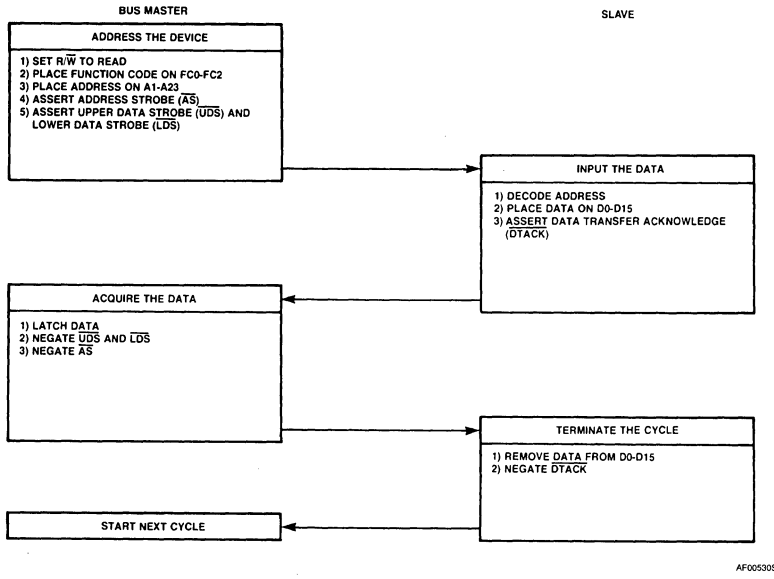


Figure 9. Word Read Cycle Flowchart

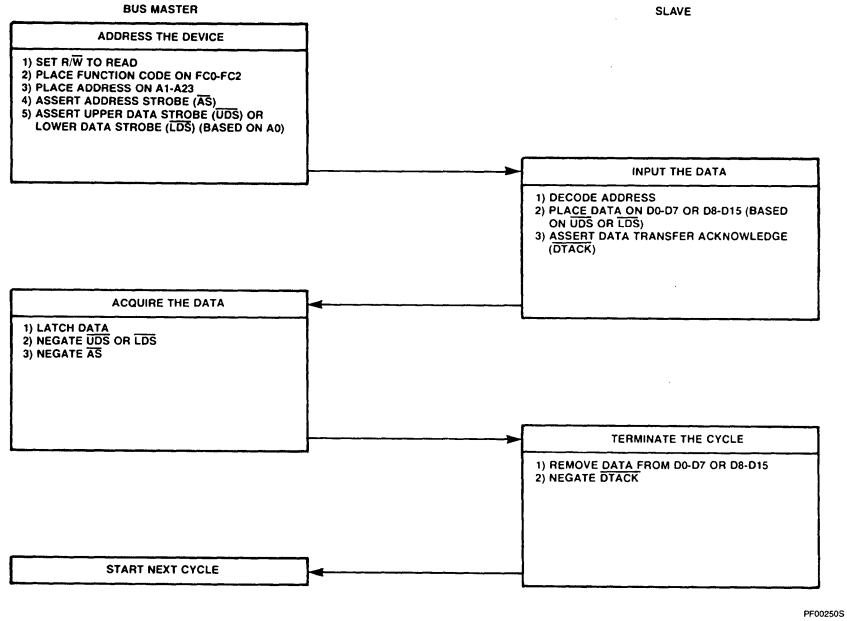


Figure 10. Byte Read Cycle Flowchart

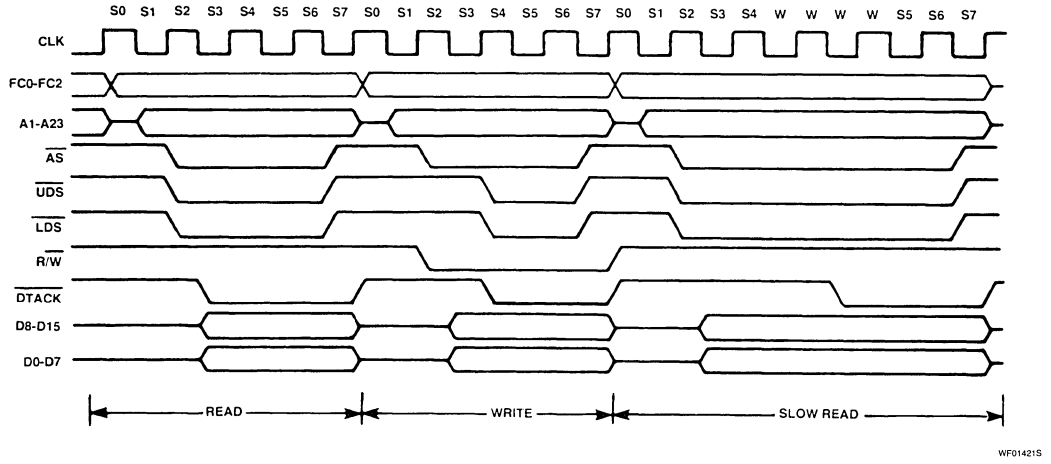


Figure 11. Read and Write Cycle Timing Diagram

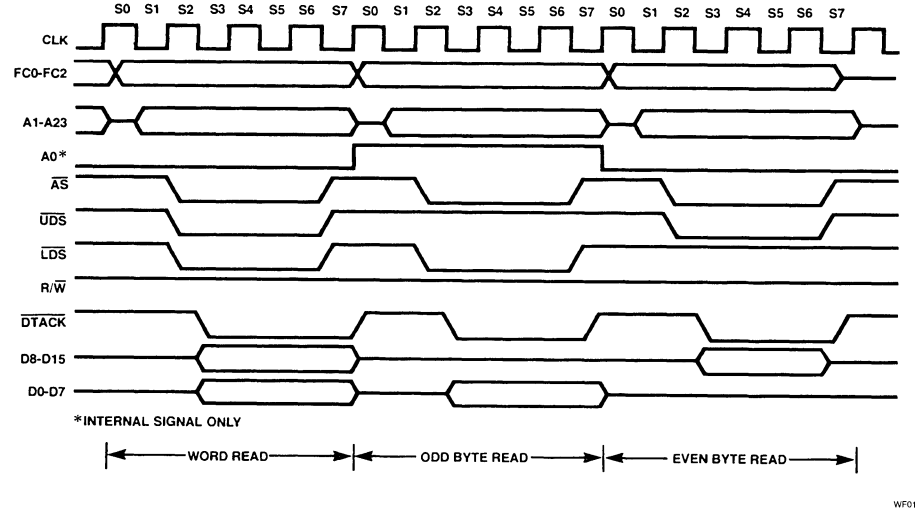


Figure 12. Word and Byte Read Cycle Timing Diagram

Write Cycle — During a write cycle, the processor sends data to either the memory or a peripheral device. The processor writes bytes of data in all cases. If the instruction specifies a word operation, the processor writes both bytes. When the instruction specifies a byte operation, the processor uses an internal A0 bit to determine which byte to write and then issues the data strobe required for that byte. For byte operations, when the A0 bit equals zero, the upper data strobe is issued. When the A0 bit equals one, the lower data strobe is issued. A word write cycle flowchart is given in figure 13. A byte write cycle flowchart is given in figure 14. Write cycle timing is given in figure 11. Figure 15 details word and byte write cycle operation.

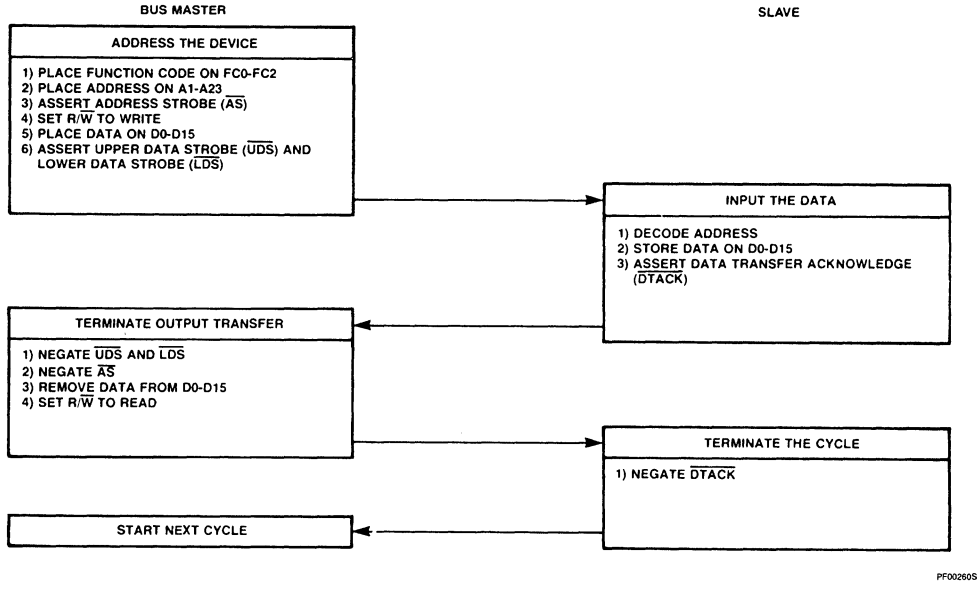


Figure 13. Word Write Cycle Flowchart

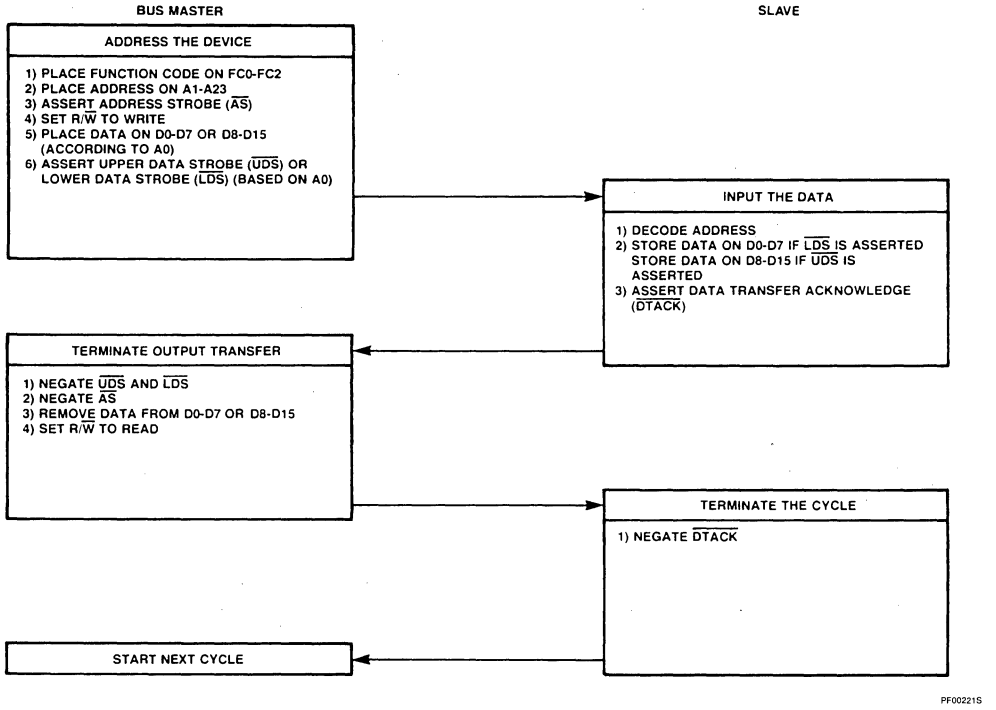


Figure 14. Byte Write Cycle Flowchart

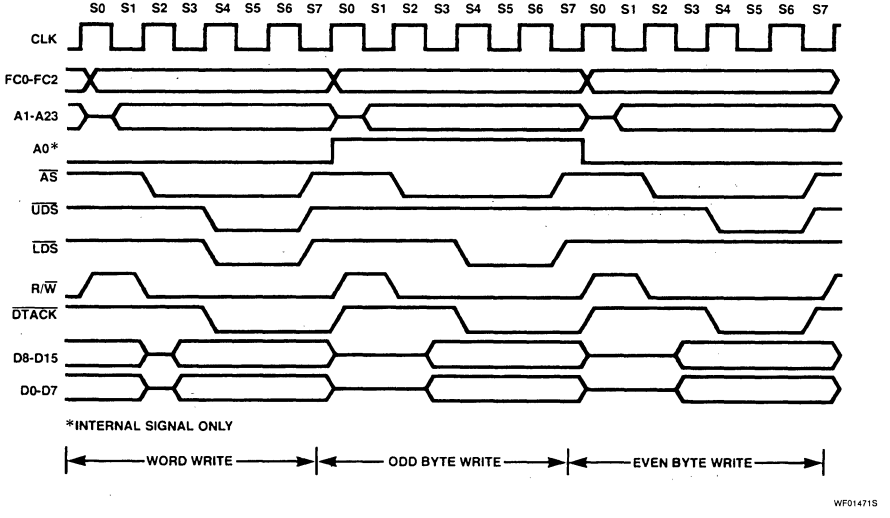


Figure 15. Word and Byte Write Cycle Timing Diagram

Read-Modify-Write Cycle — The read-modify-write cycle performs a read, modifies the data in the arithmetic-logic unit, and writes the data back to the same address. In the SCN68000, this cycle is indivisible in that the address strobe is asserted throughout the entire cycle. The test and set (TAS) instruction uses this cycle to provide meaningful communication between processors in a multiple processor environment. This instruction is the only instruction that uses the read-modify-write cycles and since the test and set instruction only operates on bytes, all read-modify-write cycles are byte operations. A read-modify-write cycle flowchart is given in figure 16 and a timing diagram is given in figure 17.

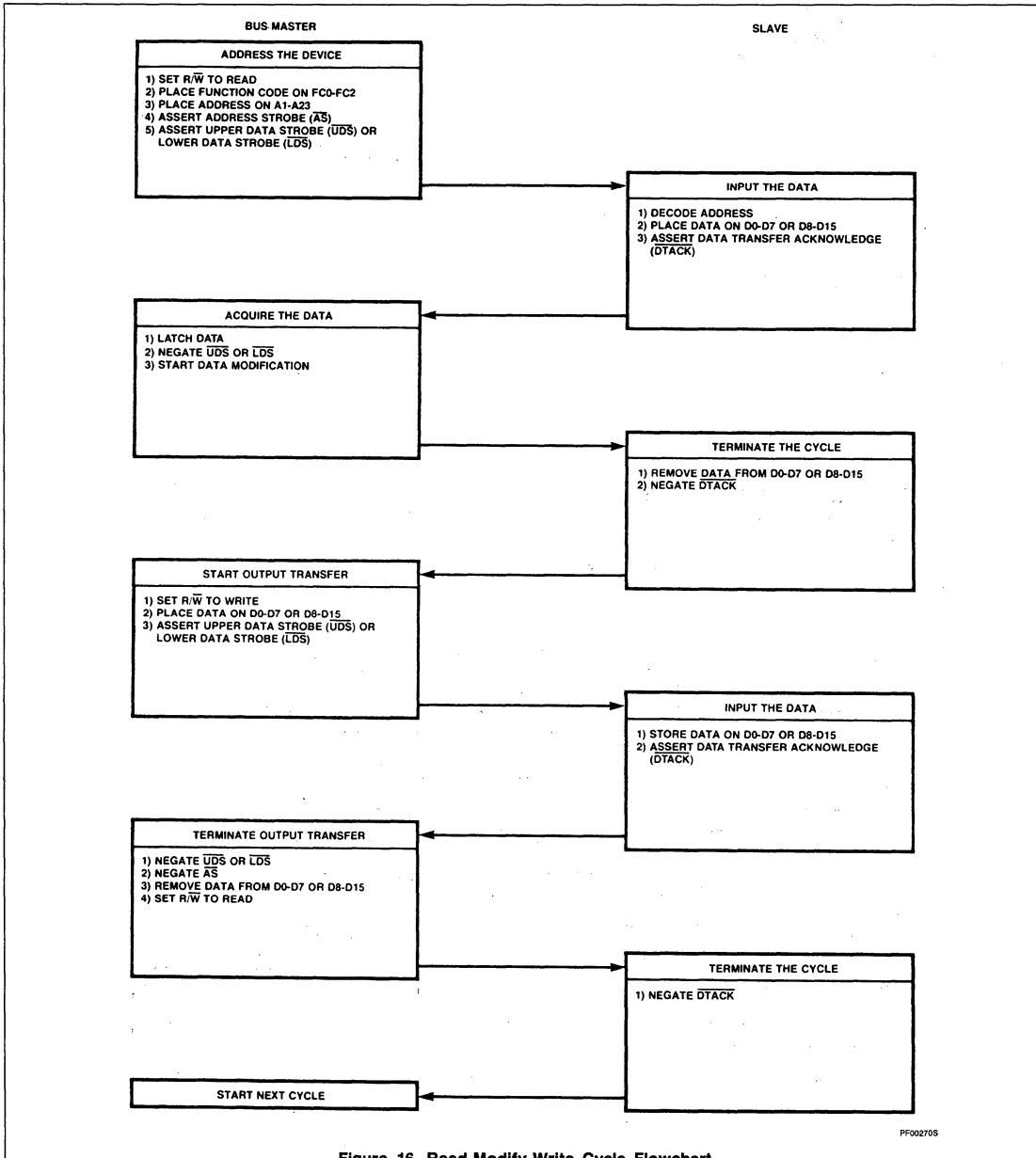


Figure 16. Read-Modify-Write Cycle Flowchart

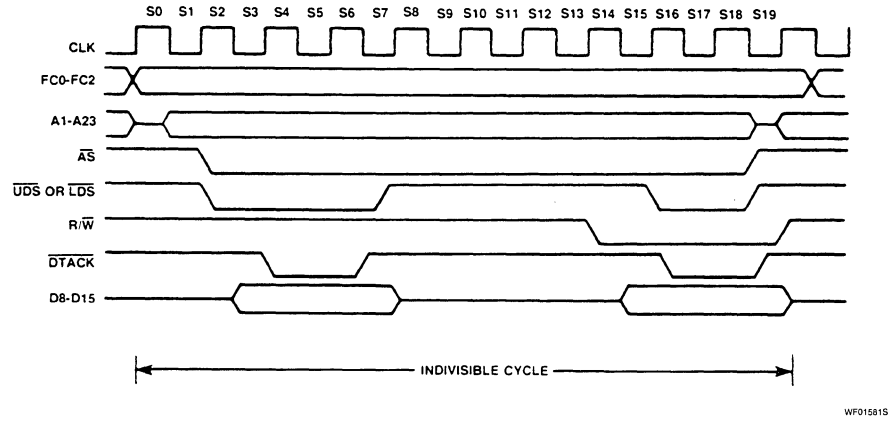


Figure 17. Read-Modify-Write Cycle Timing Diagram

Bus Arbitration

Bus arbitration is a technique used by master-type devices to request, be granted, and acknowledge bus mastership. In its simplest form, it consists of the following:

1. asserting a bus mastership request,
2. receiving a grant that the bus is available at the end of the current cycle, and
3. acknowledging that mastership has been assumed.

Figure 18 is a flowchart showing the detail involved in a request from a single device. Figure 19 is a timing diagram for the same operation. This technique allows processing of bus requests during data transfer cycles.

The timing diagram shows that the bus request is negated at the time that an acknowledge is asserted. This type of operation would be true for a system consisting of the processor and one device capable of bus mastership. In systems having a number of devices capable of bus mastership, the bus request line from each device is wire-ORed to the processor. In this system, it is easy to see that there could be more than one bus request being made. The timing diagram shows that the bus grant signal is negated a few clock cycles after the transition of the acknowledge ($\overline{BGACK}$) signal.

However, if the bus requests are still pending, the processor will assert another bus grant within a few clock cycles after it was negated. This additional assertion of bus grant allows external arbitration circuitry to select the next bus master before the current bus master has completed its requirements. The following paragraphs provide additional information about the three steps in the arbitration process.

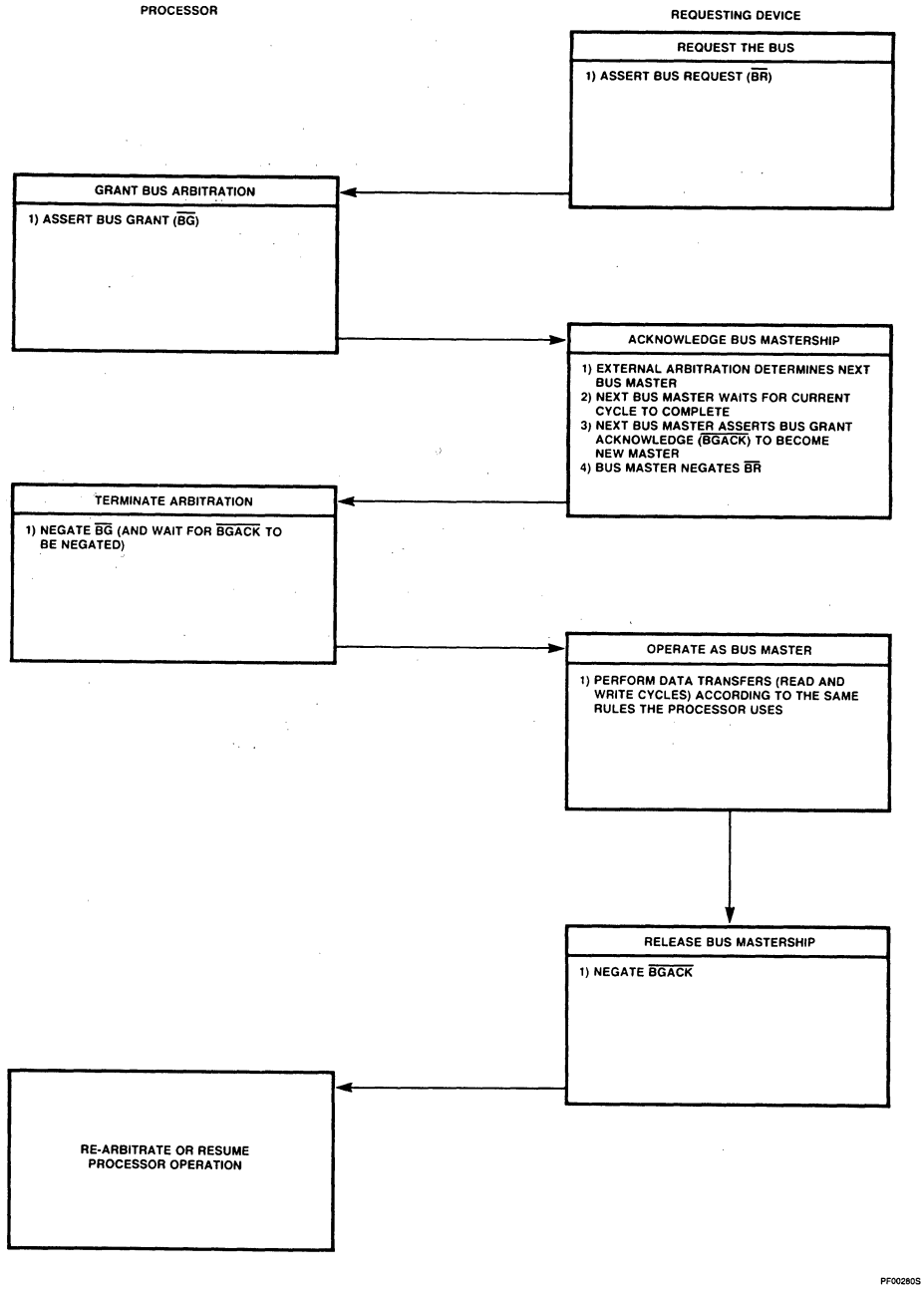


Figure 18. Bus Arbitration Cycle Flowchart

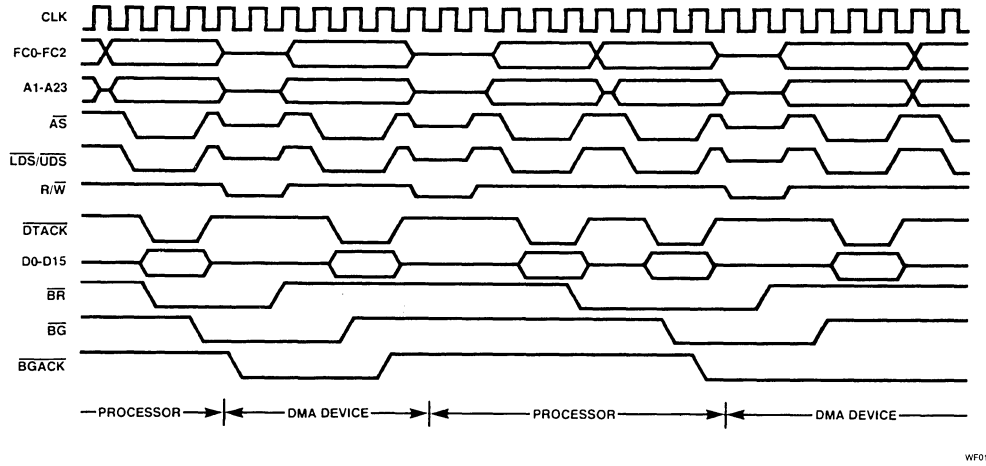


Figure 19. Bus Arbitration Cycle Timing Diagram

Requesting the Bus — External devices capable of becoming bus masters request the bus by asserting the bus request ($\overline{BR}$) signal. This is a wire-ORed signal (although it need not be constructed from open-collector devices) that indicates to the processor that some external device requires control of the external bus. The processor is effectively at a lower bus priority level than the external device and will relinquish the bus after it has completed the last bus cycle it has started.

When no acknowledge is received before the bus request signal goes inactive, the processor will continue processing when it detects that the bus request is inactive. This allows ordinary processing to continue if the arbitration circuitry responded to noise inadvertently.

Receiving the Bus Grant — The processor asserts bus grant ($\overline{BG}$) as soon as possible. Normally this is immediately after internal synchronization. The only exception to this occurs when the processor has made an internal decision to execute the next bus cycle but has not progressed far enough into the cycle to have asserted the address strobe ($\overline{AS}$) signal. In this case, bus grant will be delayed until $\overline{AS}$ is asserted to indicate to external devices that a bus cycle is being executed.

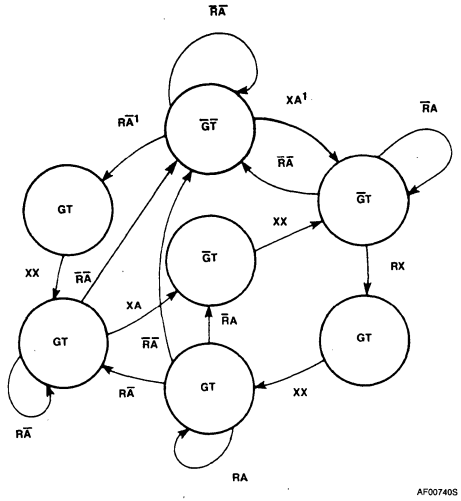
The bus grant signal may be routed through a daisy-chained network or through a specific priority-encoded network. The processor is not affected by the external method of arbitration as long as the protocol is obeyed.

Acknowledgement of Mastership — Upon receiving a bus grant, the requesting device waits until address strobe, data transfer acknowledge, and bus grant acknowledge are negated before issuing its own $\overline{BGACK}$. The negation of the address strobe indicates that the previous master has completed its cycle; the negation of bus grant acknowledge indicates that the previous master has released the bus. (While address strobe is asserted, no device is allowed to "break into" a cycle.) The negation of data transfer acknowledge indicates the previous slave has terminated its connection to the previous master. Note that in some applications data transfer acknowledge might not enter into this function. General purpose devices would then be connected such that they were only dependent on address strobe. When bus grant acknowledge is issued, the device is a bus master until it negates bus grant acknowledge. Bus grant acknowledge should not be negated until after the bus cycle(s) is (are) completed. Bus mastership is terminated at the negation of bus grant acknowledge.

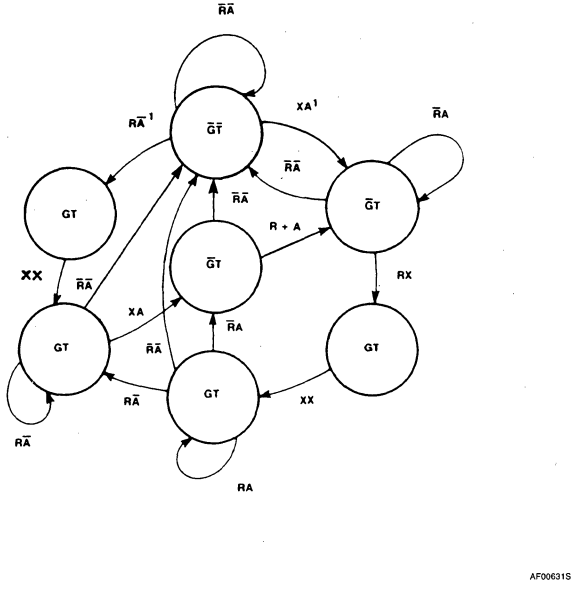
The bus request from the granted device should be dropped after bus grant acknowledge is asserted. If a bus request is still pending, another bus grant will be asserted within a few clocks of the negation of the bus grant. Refer to **Bus Arbitration Control.** Note that the processor does not perform any external bus cycles before it re-asserts bus grant.

Bus Arbitration Control

The bus arbitration control unit in the SCN68000 is implemented with a finite state machine. A state diagram of this machine is shown in figure 20. All asynchronous signals to the SCN68000 are synchronized before being used internally. This synchronization is accomplished in a maximum of one cycle of the system clock, assuming that the asynchronous input set-up time (#47) has been met (see figure 21). The input signal is sampled on the falling edge of the clock and is valid internally after the next falling edge.



a. State Diagram for Mask Sets Previous to GN7



b. State Diagram for GN7 and Later Mask Sets

NOTES:
R = Bus request internal
A = Bus grant acknowledge internal
G = Bus grant
T = Three-state control to bus control logic[2]
X = Don't care
1. State machine will not change if the bus is S0 or S1. Refer to Bus Arbitration Control.
2. The address bus will be placed in the high-impedance state if T is asserted and $\overline{AS}$ is negated.

Figure 20. SCN68000 Bus Arbitration Unit State Diagram

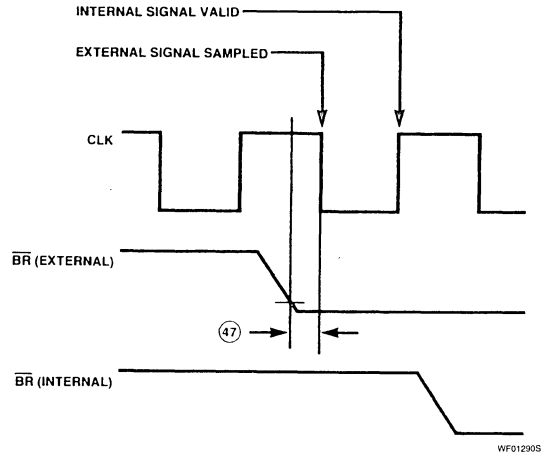


Figure 21. Timing Relationship of External Asynchronous Inputs to Internal Signals

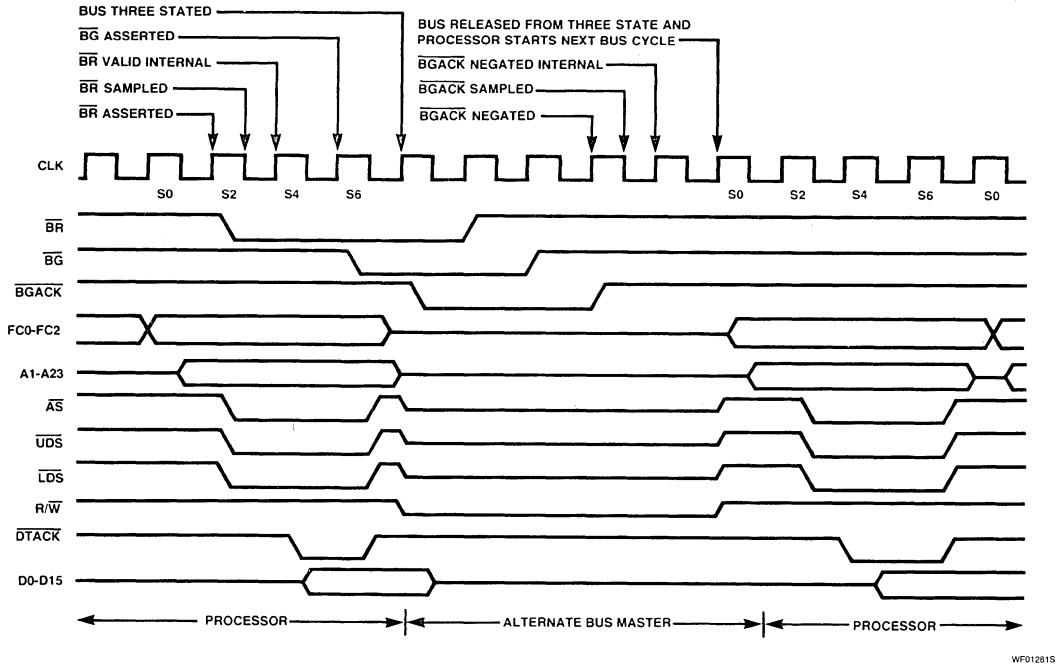


Figure 22. Bus Arbitration Timing Diagram—Processor Active

As shown in figure 20, input signals labeled R and A are internally synchronized on the bus request and bus grant acknowledge pins, respectively. The bus grant output is labeled G and the internal three-state control signal T. If T is true, the address, data, and control buses are placed in a high-impedance state when $\overline{AS}$ is negated. All signals are shown in positive logic (active high) regardless of their true active voltage level. State changes (valid outputs) occur on the next rising edge after the internal signal is valid.

A timing diagram of the bus arbitration sequence during a processor bus cycle is shown in figure 22. The bus arbitration sequence while the bus is inactive (i.e., executing internal operations such as a multiply instruction) is shown in figure 23.

If a bus request is made at a time when the MPU has already begun a bus cycle but $\overline{AS}$ has not been asserted (bus state S0), $\overline{BG}$ will not be asserted on the next rising edge. Instead, $\overline{BG}$ will be delayed until the second rising edge following its internal assertion. This sequence is shown in figure 24.

Bus Error and Halt Operation

In a bus architecture that requires a handshake from an external device, the possibility exists that the handshake might not occur. Since different systems will require a different maximum response time, a bus error input is provided. External circuitry must be used to determine the duration between address strobe and data transfer acknowledge before issuing a bus error signal. When a bus error signal is received, the processor has two options: initiate a bus error exception sequence or try running the bus cycle again.

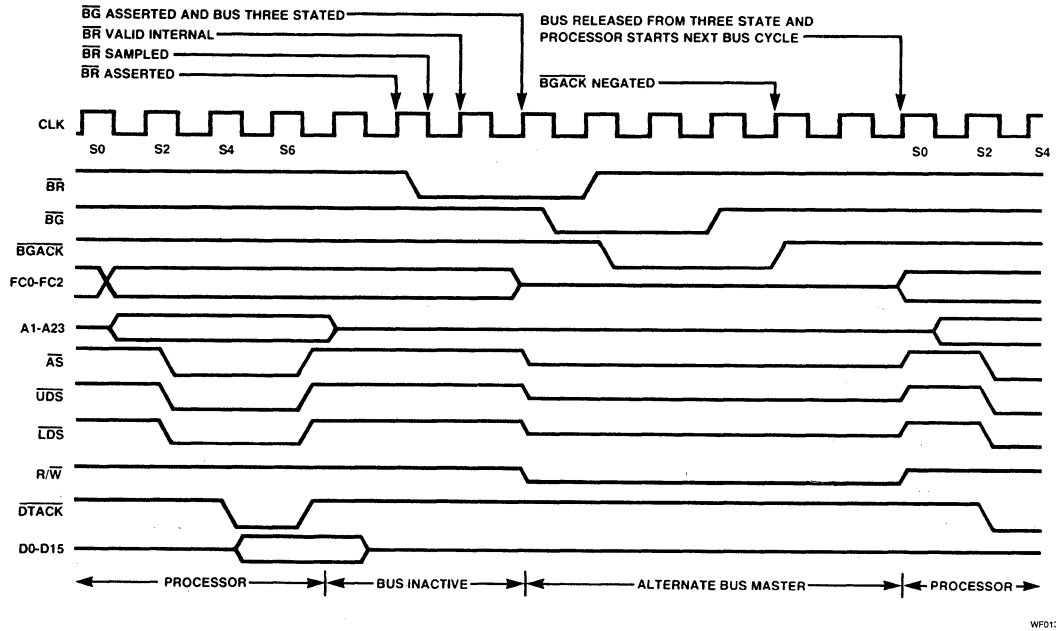


Figure 23. Bus Arbitration Timing Diagram—Bus Inactive

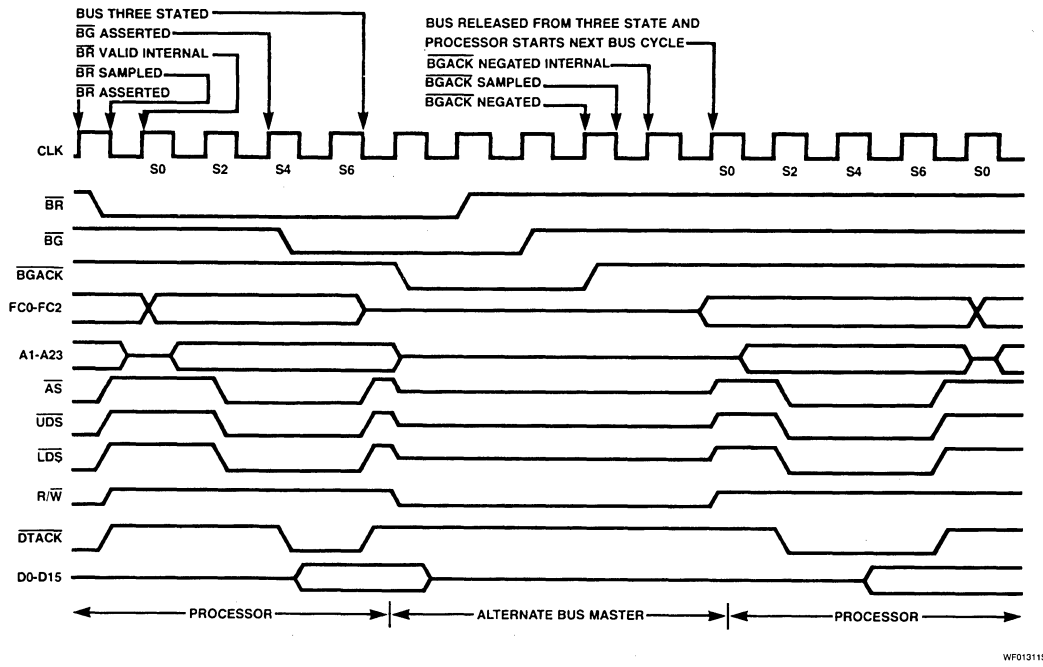


Figure 24. Bus Arbitration Timing Diagram—Special Case

Bus Error Operation — When the bus error signal is asserted, the current bus cycle is terminated. If $\overline{BERR}$ is asserted before the falling edge of S2, $\overline{AS}$ will be negated in S7 in either a read or write cycle. As long as $\overline{BERR}$ remains asserted, the data and address buses will be in the high-impedance state. When $\overline{BERR}$ is negated, the processor will begin stacking for exception processing. Figure 25 is a timing diagram for the exception sequence. The sequence is composed of the following elements:

1. stacking the program counter and status register,
2. stacking the error information,
3. reading the bus error vector table entry, and
4. executing the bus error handler routine.

The stacking of the program counter and the status register is the same as if an interrupt had occurred. Several additional items are stacked when a bus error occurs. These items are used to determine the nature of the error and correct it, if possible. The bus error vector is vector number two located at address $000008. The processor loads the new program counter from this location. A software bus error handler routine is then executed by the processor. Refer to **Exception Processing** for additional information.

Re-Run Operation — When, during a bus cycle, the processor receives a bus error signal and the halt pin is being driven by an external device, the processor enters the re-run sequence. Figure 26 is a timing diagram for re-running the bus cycle.

The processor terminates the bus cycle, then puts the address and data output lines in the high-impedance state. The processor remains "halted", and will not run another bus cycle until the halt signal is removed by external logic. Then the processor will re-run the previous cycle using the same function codes, the same data (for a write operation), and the same controls. The bus error signal should be removed at least one clock cycle before the halt signal is removed.

NOTE

The processor will not re-run a read-modify-write cycle. This restriction is made to guarantee that the entire cycle runs correctly and that the write operation of a test-and-set operation is performed without ever releasing $\overline{AS}$. If $\overline{BERR}$ and $\overline{HALT}$ are asserted during a read-modify-write bus cycle, a bus error operation results.

Halt Operation — The halt input signal to the SCN68000 performs a halt/run/single-step function in a similar fashion to the synchronous device halt function. The halt and run modes are somewhat self explanatory in that when the halt signal is constantly active the processor "halts" (does nothing) and when the halt signal is constantly inactive the processor "runs" (does something).

This single-step mode is derived from correctly timed transitions on the halt signal input. It forces the processor to execute a single bus cycle by entering the run mode until the processor starts a bus cycle then changing to the halt mode. Thus, the single-step mode allows the user to proceed through (and therefore debug) processor operations one bus cycle at a time.

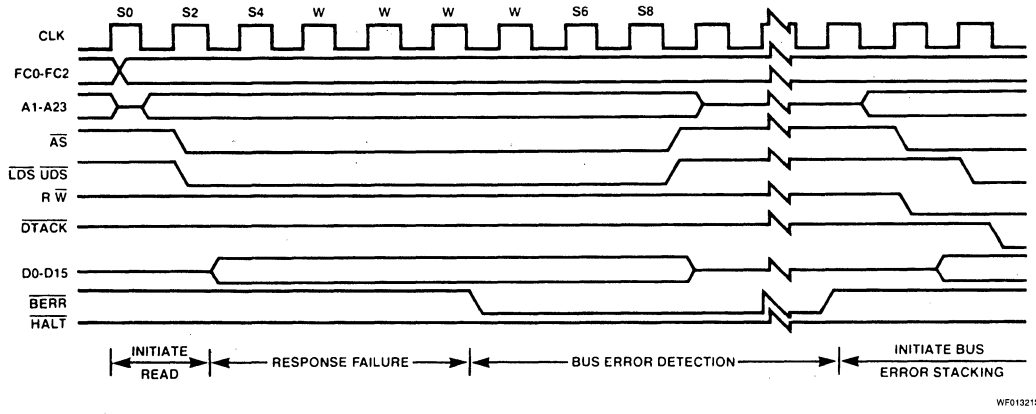


Figure 25. Bus Error Timing Diagram

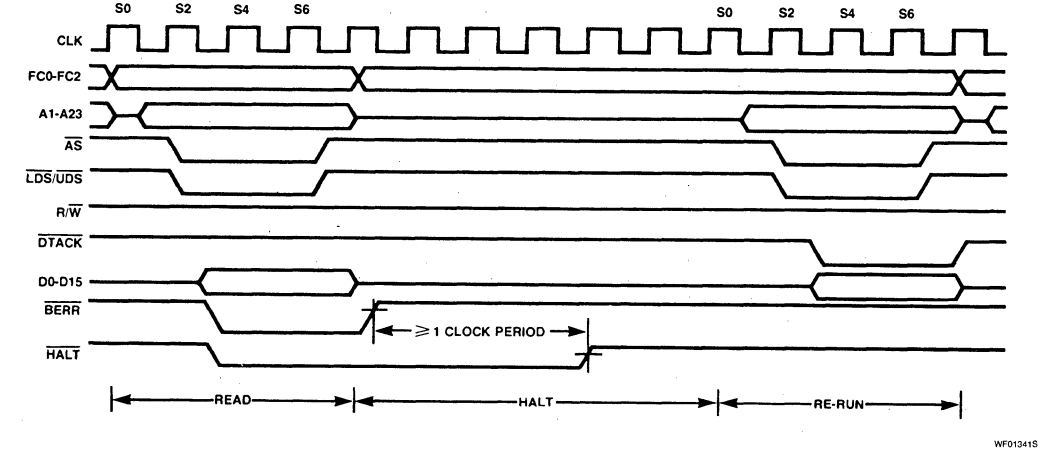


Figure 26. Re-Run Bus Cycle Timing Diagram

Figure 27 details the timing required for correct single-step operations. Some care must be exercised to avoid harmful interactions between the bus error signal and the halt pin when using the single-cycle mode as a debugging tool. This is also true of interactions between the halt and reset lines since these can reset the machine.

When the processor completes a bus cycle after recognizing that the halt signal is active, most three-state signals are put in the high-impedance state, these include:
1. address lines, and
2. data lines.

This is required for correct performance of the re-run bus cycle operation.

While the processor is honoring the halt request, bus arbitration performs as usual. That is, halting has no effect on bus arbitration. It is the bus arbitration function that removes the control signals from the bus.

The halt function and the hardware trace capability allow the hardware debugger to trace single bus cycles or single instructions at a time. These processor capabilities, along with a software debugging package, give total debugging flexibility.

Double Bus Faults — When a bus error exception occurs, the processor will attempt to stack several words containing information about the state of the machine. If a bus error exception occurs during the stacking operation, there have been two bus errors in a row. This is commonly referred to as a double bus fault. When a double bus fault occurs, the processor will halt. Once a bus error exception has occurred, any bus error exception occurring before the execution of the next instruction constitutes a double bus fault.

Note that a bus cycle which is re-run does not constitute a bus error exception and does not contribute to a double bus fault. Note also that this means that as long as the external hardware requests it, the processor will continue to re-run the same bus cycle.

The bus error pin also has an effect on processor operation after the processor receives an external reset input. The processor reads the vector table after a reset to determine the address to start program execution. If a bus error occurs while reading the vector table (or at any time before the first instruction is executed), the processor reacts as if a double bus fault has occurred and it halts. Only an external reset will start a halted processor.

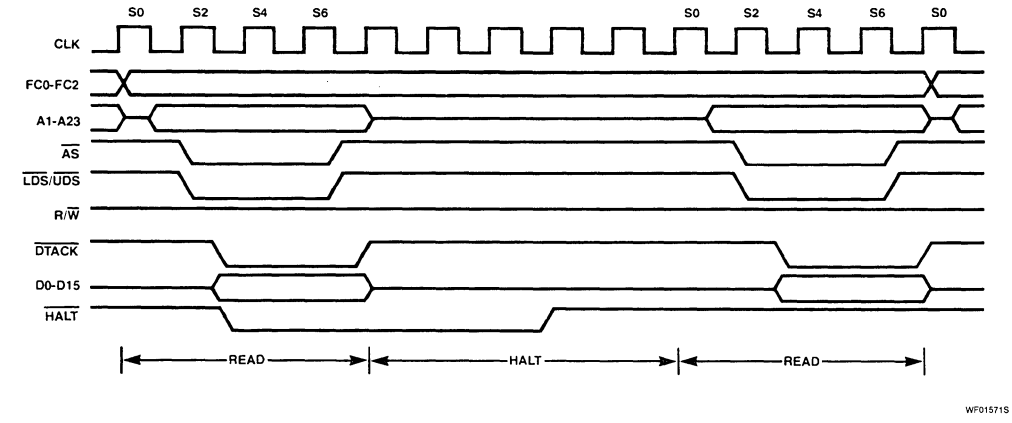


Figure 27. Halt Processor Timing Diagram

Reset Operation

The reset signal is a bidirectional signal that allows either the processor or an external signal to reset the system. Figure 28 is a timing diagram for the reset operation. Both the halt and reset lines must be asserted to ensure total reset of the processor.

When the reset and halt lines are driven by an external device, it is recognized as an entire system reset, including the processor. The processor responds by reading the reset vector table entry (vector number zero, address $000000) and loads it into the supervisor stack pointer (SSP). Vector table entry number one at address $000004 is read next and loaded into the program counter. The processor initializes the status register to an interrupt level of seven. No other registers are affected by the reset sequence.

When a reset instruction is executed, the processor drives the reset pin for 124 clock periods. In this case, the processor is trying to reset the rest of the system. Therefore, there is no effect on the internal state of the processor. All of the processor's internal registers and the status register are unaffected by the execution of a reset instruction. All external devices connected to the reset line will be reset at the completion of the reset instruction.

Asserting the reset and halt lines for ten clock cycles will cause a processor reset, except when V_{CC} is initially applied to the processor. In this case, an external reset must be applied for at least 100 milliseconds.

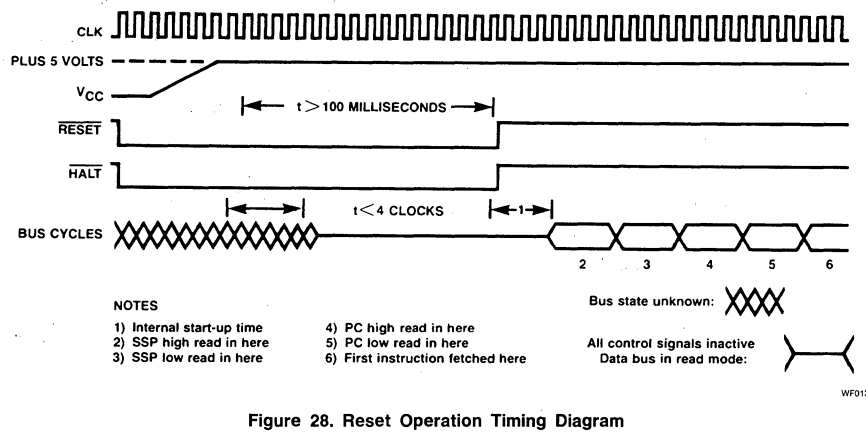


Figure 28. Reset Operation Timing Diagram

The Relationship of $\overline{DTACK}$, $\overline{BERR}$, and $\overline{HALT}$

In order to properly control termination of a bus cycle for a re-run or a bus error condition, $\overline{DTACK}$, $\overline{BERR}$, and $\overline{HALT}$ should be asserted and negated on the rising edge of the SCN68000 clock. This will assure that when two signals are asserted simultaneously, the required set-up time (#47) for both of them will be met during the same bus state.

This, or some equivalent precaution, should be designed external to the SCN68000. Parameter #48 (see AC Electrical Characteristics for # references) is intended to ensure this operation in a totally asynchronous system, and may be ignored if the above conditions are met.

The preferred bus cycle terminations may be summarized as follows (case numbers refer to table 16):

Normal Termination:
$\overline{DTACK}$ occurs first (case 1).

Halt Termination:
$\overline{HALT}$ is asserted at the same time or before $\overline{DTACK}$ and $\overline{BERR}$ remains negated (cases 2 and 3).

Bus Error Termination:
$\overline{BERR}$ is asserted in lieu of, at the same time, or before $\overline{DTACK}$ (case 4); $\overline{BERR}$ is negated at the same time or after $\overline{DTACK}$.

Re-Run Termination:
$\overline{HALT}$ and $\overline{BERR}$ are asserted in lieu of, at the same time, or before $\overline{DTACK}$ (cases 6 and 7); $\overline{HALT}$ must be held at least one cycle after $\overline{BERR}$. Case 5 indicates $\overline{BERR}$ may precede $\overline{HALT}$.

Table 16 details the resulting bus cycle termination under various combinations of control signal sequences. The negation of these same control signals under several conditions is shown in table 17 ($\overline{DTACK}$ is assumed to be negated normally in all cases; for best results, both $\overline{DTACK}$ and $\overline{BERR}$ should be negated when address strobe is negated).

EXAMPLE A:
A system uses a watch-dog timer to terminate accesses to unpopulated address space. The timer asserts $\overline{DTACK}$ and $\overline{BERR}$ simultaneously after time out (case 4).

EXAMPLE B:
A system uses error detection on RAM contents. Designer may (a) delay $\overline{DTACK}$ until data verified and return $\overline{BERR}$ and $\overline{HALT}$ simultaneously to re-run error cycle (case 6), or if valid, return $\overline{DTACK}$ (case 1); (b) delay $\overline{DTACK}$ until data verified and return $\overline{BERR}$ at same time as $\overline{DTACK}$ if data in error (case 4).

Table 16. $\overline{DTACK}$, $\overline{BERR}$, AND $\overline{HALT}$ ASSERTION RESULTS

CASE NO.	CONTROL SIGNAL	ASSERTED ON RISING EDGE OF STATE		RESULT
		N	N + 2	
1	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	S X X	Normal cycle terminate and continue.
2	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA A	S X S	Normal cycle terminate and halt. Continue when $\overline{HALT}$ removed.
3	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	NA NA A	A NA S	Normal cycle terminate and halt. Continue when $\overline{HALT}$ removed.
4	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	X A NA	X S NA	Terminate and take bus error trap.
5	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	NA A NA	X S A	Terminate and re-run.
6	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	X A A	X S S	Terminate and re-run when $\overline{HALT}$ removed.
7	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	NA NA A	X A S	Terminate and re-run when $\overline{HALT}$ removed.

NOTES:
N–the number of the current even bus state (e.g., S4, S6, etc.)
A–signal is asserted in this bus state
NA–signal is not asserted in this state
X–don't care
S–signal was asserted in previous state and remains asserted in this state

Table 17. $\overline{BERR}$ AND $\overline{HALT}$ NEGATION RESULTS

CONDITIONS OF TERMINATION IN TABLE 16	CONTROL SIGNAL	NEGATED ON RISING EDGE OF STATE		RESULTS–NEXT CYCLE
		N	N+2	
Bus Error	$\overline{BERR}$ $\overline{HALT}$	• or • • or •		Takes bus error trap.
Re-run	$\overline{BERR}$ $\overline{HALT}$	• or •	•	Illegal sequence; usually traps to vector number 0.
Re-run	$\overline{BERR}$ $\overline{HALT}$	•	•	Re-runs the bus cycle.
Normal	$\overline{BERR}$ $\overline{HALT}$	• • or	•	May lengthen next cycle.
Normal	$\overline{BERR}$ $\overline{HALT}$	• or	• none	If next cycle is started it will be terminated as a bus error.

• = Signal is negated in this bus state.

Asynchronous Versus Synchronous Operation

Asynchronous Operation

To achieve clock frequency independence at a system level, the SCN68000 can be used in an asynchronous manner. This entails using only the bus handshake lines ($\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, $\overline{DTACK}$, $\overline{BERR}$, $\overline{HALT}$, and $\overline{VPA}$) to control the data transfer. Using this method, $\overline{AS}$ signals the start of a bus cycle and the data strobes are used as a condition for valid data

on a write cycle. The slave device (memory or peripheral) then responds by placing the requested data on the data bus for a read cycle or latching data on a write cycle and asserting the data transfer acknowledge signal ($\overline{DTACK}$) to terminate the bus cycle. If no slave responds or the access is invalid, external control logic asserts the $\overline{BERR}$, or $\overline{BERR}$ and $\overline{HALT}$, signal to abort or rerun the bus cycle.

The $\overline{DTACK}$ signal is allowed to be asserted before the data from a slave device is valid on a read cycle. The length of time that $\overline{DTACK}$ may precede data is given as parameter #31 and it must be met in any asynchronous system to insure that valid data is latched into the processor. Notice that there is no maximum time specified from the assertion of $\overline{AS}$ to the assertion of $\overline{DTACK}$. This is because the MPU will insert wait cycles of one clock period each until $\overline{DTACK}$ is recognized.

Synchronous Operation

To allow for those systems which use the system clock as a signal to generate $\overline{DTACK}$ and other asynchronous inputs, the asynchronous input set-up time is given as parameter #47. If this set-up is met on an input, such as $\overline{DTACK}$, the processor is guaranteed to recognize that signal on the next falling edge of the system clock. However, the converse is not true — if the input signal does not meet the set-up time it is not guaranteed not to be recognized. In addition, if $\overline{DTACK}$ is recognized on a falling edge, valid data will be latched into the processor (on a read cycle) on the next falling edge provided that the data meets the set-up time given as parameter #27. Given this, parameter #31 may be ignored. Note that if $\overline{DTACK}$ is asserted, with the required set-up time, before the falling edge of S4, no wait states will be incurred and the bus cycle will run at its maximum speed of four clock periods.

NOTE

During an active bus cycle, $\overline{BERR}$ is sampled on every falling edge of the clock starting with S2. $\overline{DTACK}$ is sampled on every falling edge of the clock starting with S4 and data is latched on the falling edge of S6 during a read. The bus cycle will then be terminated in S7 except when $\overline{BERR}$ is asserted in the absence of $\overline{DTACK}$, in which case it will terminate one clock cycle later in S9. $\overline{VPA}$ is sampled only on the third falling edge of the system clock before the rising edge of the E clock.

PROCESSING STATES

This section describes the actions of the SCN68000 which are outside the normal processing associated with the execution of instructions. The functions of the bits in the supervisor portion of the status register are covered: the supervisor/user bit, the trace enable bit, and the processor interrupt priority mask. Finally, the sequence of memory references and actions taken by the processor on exception conditions are detailed.

The SCN68000 is always in one of three processing states: normal, exception, or halted. The normal processing state is that associated with instruction execution; the memory references are to fetch instructions and operands, and to store results. A special case of the normal state is the stopped state which the processor enters when a stop instruction is executed. In this state, no further references are made.

The exception processing state is associated with interrupts, trap instructions, tracing, and other exceptional conditions. The exception may be internally generated by an instruction or by an unusual condition arising during the execution of an instruction. Externally, exception processing can be forced by an interrupt, by a bus error, or by a reset. Exception processing is designed to provide an efficient context switch so that the processor may handle unusual conditions.

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts. Only an external reset can restart a halted processor. Note that a processor in the stopped state is not in the halted state, nor vice versa.

Privilege States

The processor operates in one of two states of privilege: the "supervisor" state or the "user" state. The privilege state determines which operations are legal, are used to choose between the supervisor stack pointer and the user stack pointer in instruction references, and may be used by an external memory management device to control and translate accesses.

The privilege state is a mechanism for providing security in a computer system. Programs should access only their own code and data areas, and ought to be restricted from accessing information which they do not need and must not modify.

The privilege mechanism provides security by allowing most programs to execute in user state. In this state, the accesses are controlled, and the effects on other parts of the system are limited. The operating system executes in the supervisor state, has access to all resources, and performs the overhead tasks for the user state programs.

Supervisor State

The supervisor state is the higher state of privilege. For instruction execution, the supervisor state is determined by the S bit of the status register; if the S bit is asserted (high), the processor is in the supervisor state. All instructions can be executed in the supervisor state. The bus cycles generated by instructions executed in the supervisor state are classified as supervisor references. While the processor is in the supervisor privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly access the supervisor stack pointer.

All exception processing is done in the supervisor state, regardless of the setting of the S bit. The bus cycles generated during exception processing are classified as supervisor references. All stacking operations during exception processing use the supervisor stack pointer.

User State

The user state is the lower state of privilege. For instruction execution, the user state is determined by the S bit of the status register; if the S bit is negated (low), the processor is executing instructions in the user state.

Most instructions execute the same in user state as in the supervisor state. However, some instructions which have important system effects are made privileged. User programs are not permitted to execute the stop instruction or the reset instruction. To ensure that a user program cannot enter the supervisor state except in a controlled manner, the instructions which modify the whole state register are privileged. To aid in debugging programs which are to be used as operating systems, the move to user stack pointer (MOVE to USP) and move from user stack pointer (MOVE from USP) instructions are also privileged.

The bus cycles generated by an instruction executed in the user state are classified as user state references. This allows an external memory management device to translate the address and to control access to protected portions of the address space. While the processor is in the user privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly, access the user stack pointer.

Privilege State Changes

Once the processor is in the user state and executing instructions, only exception processing can change the privilege state. During exception processing, the current setting of the S bit of the status register is saved and the S bit is asserted, putting the processor in the supervisor state. Therefore, when instruction execution resumes at the address specified to process the exception, the processor is in the supervisor privilege state.

Table 18. BUS CYCLE CLASSIFICATION

FUNCTION CODE OUTPUT			REFERENCE CLASS
FC2	FC1	FC0	
0	0	0	(Unassigned)
0	0	1	User data
0	1	0	User program
0	1	1	(Unassigned)
1	0	0	(Unassigned)
1	0	1	Supervisor data
1	1	0	Supervisor program
1	1	1	Interrupt acknowledge

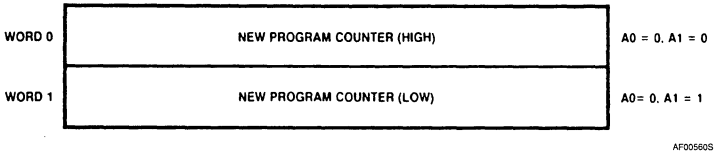
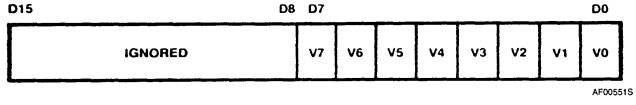


Figure 29. Format of Vector Table Entries



Where:
V7 is the MSB of the vector number.
V8 is the LSB of the vector number.

Figure 30. Vector Number Format

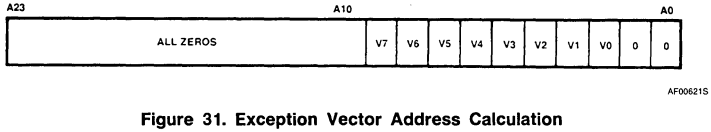


Figure 31. Exception Vector Address Calculation

Reference Classification

When the processor makes a reference, it classifies the kind of reference being made, using the encoding on the three function code output lines. This allows external translation of addresses, control of access, and differentiation of special processor state, such as interrupt acknowledge. Table 18 lists the classification of references.

Exception Processing

Before discussing the details of interrupts, traps, and tracing, a general description of exception processing is in order. The processing of an exception occurs in four steps, with variations for different exception causes. During the first step, a temporary copy of the status register is made and the status register is set for exception processing. In the second step the exception vector is determined and the third step is the saving of the current processor context. In the fourth step a new context is obtained and the processor switches to instruction processing.

Exception Vectors

Exception vectors are memory locations from which the processor fetches the address of a routine which will handle that exception. All exception vectors are two words in length (figure 29), except for the reset vector which is four words. All exception vectors lie in the supervisor data space, except for the reset vector which is in the supervisor program space. A vector number is an 8-bit number which, when multiplied by four, gives the address of an exception vector. Vector numbers are generated internally or externally, depending on the cause of the exception. In the case of interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (figure 30) to the processor on data bus lines D0 through D7. The processor translates the vector number into a full 24-bit address, shown in figure 31. The memory layout for exception vectors is given in table 19.

Table 19. EXCEPTION VECTOR TABLE

VECTOR NUMBER(S)	ADDRESS			ASSIGNMENT
	Dec	Hex	Space	
0	0	000	SP	Reset: initial SSP
–	4	004	SP	Reset: initial PC
2	8	008	SD	Bus error
3	12	00C	SD	Address error
4	16	010	SD	Illegal instruction
5	20	014	SD	Zero divide
6	24	018	SD	CHK instruction
7	28	01C	SD	TRAPV instruction
8	32	020	SD	Privilege violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 emulator
11	44	02C	SD	Line 1111 emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15	60	03C	SD	Uninitialized interrupt vector
16 – 23*	64	04C	SD	(Unassigned, reserved)
	95	05F		–
24	96	060	SD	Spurious interrupt
25	100	064	SD	Level 1 interrupt autovector
26	104	068	SD	Level 2 interrupt autovector
27	108	06C	SD	Level 3 interrupt autovector
28	112	070	SD	Level 4 interrupt autovector
29	116	074	SD	Level 5 interrupt autovector
30	120	078	SD	Level 6 interrupt autovector
31	124	07C	SD	Level 7 interrupt autovector
32 – 47	128	080	SD	TRAP instruction vectors
	191	0BF		–
48 – 63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		–
64 – 255	256	100	SD	User interrupt vectors
	1023	3FF		–

*Vector numbers 12, 13, 14, 16 through 23, and 48 through 63 are reserved for future enhancements by Signetics. No user peripheral devices should be assigned these numbers.

As shown in table 19, the memory layout is 512 words long (1024 bytes). It starts at address 0 and proceeds through address 1023. This provides 255 unique vectors; some of these are reserved for TRAPs and other system functions. Of the 255, there are 192 reserved for user interrupt vectors. However, there is no protection on the first 64 entries, so user interrupt vectors may overlap at the discretion of the systems designer.

Kinds of Exceptions

Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts and the bus error and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset inputs are used for access control and processor restart. The internally generated exceptions come from instructions, or from address errors or tracing. The trap (TRAP), trap on overflow (TRAPV), check data register against upper bounds (CHK), and divide (DIV) instructions all can generate exceptions as part of their instruction execution. In addition, illegal instructions, word fetches from odd addresses, and privilege violations cause exceptions. Tracing behaves like a very high-priority internally-generated interrupt after each instruction execution.

Exception Processing Sequence

Exception processing occurs in four identifiable steps. In the first step, an internal copy is made of the status register. After the copy is made, the S bit is asserted, putting the processor into the supervisor privilege state. Also, the T bit is negated which will allow the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor fetch and classified as an interrupt acknowledge. For all other exceptions, internal logic provides the vector number. This vector number is then used to generate the address of the exception vector.

The third step is to save the current processor status, except for the reset exception. The current program counter value and the saved copy of the status register are stacked using the supervisor stack pointer as shown in figure 32. The program counter value stacked usually points to the next unexecuted instruction; however, for bus error and address error, the value stacked for the program counter is unpredictable, and may be incremented from the address of the instruction which caused the error. Additional information defining the current context is stacked for the bus error and address error exceptions.

The last step is the same for all exceptions. The new program counter value is fetched from the exception vector. The processor then resumes instruction execution. The instruction at the address given in the exception vector is fetched, and normal instruction decoding and execution is started.

Multiple Exceptions

These paragraphs describe the processing which occurs when multiple exceptions arise simultaneously. Exceptions can be grouped according to their occurrence and priority. The group 0 exceptions are reset, bus error, and address error. These exceptions cause the instruction currently being executed to be aborted and the exception processing to commence within two clock cycles.

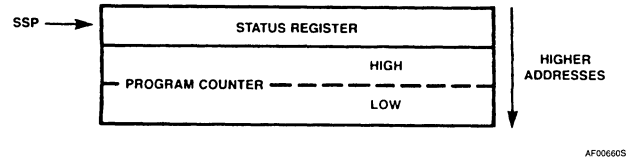


Figure 32. Exception Stack Order (Groups 1 and 2)

Table 20. EXCEPTION GROUPING AND PRIORITY

GROUP	EXCEPTION	PROCESSING
0	Reset address error bus error	Exception processing begins within two clock cycles
1	Trace interrupt illegal privilege	Exception processing begins before the next instruction
2	TRAP, TRAPV, CHK, zero divide	Exception processing is started by normal instruction execution

The group 1 exceptions are trace and interrupt, as well as the privilege violations and illegal instructions. These exceptions allow the current instruction to execute to completion, but pre-empt the execution of the next instruction by forcing exception processing to occur (privilege violations and illegal instructions are detected when they are the next instruction to be executed). The group 2 exceptions occur as part of the normal processing of instructions. The TRAP, TRAPV, CHK, and zero divide exceptions are in this group. For these exceptions, the normal execution of an instruction may lead to exception processing.

Group 0 exceptions have highest priority, while group 2 exceptions have lowest priority. Within group 0, reset has highest priority, followed by bus error and then address error. Within group 1, trace has priority over external interrupts, which in turn takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, there is no priority relation within group 2.

The priority relation between two exceptions determines which is taken, or taken first, if the conditions for both arise simultaneously. Therefore, if a bus error occurs during a TRAP instruction, the bus error takes precedence, and the TRAP instruction processing is aborted. In another example, if an interrupt request occurs during the execution of an instruction while the T bit is asserted, the trace exception has priority, and is processed first. Before instruction processing resumes, however, the interrupt exception is also processed, and instruction processing commences finally in the interrupt handler routine. A summary of exception grouping and priority is given in table 20.

Exception Processing Detailed Discussion

Exceptions have a number of sources and each exception has processing which is peculiar to it. The following paragraphs detail the sources of exceptions, how each arises, and how each is processed.

Reset

The reset input provides the highest exception level. The processing of the reset signal is designed for system initiation and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state and the trace state is forced off. The processor interrupt priority mask is set at level seven. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the supervisor stack pointer, neither the program counter nor the status register is saved. The address contained in the first two words of the reset exception vector is fetched as the initial supervisor stack pointer, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, instruction execution is started at the address in the program counter. The power-up/restart code should be pointed to by the initial program counter.

The reset instruction does not cause loading of the reset vector, but does assert the reset line to reset external devices. This allows the software to reset the system to a known state and then continue processing with the next instruction.

Interrupts

Seven levels of interrupt priorities are provided. Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. Interrupt priority levels are numbered from one to seven, with level seven being the highest priority. The status register contains a 3-bit mask which indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

An interrupt request is made to the processor by encoding the interrupt request level on the interrupt request lines; a zero indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception processing is postponed. (The recognition of level seven is slightly different, as explained in the following paragraph.)

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. A copy of the status register is saved, the privilege state is sent to the supervisor stack, tracing is suppressed, and the processor priority level is set to the level of the interrupt acknowledged. The processor fetches the vector number from the interrupting device, classifying the reference as an interrupt acknowledge and displaying the level number of the interrupt being acknowledged on the address bus. If external logic requests an automatic vectoring, the processor internally generates a vector number which is determined by the interrupt level number. If external logic indicates a bus error, the interrupt is taken to be spurious, and the generated vector number references the spurious interrupt vector. The processor then proceeds with the usual exception processing, saving the program counter and status register on the supervisor stack. The saved value of the program counter is the address of the instruction which would have been executed had the interrupt not been present. The content of the interrupt vector whose vector number was previously obtained is fetched and loaded into the program counter, and normal instruc-

tion execution commences in the interrupt handling routine. A flowchart for the interrupt acknowledge sequence is given in figure 33, a timing diagram is given in figure 34, and the interrupt processing sequence is shown in figure 35.

Priority level seven is a special case. Level seven interrupts cannot be inhibited by the interrupt priority mask, thus providing a "non-maskable interrupt" capability. An interrupt is generated each time the interrupt request level changes from some lower level to level seven. Note that a level seven interrupt may still be caused by the level comparison if the request level is a seven and the processor priority is set to a lower level by an instruction.

Uninitialized Interrupt

An interrupting device asserts $\overline{VPA}$ or provides an interrupt during an interrupt acknowledge cycle to the SCN68000. If the vector register has not been initialized, the responding S68000 family peripheral will provide vector 15, the uninitialized interrupt vector. This provides a uniform way to recover from a programming error.

Spurious Interrupt

If during the interrupt acknowledge cycle no device responds by asserting $\overline{DTACK}$ or $\overline{VPA}$, the bus error line should be asserted to terminate the vector acquisition. The processor separates the processing of this error from bus error by fetching the spurious interrupt vector instead of the bus error vector. The processor then proceeds with the usual exception processing.

Instruction Traps

Traps are exceptions caused by instructions. They arise either from processor recognition of abnormal conditions during instruction execution, or from use of instructions whose normal behavior is trapping.

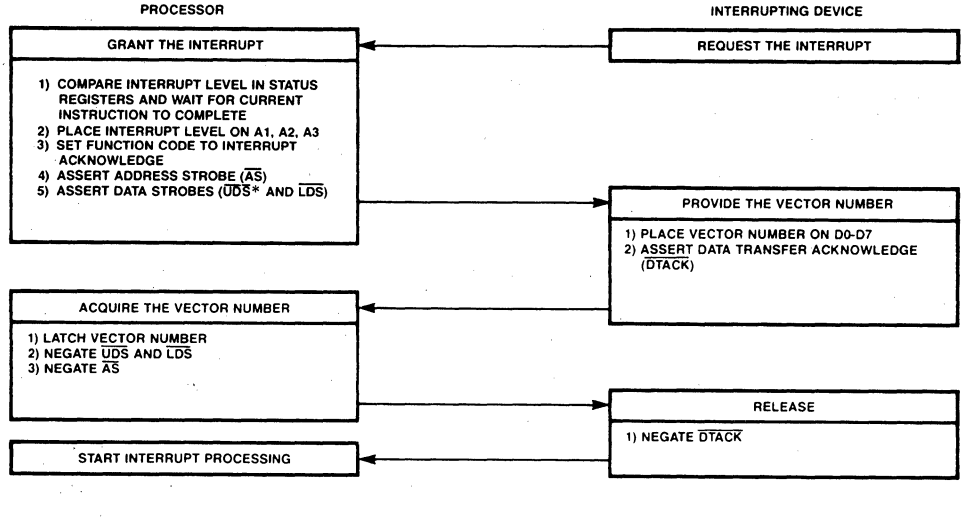
Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception and is useful for implementing system calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a runtime error, which may be an arithmetic overflow or a subscript out of bounds.

The signed divide (DIVS) and unsigned (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

Illegal and Unimplemented Instructions

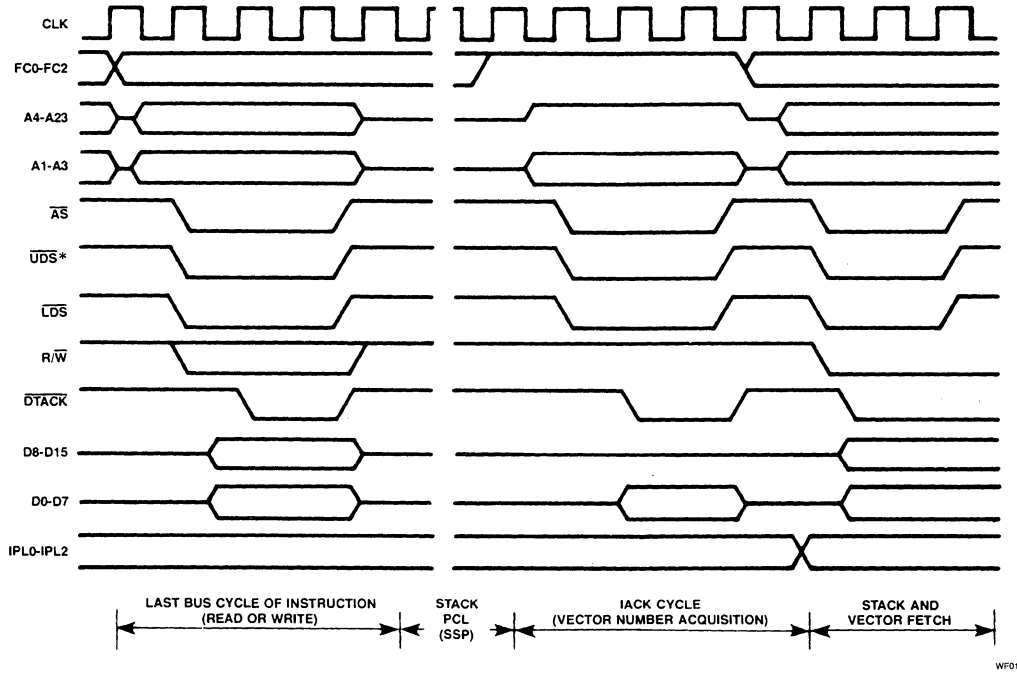
"Illegal instruction" is the term used to refer to any of the word bit patterns which are not the bit pattern of the first word of a legal instruction. During instruction execution, if such an instruction is fetched, an illegal instruction exception occurs. Signetics reserves the right to define instructions whose opcodes may be any of the illegal instructions. Three bit patterns will always force an illegal instruction trap on all S68000 family compatible microprocessors. They are: $4AFA, $4AFB, and $4AFC. Two of the patterns, $4AFA and $4AFB, are reserved for Signetics systems products. The third pattern, $4AFC, is reserved for customer use.

Word patterns with bits 15 through 12 equaling 1010 or 1111 are distinguished as unimplemented instructions and separate exception vectors are given to these patterns to permit efficient emulation. This facility allows the operating system to detect program errors, or to emulate unimplemented instructions in software.



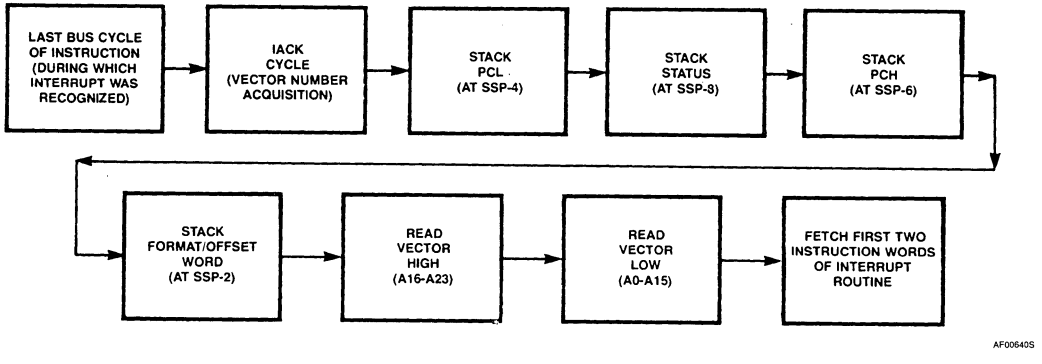
*Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D8 through D15 at this time.

Figure 33. Vector Acquisition Flowchart



*Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D8 through D15 at this time.

Figure 34. Interrupt Acknowledge Cycle Timing Diagram



NOTE:
SSP refers to the value of the supervisor stack pointer before the interrupt occurs.

Figure 35. Interrupt Processing Sequence

Privilege Violations

In order to provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user state will cause an exception. The privileged instructions are:

STOP	AND Immediate to SR
RESET	EOR Immediate to SR
RTE	OR Immediate to SR
MOVE to SR	MOVE to USP

Tracing

To aid in program development, the SCN68000 includes a facility to allow instruction-by-instruction tracing. In the trace state, after each instruction is executed an exception is forced, allowing a debugging program to monitor the execution of the program under test.

The trace facility uses the T bit in the supervisor portion of the status register. If the T bit is negated (off), tracing is disabled, and instruction execution proceeds from instruction to instruction as normal. If the T bit is asserted (on) at the beginning of the execution of an instruction, a trace exception will be generated after the execution of that instruction is completed. If the instruction is not executed, either because an interrupt is taken, or the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus error, or address error exception. If the instruction is indeed executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. If, during the execution of the instruction an exception is forced by that instruction, the forced exception is processed before the trace exception.

As an extreme illustration of the above rules, consider the arrival of an interrupt during the execution of a TRAP instruction while tracing is enabled. First the trap exception is processed, then the trace exception, and finally the interrupt exception. Instruction execution resumes in the interrupt handler routine.

Bus Error

Bus error exceptions occur when the external logic requests that a bus error be processed by an exception. The current bus cycle which the processor is making is then aborted. Whether the processor was doing instruction or exception processing, that processing is terminated, and the processor immediately begins exception processing.

Exception processing for the bus error follows the usual sequence of steps. The status register is copied, the supervisor state is entered, and the trace state is turned off. The vector number is generated to refer to the bus error vector. Since the processor was not between instructions when the bus error exception request was made, the context of the processor is more detailed. To save more of this context, additional information is saved on the supervisor stack. The program counter and the copy of the status register are of course saved. The value saved for the program counter is advanced by some amount, one to five words beyond the address of the first word of the instruction which made the reference causing the bus error. If the bus error occurred during the fetch of the next instruction, the saved program counter has a value in the vicinity of the current instruction, even if the current instruction is a branch, a jump, or a return instruction. Besides the usual information, the processor saves its internal copy of the first word of the instruction being processed and the address which was being accessed by the aborted bus cycle. Specific information about the access is also saved: whether it was a read or a write, whether or not the processor was processing an instruction, and the classification displayed on the function code outputs when the bus error occurred. The processor is processing an instruction if it is in the normal state or processing a group 2 exception; the processor is not processing an instruction if it is processing a group 0 or a group 1 exception. Figure 36 illustrates how this information is organized on the supervisor stack. Although this information is not sufficient in general to effect full recovery from the bus error, it does allow software diagnosis. Finally, the processor commences instruction processing at the address contained in vector number two. It is the responsibility of the error handler routine to clean up the stack and determine where to continue execution.

If a bus error occurs during the exception processing for a bus error, address error, or reset, the processor is halted and all processing ceases. This simplifies the detection of catastrophic system failure, since the processor removes itself from the system rather than destroy any memory contents. Only the $\overline{\text{RESET}}$ pin can restart a halted processor.

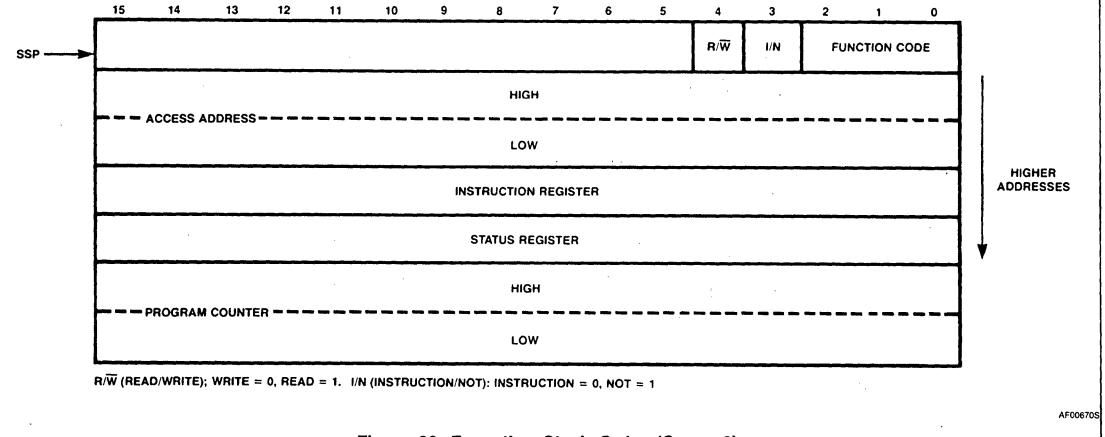


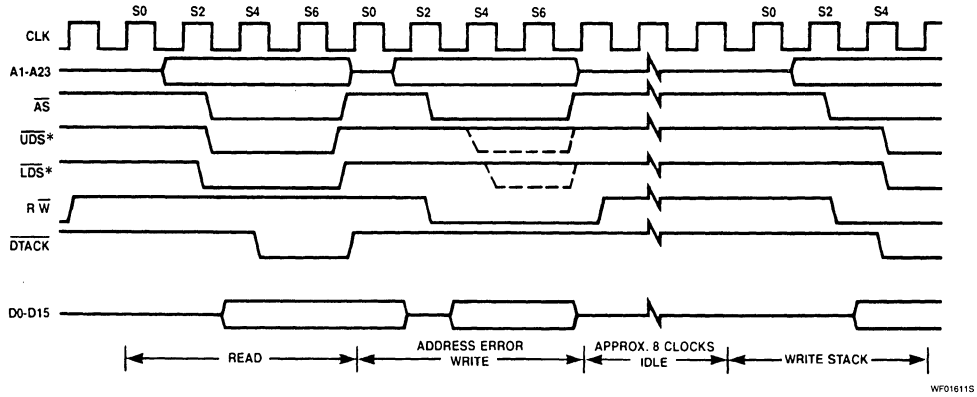
Figure 36. Exception Stack Order (Group 0)

Address Error

Address error exceptions occur when the processor attempts to access a word or a long word operand or an instruction at an odd address. The effect is much like an internally generated bus error, so that the bus cycle is aborted and the processor ceases whatever processing it is currently doing and begins exception processing. After the exception processing commences, the sequence is the same as that for bus error including the information that is stacked, except that the vector number refers to the address error vector instead. Likewise, if an address error occurs during the exception processing for a bus error, address error, or reset, the processor is halted. As shown in figure 37, an address error will execute a short bus cycle followed by exception processing.

INTERFACE WITH SYNCHRONOUS PERIPHERALS

To interface the synchronous peripherals with the asynchronous SCN68000, the processor modifies its bus cycle to meet the synchronous cycle requirements whenever a synchronous device address is detected. This is possible since both processors use memory mapped I/O. Figure 38 is a flowchart of the interface operation between the processor and synchronous devices.



*UDS and LDS are asserted on mask sets R9M, BF4, T6E, DL6, CC1, and GN7.

Figure 37. Address Error Timing Diagram

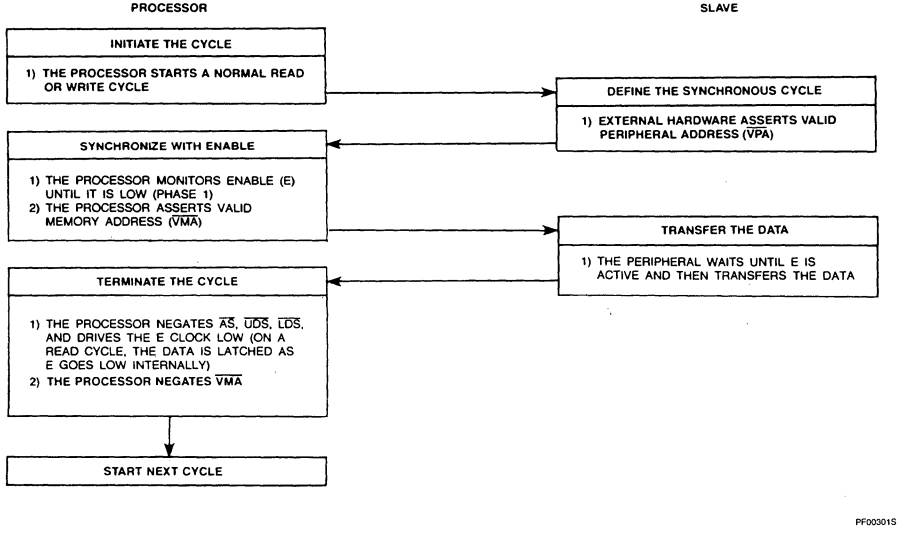


Figure 38. Synchronous Interfacing Flowchart

Data Transfer Operation

Three signals on the processor provide the synchronous interface. They are: enable (E), valid memory address ($\overline{VMA}$), and valid peripheral address ($\overline{VPA}$). Enable corresponds to the E or phase 2 signal in existing synchronous systems. The bus frequency is one tenth of the incoming SCN68000 clock frequency. The timing of E allows 1 megahertz peripherals to be used with 8 megahertz SCN68000s. Enable has a 60/40 duty cycle; that is, it is low for six input clocks and high for four input clocks. This duty cycle allows the processor to do successive $\overline{VPA}$ accesses on successive E pulses.

Synchronous cycle timing is given in figures 39, 40, 48, and 49. At state zero (S0) in the cycle, the address bus is in the high-impedance state. A function code is asserted on the function code output lines. One-half clock later, in state 1, the address bus is released from the high-impedance state.

During state 2, the address strobe ($\overline{AS}$) is asserted to indicate that there is a valid address on the address bus. If the bus cycle is a read cycle, the upper and/or lower data strobes are also asserted in state 2. If the bus cycle is a write cycle, the read/write (R/$\overline{W}$) signal is switched to low (write) during state 2. One-half clock later, in state 3, the write data is placed on the data bus, and in state 4 the data strobes are issued to indicate valid data on the data bus. The processor now inserts wait states until it recognizes the assertion of $\overline{VPA}$.

The $\overline{VPA}$ input signals the processor that the address on the bus is the address of a synchronous device (or an area reserved for synchronous devices) and that the bus should conform to the phase 2 transfer characteristics of the synchronous bus. Valid peripheral address is derived by decoding the address bus, conditioned by the address strobe. Chip select for the synchronous peripherals should be derived by decoding the address bus conditioned by $\overline{VMA}$.

After recognition of $\overline{VPA}$, the processor assures that the enable (E) is low, by waiting if necessary, and subsequently asserts $\overline{VMA}$. Valid memory address is then used as part of the chip select equation of the peripheral. This ensures that the synchronous peripherals are selected and deselected at the correct time. The peripheral now runs its cycle during the high portion of the E signal. Figures 39 and 40 depict the best and worst case synchronous cycle timing. This cycle length is dependent strictly on when $\overline{VPA}$ is asserted in relationship to the E clock.

If we assume that external circuitry asserts $\overline{VPA}$ as soon as possible after the assertion of $\overline{AS}$, then $\overline{VPA}$ will be recognized as being asserted on the falling edge of S4. In this case, no "extra" wait cycles will be inserted prior to the recognition of $\overline{VPA}$ asserted and only the wait cycles inserted to synchronize with the E clock will determine the total length of the cycle. In any case, the synchronization delay will be some integral number of clock cycles within the following two extremes:

1. Best Case — $\overline{VPA}$ is recognized as being asserted on the falling edge three clock cycles before E rises (or three clock cycles after E falls).
2. Worst Case — $\overline{VPA}$ is recognized as being asserted on the falling edge two clock cycles before E rises (or four clock cycles after E falls).

During a read cycle, the processor latches the peripheral data in state 6. For all cycles, the processor negates the address and data strobes one-half clock cycle later in state 7 and the enable signal goes low at this time. Another half clock later, the address bus is put in the high-impedance state. During a write cycle, the data bus is put in the high-impedance state and the read/write signal is switched high. The peripheral logic must remove $\overline{VPA}$ within one clock after the address strobe is negated.

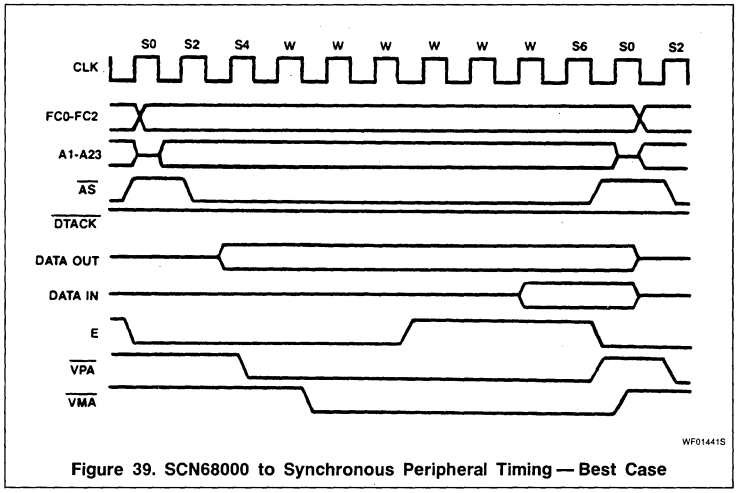


Figure 39. SCN68000 to Synchronous Peripheral Timing — Best Case

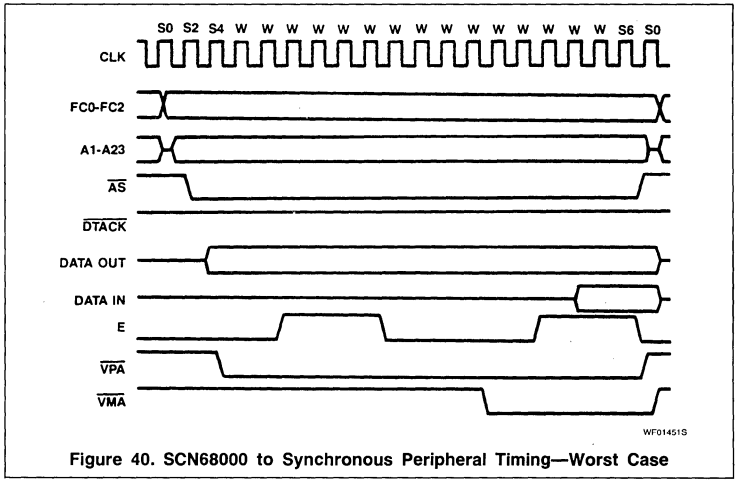


Figure 40. SCN68000 to Synchronous Peripheral Timing—Worst Case

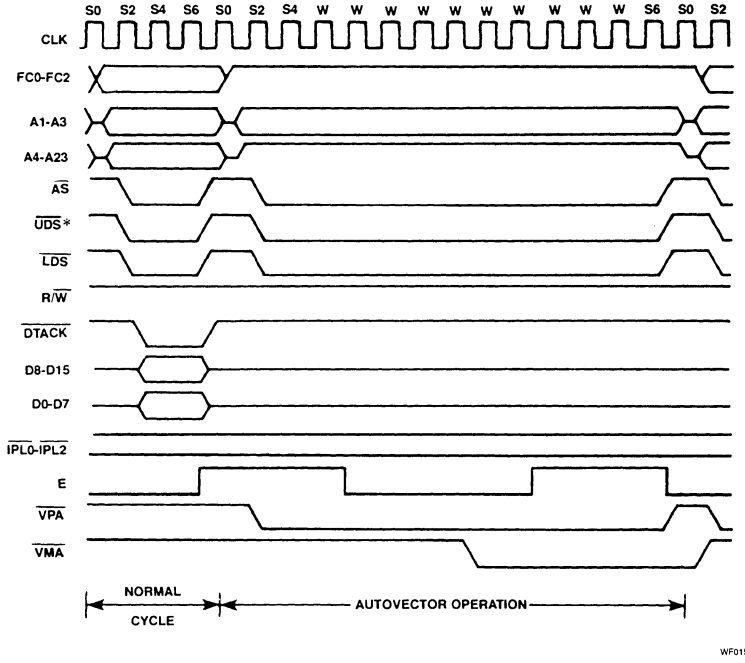
$\overline{DTACK}$ should not be asserted while $\overline{VPA}$ is asserted. Notice that the SCN68000 $\overline{VMA}$ is active low, contrasted with the active high synchronous $\overline{VMA}$. This allows the processor to put its buses in the high-impedance state on DMA requests without inadvertently selecting the peripherals.

Interrupt Interface Operation

During an interrupt acknowledge cycle while the processor is fetching the vector, the $\overline{VPA}$ is asserted, the SCN68000 will assert $\overline{VMA}$ and complete a normal synchronous read cycle as shown in figure 41. The processor will then use an internally generated vector that is a function of the interrupt being serviced. This process is known as autovectoring. The seven autovectors are vector numbers 25 through 31 (decimal).

Autovectoring operates in the same fashion (but is not restricted to) the synchronous interrupt sequence. The basic difference is that there are six normal interrupt vectors and one NMI type vector. As with both the synchronous and the SCN68000's normal vectored interrupt, the interrupt service routine can be located anywhere in the address space. This is due to the fact that while the vector numbers are fixed, the contents of the vector table entries are assigned by the user.

Since $\overline{VMA}$ is asserted during autovectoring, care should be taken to insure the synchronous peripheral address decoding prevents unintended accesses.



*Although $\overline{UDS}$ and $\overline{LDS}$ are asserted, no data is read from the bus during the autovector cycle. The vector number is generated internally.

Figure 41. Autovector Operation Timing Diagram

INSTRUCTION SET AND EXECUTION TIMES

Instruction Set

The following paragraphs provide information about the addressing categories and instruction set of the SCN68000.

Addressing Categories

Effective address modes may be categorized by the ways in which they may be used. The following classifications will be used in the instruction definitions.

These categories may be combined, so that additional, more restrictive, classifications may be defined. For example, the instruction descriptions use such classifications as alterable memory or data alterable. The former refers to those addressing modes which are both alterable and memory addresses, and the latter refers to addressing modes which are both data and alterable.

Table 21 shows the various categories to which each of the effective address modes belong. Table 22 is the instruction set summary.

Data — If an effective address mode may be used to refer to data operands, it is considered a data addressing effective address mode.

Memory — If an effective address mode may be used to refer to memory operands, it is considered a memory addressing effective address mode.

Alterable — If an effective address mode may be used to refer to alterable (writable) operands, it is considered an alterable addressing effective address mode.

Control — If an effective address mode may be used to refer to memory operands without an associated size, it is considered a control addressing effective address mode.

Table 21. EFFECTIVE ADDRESSING MODE CATEGORIES

EFFECTIVE ADDRESS MODES	MODE	REGISTER	ADDRESSING CATEGORIES			
			Data	Memory	Control	Alterable
Dn	000	Register number	X	–	–	X
An	001	Register number	–	–	–	X
(An)	010	Register number	X	X	X	X
(An)+	011	Register number	X	X	–	X
–(An)	100	Register number	X	X	–	X
d(An)	101	Register number	X	X	X	X
d(An, ix)	110	Register number	X	X	X	X
xxx.W	111	000	X	X	X	X
xxx.L	111	001	X	X	X	X
d(PC)	111	010	X	X	X	–
d(PC, ix)	111	011	X	X	X	–
#xxx	111	100	X	X	–	–

2

Table 22. INSTRUCTION SET

MNEMONIC	DESCRIPTION	OPERATION	CONDITION CODES				
			X	N	Z	V	C
ABCD	Add decimal with extend	$(Destination)_{10} + (Source)_{10} + X \rightarrow Destination$	*	U	*	U	*
ADD	Add binary	(Destination) + (Source) → Destination	*	*	*	*	*
ADDA	Add address	(Destination) + (Source) → Destination	–	–	–	–	–
ADDI	Add immediate	(Destination) + Immediate Data → Destination	*	*	*	*	*
ADDQ	Add quick	(Destination) + Immediate Data → Destination	*	*	*	*	*
ADDX	Add extended	(Destination) + (Source) + X → Destination	*	*	*	*	*
AND	AND logical	(Destination) ∧ (Source) → Destination	–	*	*	0	0
ANDI	AND immediate	(Destination) ∧ Immediate Data → Destination	–	*	*	0	0
ANDI to CCR	AND immediate to condition codes	(Source) ∧ CCR → CCR	*	*	*	*	*
ANDI to SR	AND immediate to status register	(Source) ∧ SR → SR	*	*	*	*	*
ASL, ASR	Arithmetic shift	(Destination) Shifted by < count > → Destination	*	*	*	*	*
B_{CC}	Branch conditionally	If $_{CC}$ then PC + d → PC	–	–	–	–	–
BCHG	Test a bit and change	~(< bit number >) OF Destination → Z ~(< bit number >) OF Destination → < bit number > OF Destination	–	–	*	–	–
BCLR	Test a bit and clear	~(< bit number >) OF Destination → Z 0 → < bit number > → OF Destination	–	–	*	–	–
BRA	Branch always	PC + d → PC	–	–	–	–	–
BSET	Test a bit and set	~(< bit number >) OF Destination → Z 1 → < bit number > OF Destination	–	–	*	–	–
BSR	Branch to subroutine	PC → –(SP); PC + d → PC	–	–	–	–	–
BTST	Test a bit	~(< bit number >) OF Destination → Z	–	–	*	–	–
CHK	Check register against bounds	If Dn < 0 or Dn > (< ea >) then TRAP	–	*	U	U	U
CLR	Clear an operand	0 → Destination	–	0	1	0	0
CMP	Compare	(Destination) – (Source)	–	*	*	*	*
CMPA	Compare address	(Destination) – (Source)	–	*	*	*	*
CMPI	Compare immediate	(Destination) – Immediate Data	–	*	*	*	*
CMPM	Compare memory	(Destination) – (Source)	–	*	*	*	*
DB_{CC}	Test condition, decrement and branch	if ~$_{CC}$ then Dn – 1 → Dn; if Dn ≠ –1 then PC + d → PC	–	–	–	–	–
DIVS	Signed divide	(Destination)/(Source) → Destination	–	*	*	*	0
DIVU	Unsigned divide	(Destination)/(Source) → Destination	–	*	*	*	0
EOR	Exclusive OR logical	(Destination) ⊕ (Source) → Destination	–	*	*	0	0
EORI	Exclusive OR immediate	(Destination) ⊕ Immediate Data → Destination	–	*	*	0	0
EORI to CCR	Exclusive OR immediate to condition codes	(Source) ⊕ CCR → CCR	*	*	*	*	*
EORI to SR	Exclusive OR immediate to status register	(Source) ⊕ SR → SR	*	*	*	*	*
EXG	Exchange register	Rx ↔ Ry	–	–	–	–	–
EXT	Sign extend	(Destination) Sign-extended → Destination	–	*	*	0	0
JMP	Jump	Destination → PC	–	–	–	–	–
JSR	Jump to subroutine	PC → –(SP); Destination → PC	–	–	–	–	–
LEA	Load effective address	Destination → An	–	–	–	–	–
LINK	Link and allocate	An → –(SP); SP → An; SP + Displacement → SP	–	–	–	–	–
LSL, LSR	Logical shift	(Destination) Shifted by < count > → Destination	*	*	*	0	*
MOVE	Move data from source to destination	(Source) → Destination	–	*	*	0	0
MOVE to CCR	Move to condition code	(Source) → CCR	*	*	*	*	*
MOVE to SR	Move to status register	(Source) → SR	*	*	*	*	*
MOVE from SR	Move from status register	SR → Destination	–	–	–	–	–

Table 22. INSTRUCTION SET (Continued)

MNEMONIC	DESCRIPTION	OPERATION	CONDITION CODES				
			X	N	Z	V	C
MOVE USP	Move user stack pointer	USP → An; An → USP	–	–	–	–	–
MOVEA	Move address	(Source) → Destination	–	–	–	–	–
MOVEM	Move multiple registers	Registers → Destination (Source) → Registers	–	–	–	–	–
MOVEP	Move peripheral data	(Source) → Destination	–	–	–	–	–
MOVEQ	Move quick	Immediate Data → Destination	–	*	*	0	0
MULS	Signed multiply	(Destination) X (Source) → Destination	–	*	*	0	0
MULU	Unsigned multiply	(Destination) X (Source) → Destination	–	*	*	0	0
NBCD	Negate decimal with extend	$0 - (\text{Destination})_{10} - X \rightarrow$ Destination	*	U	*	U	*
NEG	Negate	0 – (Destination) → Destination	*	*	*	*	*
NEGX	Negate with extend	0 – (Destination) – X → Destination	*	*	*	*	*
NOP	No operation	–	–	–	–	–	–
NOT	Logical complement	~ (Destination) → Destination	–	*	*	0	0
OR	Inclusive OR logical	(Destination) v (Source) → Destination	–	*	*	0	0
ORI	Inclusive OR immediate	(Destination) v Immediate Data → Destination	–	*	*	0	0
ORI to CCR	Inclusive OR immediate to conditions codes	(Source) v CCR → CCR	*	*	*	*	*
ORI to SR	Inclusive OR immediate to status register	(Source) v SR → SR	*	*	*	*	*
PEA	Push effective address	Destination → – (SP)	–	–	–	–	–
RESET	Reset external devices	–	–	–	–	–	–
ROL, ROR	Rotate (without extend)	(Destination) Rotated by < count > → Destination	–	*	*	0	*
ROXL, ROXR	Rotate with extend	(Destination) Rotated by < count > → Destination	*	*	*	0	*
RTE	Return from exception	(SP) + → SR; (SP) + → PC	*	*	*	*	*
RTR	Return and restore condition codes	(SP) + → CC; (SP) + → PC	*	*	*	*	*
RTS	Return from subroutine	(SP) + → PC	–	–	–	–	–
SBCD	Subtract decimal with extend	$(\text{Destination})_{10} - (\text{Source})_{10} - X \rightarrow$ Destination	*	U	*	U	*
S_{CC}	Set according to condition	If CC then 1's → Destination else 0's → Destination	–	–	–	–	–
STOP	Load status register and stop	Immediate Data → SR; STOP	*	*	*	*	*
SUB	Subtract binary	(Destination) – (Source) → Destination	*	*	*	*	*
SUBA	Subtract address	(Destination) – (Source) → Destination	–	–	–	–	–
SUBI	Subtract immediate	(Destination) – Immediate Data → Destination	*	*	*	*	*
SUBQ	Subtract quick	(Destination) – Immediate Data → Destination)	*	*	*	*	*
SUBX	Subtract with extend	(Destination) – (Source) – X → Destination	*	*	*	*	*
SWAP	Swap register halves	Register [31:16] ↔ Register [15:0]	–	*	*	0	0
TAS	Test and set an operand	(Destination) Tested → CC; 1 → [7] OF Destination	–	*	*	0	0
TRAP	Trap	PC → – (SSP); SR → – (SSP); (Vector) → PC	–	–	–	–	–
TRAPV	Trap on overflow	If V then TRAP	–	–	–	–	–
TST	Test and operand	(Destination) Tested → CC	–	*	*	0	0
UNLK	Unlink	An → SP; (SP) + → An	–	–	–	–	–

NOTES:

[ ] = bit number
⊕ logical exclusive OR
∧ logical AND
∨ logical OR
~ logical complement

* affected
– unaffected
0 cleared
1 set
U undefined

Instruction Prefetch

The SCN68000 uses a two-word tightly-coupled instruction prefetch mechanism to enhance performance. This mechanism is described in terms of the microcode operations involved. If the execution of an instruction is defined to begin when the microroutine for that instruction is entered, some features of the prefetch mechanism can be described.

1. When execution of an instruction begins, the operation word and the word following have already been fetched. The operation word is in the instruction decoder.
2. In the case of multi-word instructions, as each additional word of the instruction is used internally, a fetch is made to the instruction stream to replace it.
3. The last fetch for an instruction from the instruction stream is made when the operation word is discarded and decoding is started on the next instruction.
4. If the instruction is a single-word instruction causing a branch, the second word is not used. But because this word is fetched by the preceding instruction, it is impossible to avoid this superfluous fetch.
5. In the case of an interrupt or trace exception, both words are not used.
6. The program counter usually points to the last word fetched from the instruction stream.

Instruction Execution Times

The following paragraphs contain listings of the instruction execution times in terms of external clock (CLK) periods. In this timing data, it is assumed that both memory read and write cycle times are four clock periods. Any wait states caused by a longer memory cycle must be added to the total instruction time. The number of bus read and write cycles for each instruction is also included with the timing data. This timing data is enclosed in parentheses following the execution periods and is shown as (r/w) where r is the number of read cycles and w is the number of write cycles.

NOTE

The number of periods includes instruction fetch and all applicable operand fetches and stores.

Effective Address Operand Calculation Timing

Table 23 lists the number of clock periods required to compute an instruction's effective address. It includes fetching of any extension words, the address computation, and fetching of the memory operand. The number of bus read and write cycles is shown in parentheses as (r/w). Note there are no write cycles involved in processing the effective address.

Table 23. EFFECTIVE ADDRESS CALCULATION TIMES

ADDRESSING MODE		BYTE,WORD	LONG
Register			
Dn	Data register direct	0(0/0)	0(0/0)
An	Address register direct	0(0/0)	0(0/0)
Memory			
(An)	Address register indirect	4(1/0)	8(2/0)
(An)+	Address register indirect with postincrement	4(1/0)	8(2/0)
-(An)	Address register indirect with predecrement	6(1/0)	10(2/0)
d(An)	Address register indirect with displacement	8(2/0)	12(3/0)
d(An,ix)*	Address register indirect with index	10(2/0)	14(3/0)
xxx.W	Absolute short	8(2/0)	12(3/0)
xxx.L	Absolute long	12(3/0)	16(4/0)
d(PC)	Program counter with displacement	8(2/0)	12(3/0)
d(PC,ix)*	Program counter with index	10(2/0)	14(3/0)
#xxx	Immediate	4(1/0)	8(2/0)

*The size of the index register (ix) does not affect execution time.

Move Instruction Execution Times

Tables 24 and 25 indicate the number of clock periods for the move instruction. This data includes instruction fetch, operand reads, and operand writes. The number of bus read and write cycles is shown in parentheses as (r/w).

Table 24. MOVE BYTE INSTRUCTION EXECUTION TIMES

SOURCE	DESTINATION								
	Dn	An	(An)	(An)+	-(An)	d(An)	d(An,ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
(An)	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
(An)+	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
-(An)	10(2/0)	10(2/0)	14(2/1)	14(2/1)	14(2/1)	18(3/1)	20(3/1)	18(3/1)	22(4/1)
d(An)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
d(An,ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
xxx.W	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
xxx.L	16(4/0)	16(4/0)	20(4/1)	20(4/1)	20(4/1)	24(5/1)	26(5/1)	24(5/1)	28(6/1)
d(PC)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
d(PC,ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
#xxx	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)

*The size of the index register (ix) does not affect execution time.

Table 25. MOVE LONG INSTRUCTION EXECUTION TIMES

SOURCE	DESTINATION								
	Dn	An	(An)	(An)+	-(An)	d(An)	d(An,ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	12(1/2)	12(1/2)	12(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An	4(1/0)	4(1/0)	12(1/2)	12(1/2)	12(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
(An)	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
(An)+	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
-(An)	14(3/0)	14(3/0)	22(3/2)	22(3/2)	22(3/2)	26(4/2)	28(4/2)	26(4/2)	30(5/2)
d(An)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
d(An,ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
xxx.W	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
xxx.L	20(5/0)	20(5/0)	28(5/2)	28(5/2)	28(5/2)	32(6/2)	34(6/2)	32(6/2)	36(7/2)
d(PC)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(5/2)
d(PC,ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
#xxx	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)

*The size of the index register (ix) does not affect execution time.

Standard Instruction Execution Times

The number of clock periods shown in table 26 indicates the time required to perform the operations, store the results, and read the next instruction. The number of bus read and write cycles is shown in parentheses as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In table 26 the headings have the following meanings: An = address register operand, DN=data register operand, ea=an operand specified by an effective address, and M = memory effective address operand.

Table 26. STANDARD INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	op<ea>, An†	op<ea>, Dn	op Dn, <M>
ADD	byte, word	8(1/0)+	4(1/0)+	8(1/1)+
	long	6(1/0)+**	6(1/0)+**	12(1/2)+
AND	byte, word	-	4(1/0)+	8(1/1)+
	long	-	6(1/0+**	12(1/2)+
CMP	byte, word	6(1/0)+	4(1/0)+	-
	long	6(1/0)+	6(1/0)+	-
DIVS	-	-	158(1/0)+*	-
DIVU	-	-	140(1/0)+*	-
EOR	byte, word	-	4(1/0)***	8(1/1)+
	long	-	8(1/0)***	12(1/2)+
MULS	-	-	70(1/0)+*	-
MULU	-	-	70(1/0)+*	-
OR	byte, word	-	4(1/0)+	8(1/1)+
	long	-	6(1/0)+**	12(1/2)+
SUB	byte, word	8(1/0)+	4(1/0)+	8(1/1)+
	long	6(1/0)+**	6(1/0)+**	12(1/2)+

NOTES:

+ Add effective address calculation time.

† Word or long word only.

* Indicates maximum value.

** The base time of 6 clock periods is increased to 8 if the effective address mode is register direct or immediate (effective address time should also be added).

*** Only available effective address mode is data register direct.

DIVS, DIVU – The divide algorithm used by the SCN68000 provides less than 10% difference between the best and worst case timings.

MULS, MULU – The multiply algorithm requires 38 + 2n clocks where n is defined as:

MULU: n = the number of ones in the < ea > .

MULS: n = concatenate the < ea > with a zero as the LSB; n is the resultant number of 10 or 01 patterns in the 17-bit source; i.e., worst case happens when the source is $5555.

Immediate Instruction Execution Times

The number of clock periods shown in table 27 includes the time to fetch immediate operands, perform the operations, store the results, and read the next operation. The number of bus read and write cycles is shown in parentheses as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In table 27, the headings have the following meanings: # = immediate operand, DN = data register operand, An = address register operand, M = memory operand, and SR = status register.

Table 27. IMMEDIATE INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	op#, Dn	op#, An	op#, M
ADDI	byte, word	8(2/0)	–	12(2/1)+
	long	16(3/0)	–	20(3/2)+
ADDQ	byte, word	4(1/0)	8(1/0)*	8(1/1)+
	long	8(1/0)	8(1/0)	12(1/2)+
ANDI	byte, word	8(2/0)	–	12(2/1)+
	long	16(3/0)	–	20(3/1)+
CMPI	byte, word	8(2/0)	–	8(2/0)+
	long	14(3/0)	–	12(3/0)+
EORI	byte, word	8(2/0)	–	12(2/1)+
	long	16(3/0)	–	20(3/2)+
MOVEQ	long	4(1/0)	–	–
ORI	byte, word	8(2/0)	–	12(2/1)+
	long	16(3/0)	–	20(3/2)+
SUBI	byte, word	8(2/0)	–	12(2/1)+
	long	16(3/0)	–	20(3/2)+
SUBQ	byte, word	4(1/0)	8(1/0)*	8(1/1)+
	long	8(1/0)	8(1/0)	12(1/2)+

+ add effective address calculation time
* word only

Single Operand Instruction Execution Times

Table 28 indicates the number of clock periods for the single operand instructions. The number of bus read and write cycles is shown in parentheses as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 28. SINGLE OPERAND INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	REGISTER	MEMORY
CLR	byte, word	4(1/0)	8(1/1)+
	long	6(1/0)	12(1/2)+
NBCD	byte	6(1/0)	8(1/1)+
NEG	byte, word	4(1/0)	8(1/1)+
	long	6(1/0)	12(1/2)+
NEGX	byte, word	4(1/0)	8(1/1)+
	long	6(1/0)	12(1/2)+
NOT	byte, word	4(1/0)	8(1/1)+
	long	6(1/0)	12(1/2)+
S_{CC}	byte, false	4(1/0)	8(1/1)+
	byte, true	6(1/0)	8(1/1)+
TAS	byte	4(1/0)	10(1/1)
TST	byte, word	4(1/0)	4(1/0)+
	long	4(1/0)	4(1/0)+

+ add effective address calculation time

Shift/Rotate Instruction Execution Times

Table 29 indicates the number of clock periods for the shift and rotate instructions. The number of bus read and write cycles is shown in parentheses as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 29. SHIFT/ROTATE INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	REGISTER	MEMORY
ASR, ASL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–
LSR, LSL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–
ROR, ROL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–
ROXR, ROXL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–

+ add effective address calculation time
n is the shift or rotate count

Bit Manipulation Instruction Execution Times

Table 30 indicates the number of clock periods required for the bit manipulation instructions. The number of bus read and write cycles is shown in parentheses as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 30. BIT MANIPULATION INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	DYNAMIC		STATIC	
		Register	Memory	Register	Memory
BCHG	byte	–	8(1/1)+	–	12(2/1)+
	long	8(1/0)*	–	12(2/0)*	–
BCLR	byte	–	8(1/1)+	–	12(2/1)+
	long	10(1/0)*	–	14(2/0)*	–
BSET	byte	–	8(1/1)+	–	12(2/1)+
	long	8(1/0)*	–	12(2/0)*	–
BTST	byte	–	4(1/0)+	–	8(2/0)+
	long	6(1/0)	–	10(2/0)	–

+ add effective calculation time
* indicates maximum value

Conditional Instruction Execution Times

Table 31 indicates the number of clock periods required for the conditional instructions. The number of bus read and write cycles is indicated in parentheses as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 31. CONDITIONAL INSTRUCTION EXECUTION TIMES

INSTRUCTION	DISPLACEMENT	BRANCH TAKEN	BRANCH NOT TAKEN
B_{CC}	byte	10(2/0)	8(1/0)
	word	10(2/0)	12(2/0)
BRA	byte	10(2/0)	–
	word	10(2/0)	–
BSR	byte	18(2/2)	–
	word	18(2/2)	–
DB_{CC}	CC true	–	12(2/0)
	CC false	10(2/0)	14(3/0)

Table 32. JMP, JSR, LEA, PEA, and MOVEM INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	(An)	(An)+	-(An)	d(An)	d(An, ix)+	xxx.W	xxx.L	d(PC)	d(PC,ix)*
JMP	–	8(2/0)	–	–	10(2/0)	14(3/0)	10(2/0)	12(3/0)	10(2/0)	14(3/0)
JSR	–	16(2/2)	–	–	18(2/2)	22(2/2)	18(2/2)	20(3/2)	18(2/2)	22(2/2)
LEA	–	4(1/0)	–	–	8(2/0)	12(2/0)	8(2/0)	12(3/0)	8(2/0)	12(2/0)
PEA	–	12(1/2)	–	–	16(2/2)	20(2/2)	16(2/2)	20(3/2)	16(2/2)	20(2/2)
MOVEM M → R	word	12 + 4n (3 + n/0)	12 + 4n (3 + n/0)	–	16 + 4n (4 + n/0)	18 + 4n (4 + n/0)	16 + 4n (4 + n/0)	20 + 4n (5 + n/0)	16 + 4n (4 + n/0)	18 + 4n (4 + n/0)
	long	12 + 8n (3 + 2n/0)	12 + 8n (3 + 2n/0)	–	16 + 8n (4 + 2n/0)	18 + 8n (4 + 2n/0)	16 + 8n (4 + 2n/0)	20 + 8n (5 + 2n/0)	16 + 8n (4 + 2n/0)	18 + 8n (4 + 2n/0)
MOVEM R → M	word	8 + 4n (2/n)	–	8 + 4n (2/n)	12 + 4n (3/n)	14 + 4n (3/n)	12 + 4n (3/n)	16 + 4n (4/n)	– –	– –
	long	8 + 8n (2/2n)	– –	8 + 8n (2/2n)	12 + 8n (3/2n)	14 + 8n (3/2n)	12 + 8n (3/2n)	16 + 8n (4/2n)	– –	– –

n is the number of registers to move
* is the size of the index register (ix) and does not affect the instruction's execution time

JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times

Table 32 indicates the number of clock periods required for the jump, jump-to-subroutine, load effective address, push effective address, and move multiple registers instructions. The number of bus read and write cycles is shown in parentheses as (r/w).

Multi-Precision Instruction Execution Times

Table 33 indicates the number of clock periods for the multi-precision instructions. The number of clock periods includes the time to fetch both operands, perform the operations, store the results, and read the next instructions. The number of read and write cycles is shown in parentheses as (r/w).

In table 33, the headings have the following meanings: Dn = data register operand and M = memory operand.

Table 33. MULTI-PRECISION INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	op Dn, Dn	op M, M
ADDX	byte, word	4(1/0)	18(3/1)
	long	8(1/0)	30(5/2)
CMPM	byte, word	–	12(3/0)
	long	–	20(5/0)
SUBX	byte, word	4(1/0)	18(3/1)
	long	8(1/0)	30(5/2)
ABCD	byte	6(1/0)	18(3/1)
SBCD	byte	6(1/0)	18(3/1)

Table 34. MISCELLANEOUS INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	REGISTER	MEMORY
ANDI to CCR	byte	20(3/0)	–
ANDI to SR	word	20(3/0)	–
CHK	–	10(1/0)+	–
EORI to CCR	byte	20(3/0)	–
EORI to SR	word	20(3/0)	–
ORI to CCR	byte	20(3/0)	–
ORI to SR	word	20(3/0)	–
MOVE from SR	–	6(1/0)	8(1/1)+
MOVE to CCR	–	12(2/0)	12(2/0)+
MOVE to SR	–	12(2/0)	12(2/0)+
EXG	–	6(1/0)	–
EXT	word	4(1/0)	–
	long	4(1/0)	–
LINK	–	16(2/2)	–
MOVE from USP	–	4(1/0)	–
MOVE to USP	–	4(1/0)	–
NOP	–	4(1/0)	–
RESET	–	132(1/0)	–
RTE	–	20(5/0)	–
RTR	–	20(5/0)	–
RTS	–	16(4/0)	–
STOP	–	4(0/0)	–
SWAP	–	4(1/0)	–
TRAPV	–	4(1/0)	–
UNLK	–	12(3/0)	–

+ add effective address calculation time

Table 35. MOVE PERIPHERAL INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	REGISTER → MEMORY	MEMORY → REGISTER
MOVEP	word	16(2/2)	16(4/0)
	long	24(2/4)	24(6/0)

Table 36. EXCEPTION PROCESSING EXECUTION TIMES

EXCEPTION	PERIODS
Address error	50(4/7)
Bus error	50(4/7)
CHK instruction	44(5/4)+
Divide by zero	42(5/4)
Illegal instruction	34(4/3)
Interrupt	44(5/3)*
Privilege violation	34(4/3)
$\overline{\text{RESET}}$**	40(6/0)
Trace	34(4/3)
TRAP instruction	38(4/4)
TRAPV instruction	34(4/3)

+ add effective address calculation time.
*The interrupt acknowledge cycle is assumed to take four clock periods.
**Indicates the time from when $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ are first sampled as negated to when instruction execution starts.

Miscellaneous Instruction Execution Times

Tables 34 and 35 indicate the number of clock periods for the following miscellaneous instructions. The number of bus read and write cycles is shown in parentheses as (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

Exception Processing Execution Times

Table 36 indicates the number of clock periods for exception processing. The number of clock periods includes the time for all stacking, the vector fetch, and the fetch of the first two instruction words of the handler routine. The number of bus read and write cycles is shown in parentheses as (r/w).

ABSOLUTE MAXIMUM RATINGS

RATING	SYMBOL	VALUE	UNIT
Supply voltage	V_{CC}	−0.3 to +7.0	V
Input voltage	V_{in}	−0.3 to +7.0	V
Operating temperature range		T_L to T_H	
SCN68000	T_A	0 to 70	°C
SCN68000 ceramic		−40 to 85	
Storage temperature	T_{stg}	−55 to 150	°C

NOTE:
This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{CC}).

THERMAL CHARACTERISTICS

CHARACTERISTIC	SYMBOL	VALUE	SYMBOL	VALUE	RATING
Thermal Resistance (Still Air)					
Ceramic	θ_{JA}	30	θ_{JC}	15*	°C/W
Pin grid array		33		15	
Plastic		30		15*	

*Estimated

2

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0Vdc ±5%; GND = 0Vdc; $T_A = T_L$ to T_H (see figures 42, 43, and 44)

CHARACTERISTIC		SYMBOL	MIN	MAX	UNIT
Input high voltage		V_{IH}	2.0	V_{CC}	V
Input low voltage		V_{IL}	GND – 0.3	0.8	V
Input leakage current @ 5.25V	$\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$, CLK, $\overline{IPL0}$-$\overline{IPL2}$, $\overline{VPA}$ $\overline{HALT}$, $\overline{RESET}$	I_{in}	– –	2.5 20	μA
Three-state (off state) input current @ 2.4V/0.4V	$\overline{AS}$, A1-A23, D0-D15, FC0-FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	I_{TSI}	–	20	μA
Output high voltage (I_{OH} = –400μA)	E* E, $\overline{AS}$, A1-A23, $\overline{BG}$, D0-D15, FC0-FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	V_{OH}	V_{CC} – 0.75 2.4	– –	V
Output low voltage (I_{OL} = 1.6mA) (I_{OL} = 3.2mA) (I_{OL} = 5.0mA) (I_{OL} = 5.3mA)	$\overline{HALT}$ A1-A23, $\overline{BG}$, FC0-FC2 $\overline{RESET}$ E, $\overline{AS}$, D0-D15, $\overline{LDS}$, R/$\overline{W}$ $\overline{UDS}$, $\overline{VMA}$	V_{OL}	– – – –	0.5 0.5 0.5 0.5	V
Power dissipation (See Power Considerations)		P_D***	–	–	W
Capacitance (V_{in} = 0V, T_A = 25°C; frequency = 1MHz**)		C_{in}	–	20.0	pF

* With external pullup resistor of 1.1kΩ.
** Capacitance is periodically sampled rather than 100% tested.
*** During normal operation instantaneous V_{CC} current requirements may be as high as 1.5A.

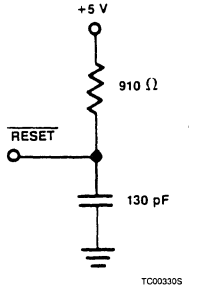


Figure 42. $\overline{RESET}$ Test Load

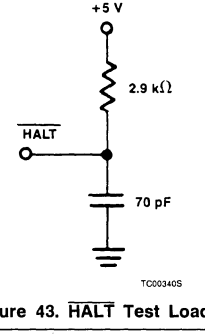


Figure 43. $\overline{HALT}$ Test Load

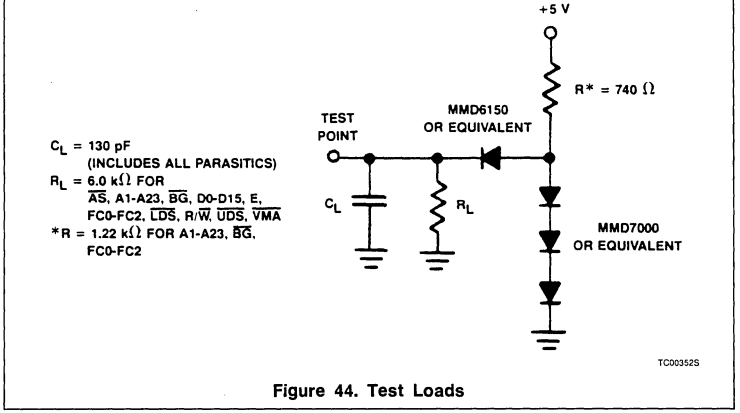


Figure 44. Test Loads

Power Considerations

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \theta_{JA}) \quad (1)$$

Where:

T_A = ambient temperature, °C

θ_{JA} = package thermal resistance, junction-to-ambient, °C/W

$P_D = P_{INT} + P_{I/O}$

$P_{INT} = I_{CC} \times V_{CC}$, watts – chip internal power

$P_{I/O}$ = power dissipation on input and output pins — user determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = T_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D 2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

Figure 45 illustrates the graphic solution to the equations, given above, for the specification power dissipations of 1.50 and 1.75 watts over the ambient temperature range of –55°C to 125°C using an average θ_{JA} of 40°C/watt to represent various SCN68000 packages.

However, actual θ_{JA}'s in the range of 30°C to 50°C/watt only change the curves slightly.

The total thermal resistance of a package (θ_{JA}) can be separated into two components, θ_{JC} and θ_{CA}, representing the barrier to heat flow from the semiconductor junction to the package (case) surface (θ_{JC}) and from the case to the outside ambient (θ_{CA}). These terms are related by the equation:

$$\theta_{JA} = \theta_{JC} + \theta_{CA} \quad (4)$$

θ_{JC} is device related and cannot be influenced by the user. However, θ_{CA} is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling and thermal convention. Thus, good thermal management on the part of the user can significantly reduce θ_{CA} so that θ_{JA} approximately equals θ_{JC}. Substitution of θ_{JC} for θ_{JA} in equation (1) will result in a lower semiconductor junction temperature.

Values for thermal resistance presented in this data sheet are provided for design purposes only. Thermal measurements are complex and dependent on procedure and set-up. User derived values for thermal resistance may differ.

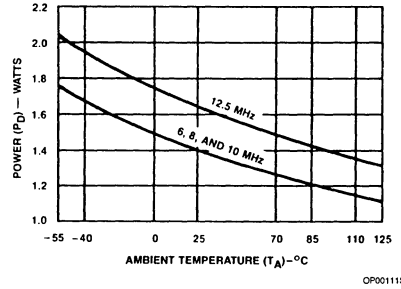


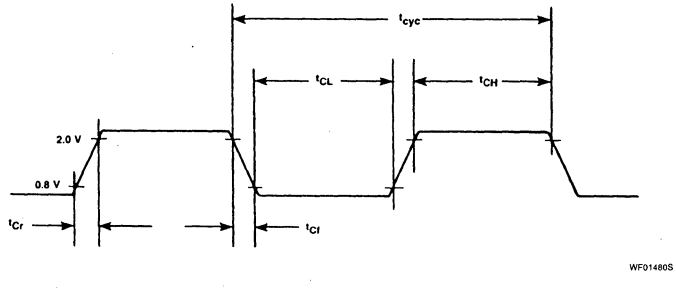
Figure 45. SCN68000 Power Dissipation (P_D) vs Ambient Temperature (T_A)

AC ELECTRICAL CHARACTERISTICS — Clock Timing (see figure 46)

CHARACTERISTIC	SYMBOL	6MHz		8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	Min	Max	
Frequency of operation	f	2.0	6.0	4.0	8.0	4.0	10.0	4.0	12.5	MHz
Cycle time	t_{cyc}	167	600	125	250	100	250	80	250	ns
Clock pulse width	t_{CL} t_{CH}	75 75	250 250	55 55	125 125	45 45	125 125	35 35	125 125	ns
Rise and fall times	t_{Cr} t_{Cf}	– –	10 10	– –	10 10	– –	10 10	– –	5 5	ns

Table 37. MAXIMUM POWER DISSIPATION BY PACKAGE TYPE MODES

PACKAGE TYPE	TEMPERATURE (°C)	MAXIMUM POWER DISSIPATION (WATTS) PER FREQUENCY (MHz)		
		6/8MHz	10MHz	12.5MHz
Ceramic	0 to 70 –40 to 85	1.50 1.65	1.50 1.65	1.75 –
Plastic	0 to 70 –40 to 85	1.50 1.65	1.50 1.65	– –
Pin grid array	0 to 70 –40 to 85	1.50 1.65	1.50 1.65	1.75 –
Plastic LCC	0 to 70	1.50	1.50	–



NOTE:
Timing measurements are referenced to and from a low voltage of 0.8V and a high voltage of 2.0V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8V and 2.0V.

Figure 46. Clock Input Timing Diagram

AC ELECTRICAL CHARACTERISTICS — Read and Write Cycles V_{CC} = 5.0Vdc ±5%; GND = 0 Vdc; $T_A = T_L$ to T_H
(see figures 47 and 48)

NO.	CHARACTERISTIC	6MHz		8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	Min	Max	
1	Clock period	167	500	125	250	100	250	80	250	ns
2	Clock width low	75	250	55	125	45	125	35	125	ns
3	Clock width high	75	250	55	125	45	125	35	125	ns
4	Clock fall time	–	10	–	10	–	10	–	5	ns
5	Clock rise time	–	10	–	10	–	10	–	5	ns
6	Clock low to address valid	–	80	–	70	–	60	–	55	ns
6A	Clock high to FC valid	–	80	–	70	–	60	–	55	ns
7	Clock high to address data high impedance (maximum)	–	100	–	80	–	70	–	60	ns
8	Clock high to address/FC invalid (minimum)	0	–	0	–	0	–	0	–	ns
9[1]	Clock high to $\overline{AS}$, $\overline{DS}$ low	–	70	0	60	0	55	0	55	ns
11[2]	Address valid to $\overline{AS}$, $\overline{DS}$ low (read)/$\overline{AS}$ low write	35	–	30	–	20	–	0	–	ns
11A[2,7]	FC valid to $\overline{AS}$, $\overline{DS}$ low (read)/$\overline{AS}$ low (write)	70	–	60	–	50	–	40	–	ns
12[1]	Clock low to $\overline{AS}$, $\overline{DS}$ high	–	80	–	70	–	55	–	50	ns
13[2]	$\overline{AS}$, $\overline{DS}$ high to address/FC invalid	40	–	30	–	20	–	10	–	ns
14[2,5]	$\overline{AS}$, $\overline{DS}$ width low (read)/$\overline{AS}$ low (write)	337	–	240	–	195	–	160	–	ns
14A[2]	$\overline{DS}$ width low (write)	170	–	115	–	95	–	80	–	ns
15[2]	$\overline{AS}$, $\overline{DS}$ width high	180	–	150	–	105	–	65	–	ns
16	Clock high to control bus high impedance	–	100	–	80	–	70	–	60	ns
17[2]	$\overline{AS}$, $\overline{DS}$ high to R/$\overline{W}$ high (read)	50	–	40	–	20	–	10	–	ns
18[1]	Clock high to R/$\overline{W}$ high	0	80	0	70	0	60	0	60	ns
20[1]	Clock high to R/$\overline{W}$ low (write)	–	80	–	70	–	60	–	60	ns
20A[8]	$\overline{AS}$ low to R/$\overline{W}$ valid	–	20	–	20	–	20	–	20	ns
21[2]	Address valid to R/$\overline{W}$ low (write)	25	–	20	–	0	–	0	–	ns
21A[2,7]	FC valid to R/$\overline{W}$ low (write)	70	–	60	–	50	–	30	–	ns
22[2]	R/$\overline{W}$ low to $\overline{DS}$ low (write)	140	–	80	–	50	–	30	–	ns
23	Clock low to data out valid (write)	–	80	–	70	–	55	–	55	ns
25[2]	$\overline{AS}$, $\overline{DS}$ high to data out invalid (write)	40	–	30	–	20	–	15	–	ns
26[2]	Data out valid to $\overline{DS}$ low (write)	35	–	30	–	20	–	15	–	ns
27[6]	Data in to clock low (set-up time on read)	25	–	15	–	10	–	10	–	ns
28[2,5]	$\overline{AS}$, $\overline{DS}$ high to $\overline{DTACK}$ high	0	325	0	245	0	190	0	150	ns
29	$\overline{AS}$, $\overline{DS}$ high to data in invalid (hold time on read)	0	–	0	–	0	–	0	–	ns
30	$\overline{AS}$, $\overline{DS}$ high to $\overline{BERR}$ high	0	–	0	–	0	–	0	–	ns
31[2,6]	$\overline{DTACK}$ low to data in (set-up time)	–	120	–	90	–	65	–	50	ns
32	$\overline{HALT}$ and $\overline{RESET}$ input transition time	0	200	0	200	0	200	0	200	ns
33	Clock high to $\overline{BG}$ low	–	80	–	70	–	60	–	50	ns
34	Clock high to $\overline{BG}$ high	–	80	–	70	–	60	–	50	ns
35	$\overline{BR}$ low to $\overline{BG}$ low	1.5	3.5	1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	Clk. per.

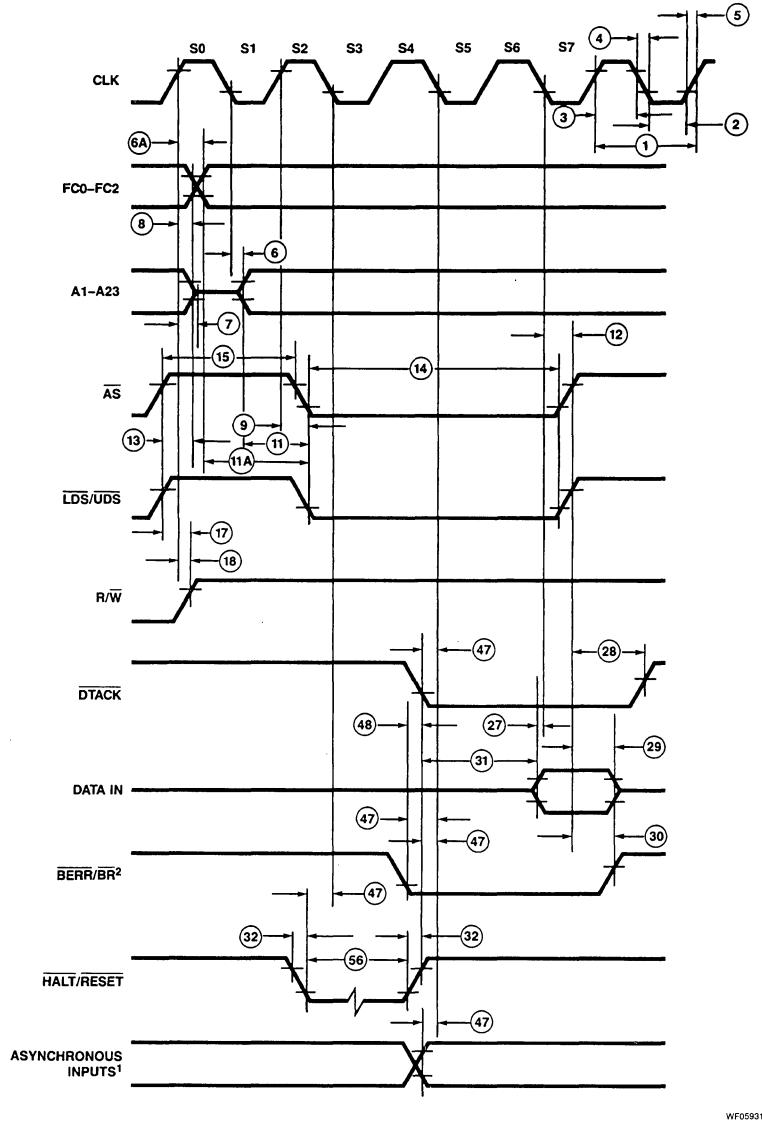
AC ELECTRICAL CHARACTERISTICS — Read and Write (Continued)

NO.	CHARACTERISTIC	6MHz		8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	Min	Max	
36[9]	$\overline{BR}$ high to $\overline{BG}$ high	1.5	3.5	1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	Clk. per.
37	$\overline{BGACK}$ low to $\overline{BG}$ low	1.5	3.0	1.5	90ns +3.0	1.5	80ns +3.0	1.5	70ns +3.0	Clk. per.
37A[10]	$\overline{BGACK}$ low to $\overline{BR}$ high	25	–	20	1.5 clocks	20	1.5 clocks	20	1.5 clocks	ns
38	$\overline{BG}$ low to control, address, data bus high impedance ($\overline{AS}$ high)	120	–	100	–	80	–	70	–	60
39	$\overline{BG}$ width high	1.5	–	1.5	–	1.5	–	1.5	–	Clk. per.
40	Clock low to $\overline{VMA}$ low	–	80	–	70	–	70	–	70	ns
41	Clock low to E transition	–	85	–	70	–	55	–	45	ns
42	E output rise and fall time	–	25	–	25	–	25	–	25	ns
43	$\overline{VMA}$ low to E high	240	–	200	–	150	–	90	–	ns
44	$\overline{AS}$, $\overline{DS}$ high to $\overline{VPA}$ high	0	160	0	120	0	90	0	70	ns
45	E low to control, address bus invalid (address hold time)	35	–	30	–	10	–	10	–	ns
46	$\overline{BGACK}$ width low	1.5	–	1.5	–	1.5	–	1.5	–	Clk. per.
47[6]	Asynchronous input set-up time	25	–	20	–	20	–	20	–	ns
48[3]	$\overline{BERR}$ low to $\overline{DTACK}$ low	25	–	20	–	20	–	20	–	ns
49[11]	$\overline{AS}$, $\overline{DS}$ high to E low	–80	–	–70	70	–55	55	–45	45	ns
50	E width high	600	–	450	–	350	–	280	–	ns
51	E width low	900	–	700	–	550	–	440	–	ns
53	Clock high to data out invalid	0	–	0	–	0	–	0	–	ns
54	E low to data out invalid	40	–	30	–	20	–	15	–	ns
55	R/$\overline{W}$ to data bus driver	35	–	30	–	20	–	10	–	ns
56[4]	$\overline{HALT}$/$\overline{RESET}$ pulse width	10	–	10	–	10	–	10	–	Clk. per.
57	$\overline{BGACK}$ high to control bus driven	1.5	–	1.5	–	1.5	–	1.5	–	Clk. per.
58[9]	$\overline{BG}$ high to control bus driven	1.5	–	1.5	–	1.5	–	1.5	–	Clk. per.

NOTES:

1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the values given in these columns.
2. Actual value depends on clock period.
3. If #47 is satisfied for both $\overline{DTACK}$ and $\overline{BERR}$, #48 may be 0 nanoseconds.
4. For power up, the MPU must be held in $\overline{RESET}$ state for 100ms to allow stabilization of on-chip circuitry. After the system is powered up, #56 refers to the minimum pulse width required to reset the system.
5. #14, #14A, and #28 are one clock period less than the given number for T6E, BF4, and R9M mask sets.
6. If the asynchronous set-up time (#47) requirements are satisfied, the $\overline{DTACK}$ low-to-data set-up time (#31) requirement can be ignored. The data must only satisfy the data-in clock-low set-up time (#27) for the following cycle.
7. For T6E, BF4, and R9M mask set #11A timing equals #11, and #21A equals #21. #20A may be 0 for T6E, BF4, and R9M mask sets.
8. When $\overline{AS}$ and R/$\overline{W}$ are equally loaded (±20%), subtract 10 nanoseconds from the values given in these columns.
9. The processor will negate $\overline{BG}$ and begin driving the bus again if external arbitration logic negates $\overline{BR}$ before asserting $\overline{BGACK}$.
10. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.
11. The falling edge of S6 triggers both the negation of the strobes ($\overline{AS}$ and $\overline{xDS}$) and the falling edge of E. Either of these events can occur first, depending upon the loading on each signal. Specification #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of the E clock.

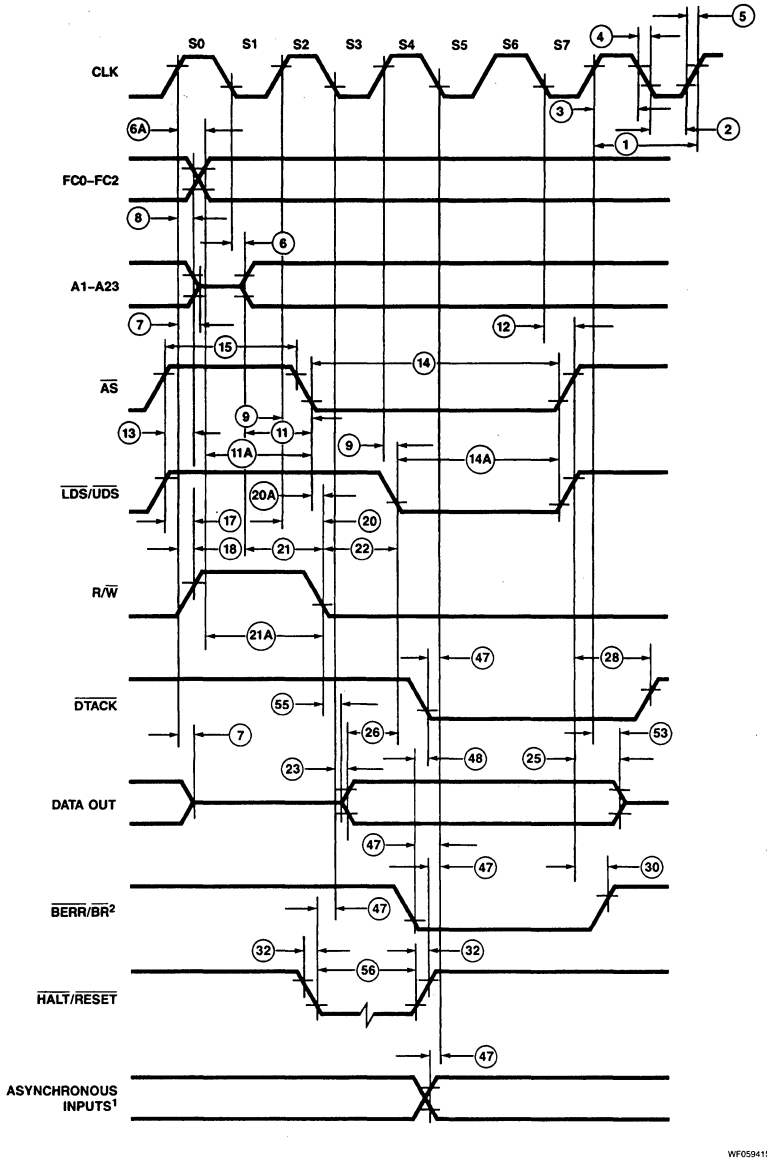
These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



NOTES:
1. Set-up time for the asynchronous inputs BGACK, IPL0-2, and VPA guarantees their recognition at the next falling edge of the clock.
2. BR need fall at this time only in order to insure being recognized at the end of this bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8V and a high voltage of 2.0V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8V and 2.0V.

Figure 47. Read Cycle Timing Diagram

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



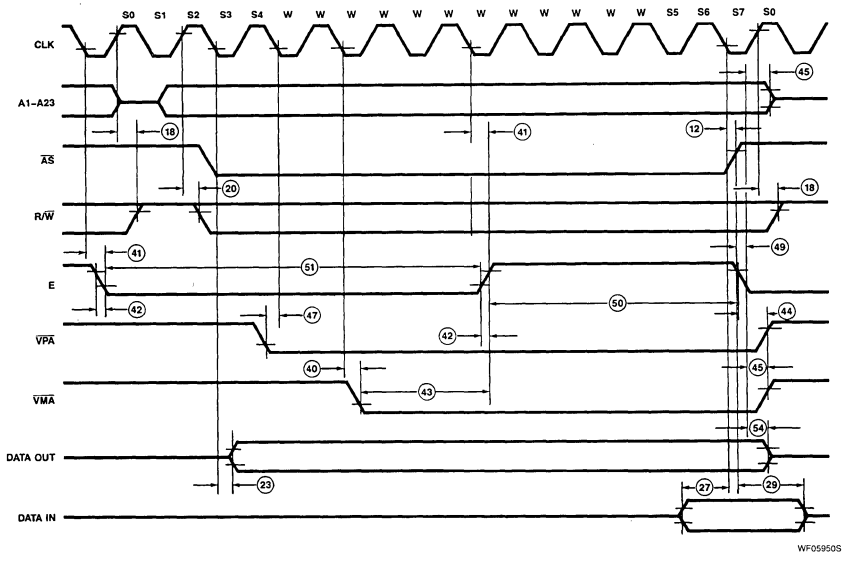
NOTES:
1. Timing measurements are referenced to and from a low voltage of 0.8V and a high voltage of 2.0V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8V and 2.0V.
2. Because of loading variations, R/$\overline{W}$ may be valid after $\overline{AS}$ even though both are initiated by the rising edge of S2 (Specification 20A).

Figure 48. Write Cycle Timing Diagram

AC ELECTRICAL CHARACTERISTICS – SCN68000 To Synchronous Peripheral V_{CC} = 5.0Vdc ± 5%; GND = 0Vdc, $T_A = T_L$ to T_H (refer to figures 49 and 50)

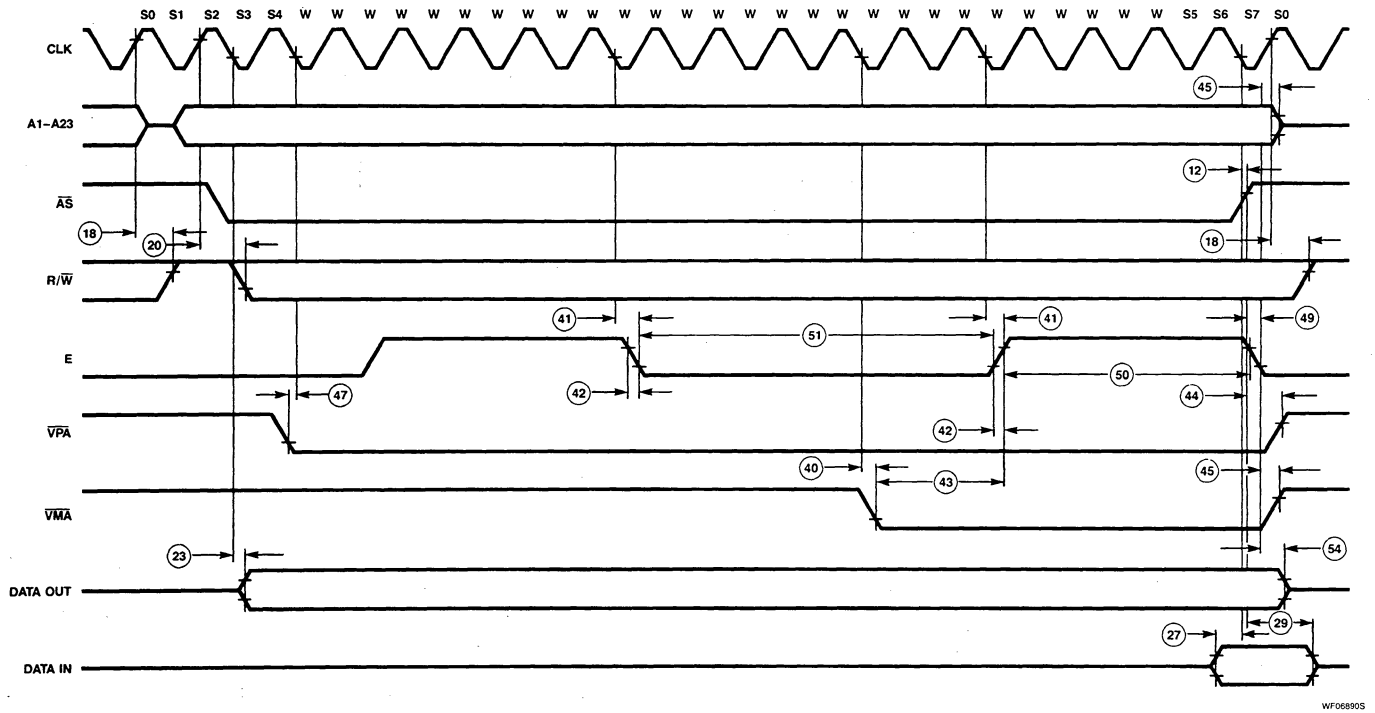
NO.	CHARACTERISTIC	6MHz		8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	Min	Max	
12	Clock low to $\overline{AS}$, $\overline{DS}$ high	–	70	–	70	–	55	–	50	ns
18	Clock high to R/$\overline{W}$ high	0	70	0	70	0	60	0	60	ns
20	Clock high to R/$\overline{W}$ low (write)	–	70	–	70	–	60	–	60	ns
23	Clock low to data out valid (write)	–	80	–	70	–	55	–	55	ns
27	Data in to clock low (set-up time on read)	25	–	15	–	10	–	10	–	ns
29	$\overline{AS}$, $\overline{DS}$ high to data in invalid (hold time on read)	0	–	0	–	0	–	0	–	ns
40	Clock low to $\overline{VMA}$ low	–	80	–	70	–	70	–	70	ns
41	Clock low to E transition	–	85	–	70	–	55	–	45	ns
42	E output rise and fall time	–	25	–	25	–	25	–	25	ns
43	$\overline{VMA}$ low to E high	240	–	200	–	150	–	90	–	ns
44	$\overline{AS}$, $\overline{DS}$ high to $\overline{VPA}$ high	0	160	0	120	0	90	0	70	ns
45	E low to control, address bus invalid (address hold time)	35	–	30	–	10	–	10	–	ns
47	Asynchronous input set-up time	25	–	20	–	20	–	20	–	ns
49[1]	$\overline{AS}$, $\overline{DS}$ high to E low	–80	–	–70	70	–55	55	–45	45	ns
50	E width high	600	–	450	–	350	–	280	–	ns
51	E width low	900	–	700	–	550	–	440	–	ns
54	E low to data out valid	40	–	30	–	20	–	15	–	ns

NOTE:
1. The falling edge of S6 triggers both the negation of the strobes ($\overline{AS}$ and $\overline{xDS}$) and the falling edge of E. Either of these events can occur first, depending upon the loading on each signal. Specification #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of the E clock.



NOTE:
1. This timing diagram is included for those who wish to design their own circuit to generate VMA. It shows the best case possibly attainable.

Figure 49. SCN68000 to Synchronous Peripheral Timing Diagram—Best Case



NOTE:
1. This timing diagram is included for those who wish to design their own circuit to generate VMA. It shows the worst case possibly attainable.

Figure 50. SCN68000 to Synchronous Peripheral Timing Diagram—Worst Case

AC ELECTRICAL CHARACTERISTICS — Bus Arbitration V_{CC} = 5.0Vdc ±5%; GND = 0Vdc; $T_A = T_L$ to T_H (see figures 51, 52, 53)

NO.	CHARACTERISTIC	6MHz		8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	Min	Max	
7	Clock high to address, data bus high impedence	–	80	–	80	–	70	–	60	ns
16	Clock high to control bus high impedence	–	80	–	80	–	70	–	60	ns
33	Clock high to $\overline{BG}$ low	–	80	–	70	–	60	–	50	ns
34	Clock high to $\overline{BG}$ high	–	80	–	70	–	60	–	50	ns
35	$\overline{BR}$ low to $\overline{BG}$ low	1.5	3.5	1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	Clk. per.
36[1]	$\overline{BR}$ high to $\overline{BG}$ high	1.5	3.5	1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	Clk. per.
37	$\overline{BGACK}$ low to $\overline{BG}$ high	1.5	3.0	1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	Clk. per.
37A[2]	$\overline{BGACK}$ low to $\overline{BR}$ high	25	–	20	1.5 Clocks	20	1.5 Clocks	20	1.5 Clocks	ns
38	$\overline{BG}$ low to control, address, data bus high impedance ($\overline{AS}$ high)	–	100	–	80	–	70	–	60	ns
39	$\overline{BG}$ width high	1.5	–	1.5	–	1.5	–	1.5	–	Clk. per.
46	$\overline{BGACK}$ width low	1.5	–	1.5	–	1.5	–	1.5	–	Clk. per.
47	Asynchronous input set-up time	20	–	20	–	20	–	20	–	ns
57	$\overline{BGACK}$ high to control bus driven	1.5	–	1.5	–	1.5	–	1.5	–	Clk. per.
58[1]	$\overline{BG}$ high to control bus driven	1.5	–	1.5	–	1.5	–	1.5	–	Clk. per.

NOTES:
1. The processor will negate $\overline{BG}$ and begin driving the bus again if external arbitration logic negates $\overline{BR}$ before asserting $\overline{BGACK}$.
2. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.

Figures 51, 52, and 53 depict the three bus arbitration cases that can arise. Figure 51 shows the timing where $\overline{AS}$ is negated when the processor asserts $\overline{BG}$ (idle bus case). Figure 52 shows the timing where $\overline{AS}$ is asserted when the processor asserts $\overline{BG}$ (active bus case). Figure 53 shows the timing where more than one bus master are requesting the bus. Refer to **Bus Arbitration** for a complete discussion of bus arbitration.

The waveforms shown in figures 51, 52, and 53 should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

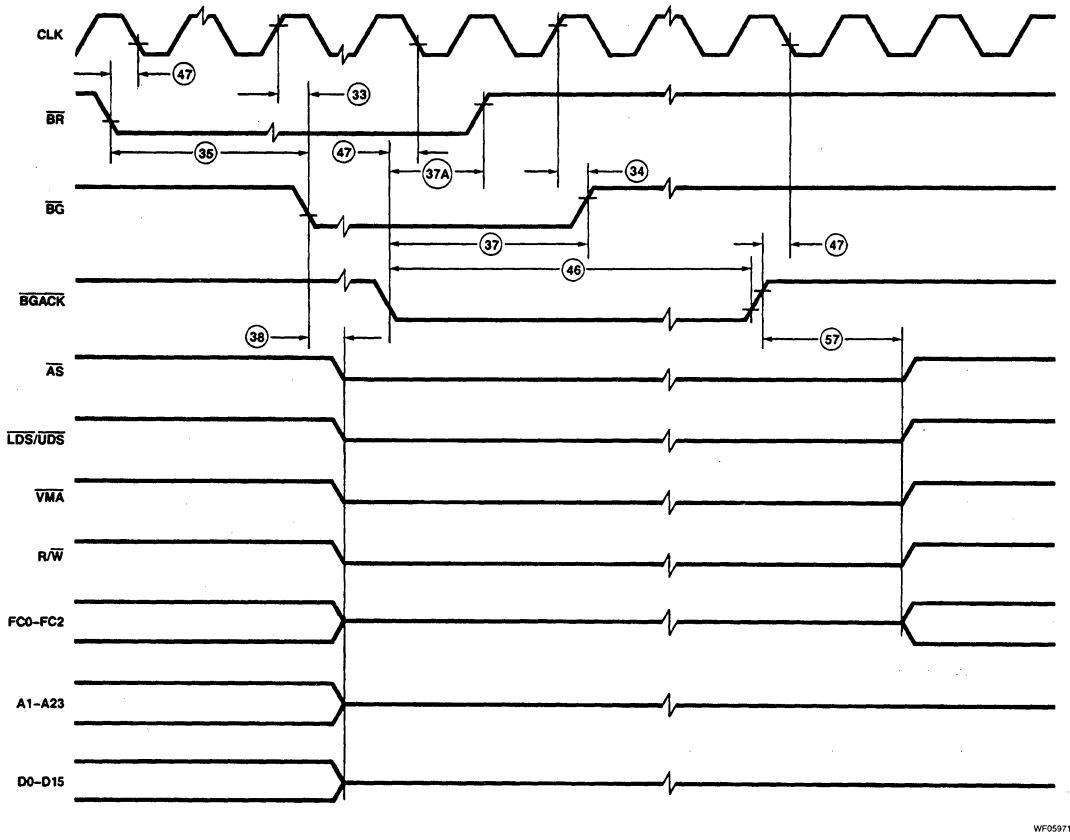


Figure 51. Bus Arbitration Timing Diagram — Idle Bus Case

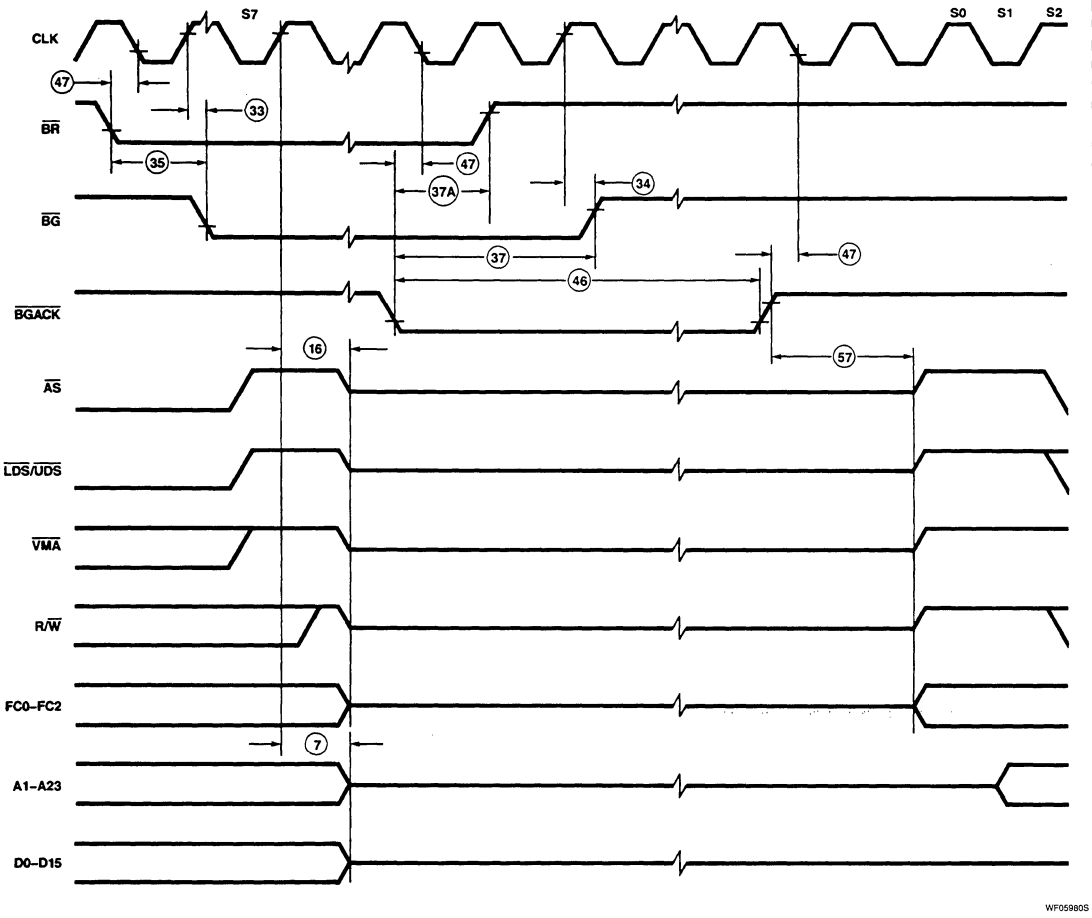


Figure 52. Bus Arbitration Timing Diagram — Active Bus Case

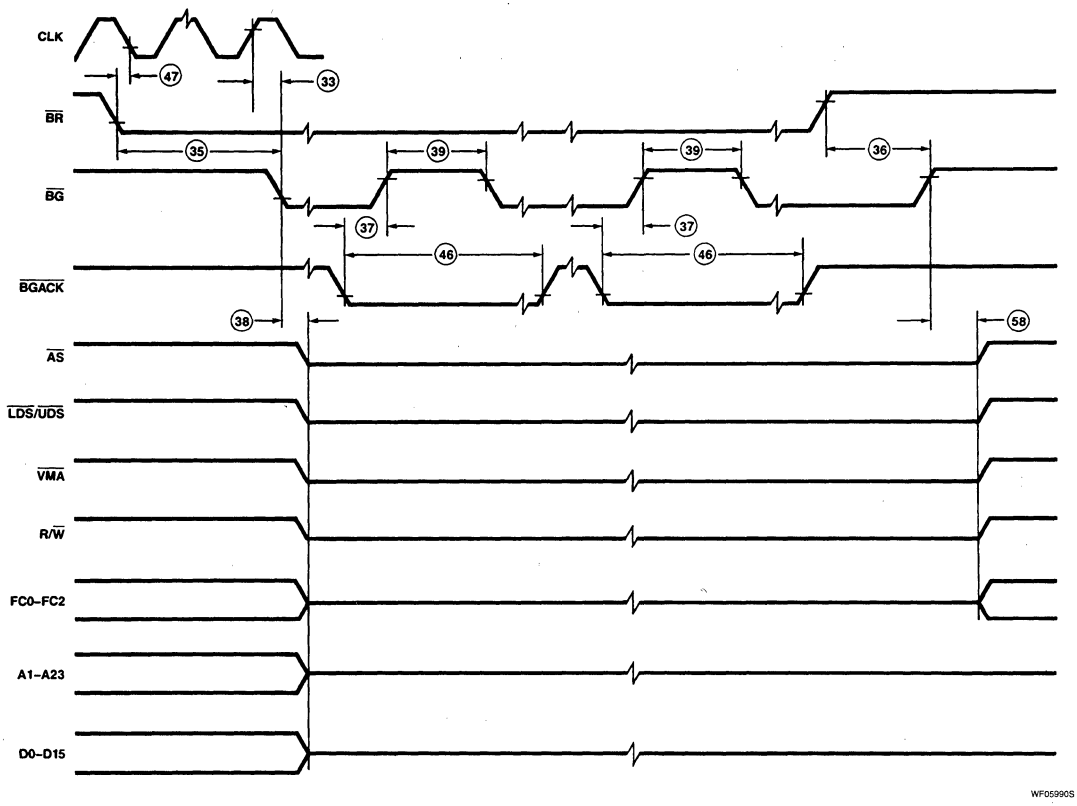


Figure 53. Bus Arbitration Timing Diagram — Multiple Bus Requests

SCN68010

16-Bit Virtual Memory Microprocessor

Preliminary Specification

DESCRIPTION

The SCN68010 is the third member of a family of advanced microprocessors from Signetics. Utilizing VLSI technology, the SCN68010 is a fully-implemented 16-bit microprocessor with 32-bit registers, a rich basic instruction set, and versatile addressing modes.

The SCN68010 is fully object code compatible with the earlier members of the S68000 family and has the added features of virtual memory support and enhanced instruction execution timing.

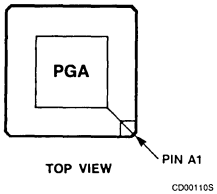
The SCN68010 possesses an asynchronous bus structure with a 24-bit address bus and a 16-bit data bus.

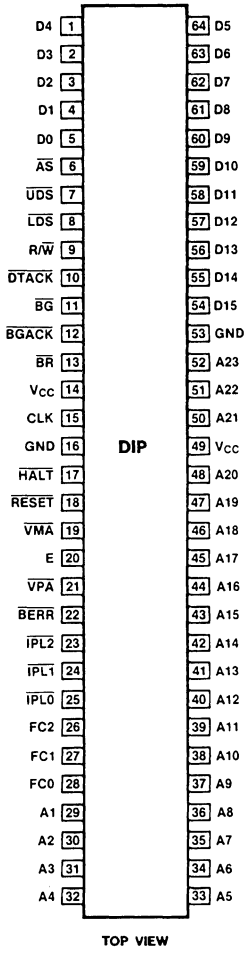
The resources available to the SCN68010 user consist of the following:

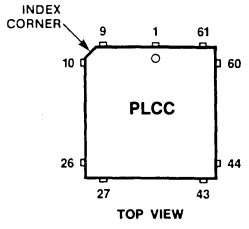
- 17 32-bit data and address registers
- 16-megabyte direct addressing range
- Virtual memory/machine support
- 57 powerful instruction types
- High performance looping instructions
- Operations on five main data types
- Memory mapped I/O
- 14 addressing modes

As shown in the programming model (figures 1 and 2), the SCN68010 offers 17 32-bit general purpose registers, a 32-bit program counter, a 16-bit status register, a 32-bit vector base register, and two 3-bit alternate function code registers. The first eight registers (D0 – D7) are used as data registers for byte (8-bit), word (16-bit), and long word (32-bit) operations. The second set of seven registers (A0 – A6) and the stack pointers (SSP, USP) may be used as software stack pointers and base address registers. In addition, the address registers may be used for word and long word operations. All of the 17 registers may be used as index registers.

PIN CONFIGURATION







PLCC	Function	PLCC	Function	PGA	Function	PGA	Function
1	D4	35	A4	A1	NC	F1	$\overline{HALT}$
2	D3	36	A5	A2	AS	F2	$\overline{RESET}$
3	D2	37	A6	A3	D1	F9	A18
4	D1	38	A7	A4	D2	F10	A19
5	D0	39	A8	A5	D4	G1	$\overline{VMA}$
6	AS	40	A9	A6	D5	G2	$\overline{VPA}$
7	$\overline{UDS}$	41	A10	A7	D7	G9	A15
8	$\overline{LDS}$	42	A11	A8	D8	G10	A17
9	R/$\overline{W}$	43	A12	A9	D10	H1	E
10	$\overline{DTACK}$	44	A13	A10	D12	H2	$\overline{IPL2}$
11	$\overline{BG}$	45	A14	B1	$\overline{DTACK}$	H3	$\overline{IPL1}$
12	$\overline{BGACK}$	46	A15	B2	$\overline{LDS}$	H8	A13
13	$\overline{BR}$	47	A16	B3	$\overline{UDS}$	H9	A12
14	V_{CC}	48	A17	B4	D0	H10	A16
15	CLK	49	A18	B5	D3	J1	$\overline{BERR}$
16	GND	50	A19	B6	D6	J2	$\overline{IPL0}$
17	GND	51	A20	B7	D9	J3	FC1
18	NC	52	V_{CC}	B8	D11	J4	NC
19	$\overline{HALT}$	53	A21	B9	D13	J5	A2
20	$\overline{RESET}$	54	A22	B10	D15	J6	A5
21	$\overline{VMA}$	55	A23	C1	$\overline{BGACK}$	J7	A8
22	E	56	GND	C2	$\overline{BG}$	J8	A10
23	$\overline{VPA}$	57	GND	C3	R/$\overline{W}$	J9	A11
24	$\overline{BERR}$	58	D15	C8	D13	J10	A14
25	$\overline{IPL2}$	59	D14	C9	A23	K1	NC
26	$\overline{IPL1}$	60	D13	C10	A22	K2	FC2
27	$\overline{IPL0}$	61	D12	D1	$\overline{BR}$	K3	FC0
28	FC2	62	D11	D2	V_{CC}	K4	A1
29	FC1	63	D10	D9	V_{SS}	K5	A3
30	FC0	64	D9	D10	A21	K6	A4
31	NC	65	D8	E1	CLK	K7	A6
32	A1	66	D7	E2	V_{SS}	K8	A7
33	A2	67	D6	E9	V_{CC}	K9	A9
34	A3	68	D5	E10	A20	K10	NC

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C		
	8MHz	10MHz	12.5MHz
Ceramic DIP	SCN68010C8I64	SCN68010CAI64	SCN68010CBI64
Plastic DIP	SCN68010C8N64	SCN68010CAN64	SCN68010CBN64
Plastic LCC	SCN68010C8A68	SCN68010CAA68	SCN68010CBA68
Pin Grid Array	SCN68010C8P68	SCN68010CAP68	SCN68010CBP68

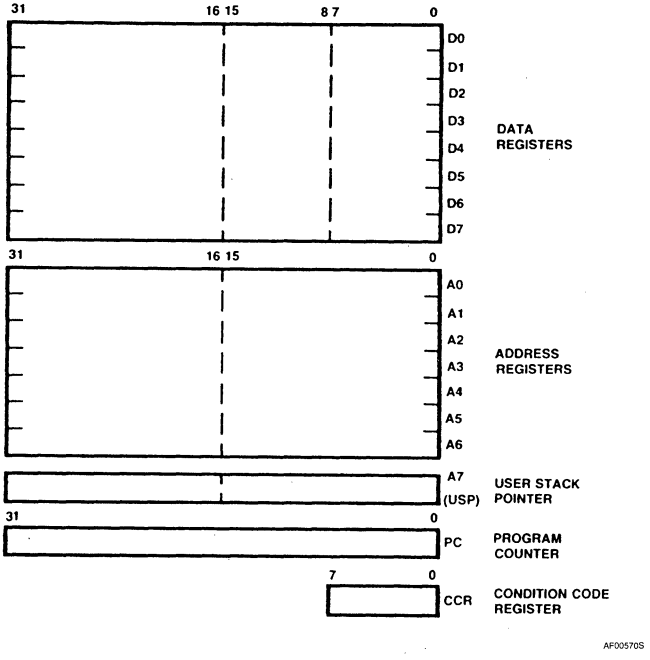


Figure 1. User Programming Model

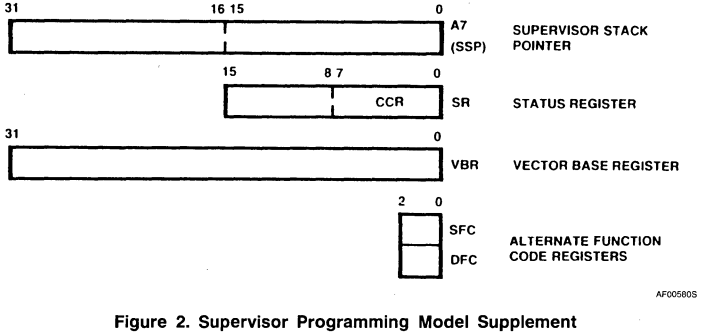


Figure 2. Supervisor Programming Model Supplement

The status register (figure 3) contains the interrupt mask (eight levels available) as well as the condition codes; extend (X), negative (N), zero (Z), overflow (V), and carry (C). Additional status bits indicate that the processor is in the trace (T) mode and in the supervisor (S) or user state.

The vector base register is used to determine the location of the exception vector table in memory to support multiple vector tables. The alternate function code registers allow the supervisor to access user data space or emulate CPU space cycles.

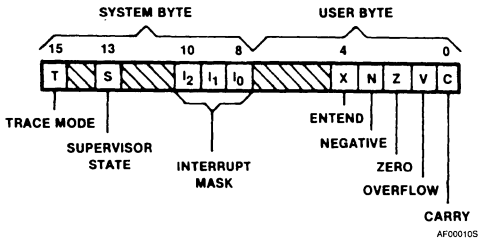


Figure 3. Status Register

Data Types and Addressing Modes

Five basic data types are supported. These data types are:

- Bits
- BCD digits (4 bits)
- Bytes (8 bits)
- Words (16 bits)
- Long words (32 bits)

In addition, operations on other data types such as memory addresses, status word data, etc., are provided in the instruction set.

The 14 address modes, shown in table 1, include six basic types:

- Register direct
- Register indirect
- Absolute
- Program counter relative
- Immediate
- Implied

Included in the register indirect addressing modes is the capability to do postincrementing, predecrementing, offsetting, and indexing. The program counter relative mode can also be modified via indexing and offsetting.

Table 1. ADDRESSING MODES

MODE	GENERATION
Register direct addressing	
Data register direct	EA = Dn
Address register direct	EA = An
Absolute data addressing	
Absolute short	EA = (next word)
Absolute long	EA = (next two words)
Program counter relative addressing	
Relative with offset	EA = (PC) + d_{16}
Relative with index and offset	EA = (PC) + (Xn) + d_8
Register indirect addressing	
Register indirect	EA = (An)
Postincrement register indirect	EA = (An), An ← An + N
Predecrement register indirect	An ← An – N, EA = (An)
Register indirect with offset	EA = (An) + d_{16}
Indexed register indirect with offset	EA = (An) + (Xn) + d_8
Immediate data addressing	
Immediate	DATA = next word(s)
Quick immediate	inherent data
Implied addressing	
Implied register	EA = SR, USP, SSP, PC VBR, SFC, DFC

NOTES:
EA = Effective address
An = Address register
Dn = Data register
Xn = Address or data register used as index register
SR = Status register
PC = Program counter
( ) = Contents of
d_8 = 8-bit offset (displacement)
d_{16} = 16-bit offset (displacement)
N = 1 for byte, 2 for word, and 4 for long word. If An is the stack pointer and the operand size is byte, N = 2 to keep the stack pointer on a word boundary.
← = Replaces

Instruction Set Overview

The SCN68010 instruction set is shown in table 2. Some additional instructions are variations, or subsets, of these and they appear in table 3. Special emphasis has been given to the instruction set's support of structured high-level languages to facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words and most instructions can use any of the 14 addressing modes. By combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned multiply and divide, "quick" arithmetic operations, BCD arithmetic, and expanded operations (through traps). Also, 33 instructions may be used in the loop mode with certain addressing modes and the DBcc instruction to provide 230 high performance string, block manipulation, and extended arithmetic operations.

Table 2. INSTRUCTION SET

MNEMONIC	DESCRIPTION
ABCD*	Add decimal with extend
ADD*	Add
AND*	Logical AND
ASL*	Arithmetic shift left
ASR*	Arithmetic shift right
B_{CC}	Branch conditionally
BCHG	Bit test and change
BCLR	Bit test and clear
BRA	Branch always
BSET	Bit test and set
BSR	Branch to subroutine
BTST	Bit test
CHK	Check register against bounds
CLR*	Clear operand
CMP*	Compare
DB_{CC}	Decrement and branch conditionally
DIVS	Signed divide
DIVU	Unsigned divide
EOR*	Exclusive OR
EXG	Exchange registers
EXT	Sign extend
JMP	Jump
JSR	Jump to subroutine
LEA	Load effective address
LINK	Link stack
LSL*	Logical shift left
LSR*	Logical shift right
MOVE*	Move source to destination
MULS	Signed multiply
MULU	Unsigned multiply
NBCD*	Negate decimal with extend
NEG*	Negate
NOP	No operation
NOT*	One's complement
OR*	Logical OR
PEA	Push effective address
RESET	Reset external devices
ROL*	Rotate left without extend
ROR*	Rotate right without extend
ROXL*	Rotate left with extend
ROXR*	Rotate right with extend
RTD	Return and deallocate
RTE	Return from exception
RTR	Return and restore
RTS	Return from subroutine
SBCD*	Subtract decimal with extend
S_{CC}	Set conditional
STOP	Stop
SUB	Subtract
SWAP	Swap data register halves
TAS	Test and set operand
TRAP	Trap
TRAPV	Trap on overflow
TST*	Test
UNLK	Unlink

* Loopable instructions

Table 3. VARIATIONS OF INSTRUCTION TYPES

INSTRUCTION TYPE	VARIATION	DESCRIPTION
ADD	ADD*	Add
	ADDA	Add address
	ADDQ	Add quick
	ADDI	Add immediate
	ADDX*	Add with extend
AND	AND*	Logical AND
	ANDI	AND immediate
	ANDI to CCR	AND immediate to condition codes
	ANDI to SR	AND immediate to status register
CMP	CMP*	Compare
	CMPA*	Compare address
	CMPM*	Compare memory
	COMPI	Compare immediate
EOR	EOR*	Exclusive OR
	EORI	Exclusive OR immediate
	EORI to CCR	Exclusive OR immediate to condition codes
	EORI to SR	Exclusive OR immediate to status register
MOVE	MOVE*	Move source to destination
	MOVEA*	Move address
	MOVEC	Move control register
	MOVEM	Move multiple registers
	MOVEP	Move peripheral data
	MOVEQ	Move quick
	MOVES	Move alternate address space
	MOVE from SR	Move from status register
	MOVE to SR	Move to status register
	MOVE from CCR	Move from condition codes
	MOVE to CCR	Move to condition codes
	MOVE USP	Move user stack pointer
NEG	NEG*	Negate
	NEGX*	Negate with extend
OR	OR*	Logical OR
	ORI	OR immediate
	ORI to CCR	OR immediate to condition codes
	ORI to SR	OR immediate to status register
SUB	SUB*	Subtract
	SUBA*	Subtract address
	SUBI	Subtract immediate
	SUBQ	Subtract quick
	SUBX*	Subtract with extend

* Loopable instructions

Virtual Memory/Machine Concepts

In most systems using the SCN68010 as the central processor, only a fraction of the 16 megabyte address space will actually contain physical memory. However, by using virtual memory techniques the system can be made to appear to the user to have 16 megabytes of physical memory available to him/her. These techniques have been used for several years in large mainframe computers and more recently in minicomputers and now, with the SCN68010, can be fully supported in microprocessor-based systems.

In a virtual memory system, a user program can be written as though it has a large amount of memory available to it when only a small amount of memory is physically present in the system. In a similar fashion, a system can be designed in such a manner as to allow user programs to access other types of devices that are not physically present in the system such as tape drives, disk drives, printers, or CRTs. With proper software emulation, a physical system can be made to appear to a user program as any other computer system and the program may be given full access to all of the resources of that emulated system. Such an emulated system is called a virtual machine.

Virtual Memory

The basic mechanisim for supporting virtual memory in computers is to provide only a limited amount of high-speed physical memory that can be accessed directly by the processor while maintaining an image of a much larger "virtual" memory on secondary storage devices such as large capacity disk drives. When the processor attempts to access a location in the virtual memory map that is not currently residing in physical memory (referred to as a page fault), the access to that location is temporarily suspended while the necessary data is fetched from the secondary storage and placed in physical memory; the suspended access is then completed. The SCN68010 provides hardware support for virtual memory with the capability of suspending an instruction's execution when a bus error is signaled and then completing the instruction after the physical memory has been updated as necessary.

The SCN68010 uses instruction continuation rather than instruction restart to support virtual memory. With instruction restart, the processor must remember the exact state of the system before each instruction is started in order to restore that state if a page fault occurs during its execution. Then, after the page fault has been repaired, the entire instruction that caused the fault is reexecuted. With instruction continuation, when a page fault occurs the processor stores its internal state and then, after the page fault is repaired, restores that internal state and continues execution of the instruction. In order for the SCN68010 to utilize instruction continuation, it stores its internal state on the supervisor stack when a bus cycle is terminated with a bus error signal. It then loads the program counter from vector table entry number two (offset $008) and resumes program execution at that new address. When the bus error exception handler routine has completed execution, an RTE instruction is executed which reloads the SCN68010 with the internal state stored on the stack, re-runs the faulted bus cycle, and continues the suspended instruction. Instruction continuation has the additional advantage of allowing hardware support for virtual I/O devices. Since virtual registers may be simulated in the memory map, an access to such a register will cause a fault and the function of the register can be emulated by software.

Virtual Machine

One typical use for a virtual machine system is in the development of software such as an

operating system for another machine with hardware also under development and not available for programming use. In such a system, the governing operating system (OS) emulates the hardware of the new system and allows the new OS to be executed and debugged as though it were running on the new hardware. Since the new OS is controlled by the governing OS, the new one must execute at a lower privilege level than the governing OS so that any attempts by the new OS to use virtual resources that are not physically present, and should be emulated, will be trapped by the governing OS and handled in software. In the SCN68010, a virtual machine may be fully supported by running the new OS in the user mode and the governing OS in the supervisor mode so that any attempts to access supervisor resources or execute privileged instructions by the new OS will cause a trap to the governing OS.

In order to fully support a virtual machine, the SCN68010 must protect the supervisor resources from access by user programs. The one supervisor resource that is not fully protected in the SCN68000 is the system byte of the status register. In the SCN68000, the MOVE from SR instruction allows user programs to test the S bit (in addition to the 1 bit and interrupt mask) and thus determine that they are running in the user mode. For full virtual machine support, a new OS must not be aware of the fact that it is running in the user mode and thus should not be allowed to access the S bit. For this reason, the MOVE from SR instruction on the SCN68010 is a privileged instruction and the MOVE from CCR instruction has been added to allow user programs unhindered access to the condition codes. By making the MOVE from SR instruction privileged, when the new OS attempts to access the S bit, a trap to the governing OS will occur and the SR image passed to the new OS by the governing OS will have the S bit set.

DATA ORGANIZATION AND ADDRESSING CAPABILITIES

This section contains a description of the registers and the data organization of the SCN68010.

Operand Size

Operand sizes are defined as follows: a byte equals 8 bits, a word equals 16 bits, and a long word equals 32 bits. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation. Implicit instructions support some subset of all three sizes.

Data Organization in Registers

The eight data registers support data operands of 1, 8, 16, or 32 bits. The seven address registers and the stack pointers support address operands of 32 bits. The four control registers support data of 1, 3, 8, 16, or 32 bits depending on the register specified.

Data Registers

Each data register is 32 bits wide. Byte operands occupy the low order 8 bits, word operands the low order 16 bits, and long word operands the entire 32 bits. The least significant bit is addressed as bit zero; the most significant bit is addressed as bit 31.

When a data register is used as either a source or destination operand, only the appropriate low order portion is changed; the remaining high order portion is neither used nor changed.

Address Registers

Each address register and stack pointer is 32 bits wide and holds a full 32-bit address. Address registers do not support the sized operands. Therefore, when an address register is used as a source operand, either the low order word or the entire long word operand is used depending upon the operation size. When an address register is used as the destination operand, the entire register is affected regardless of the operation size. If the operation size is word, any other operands are sign extended to 32 bits before the operation is performed.

Control Registers

The status register (SR) is 16 bits wide with the lower byte being accessed as the condition code register (CCR). Not all 16 bits of the SR are defined and will be read as zeroes and ignored when written. Operations to the CCR are word operations; however, the upper byte will be read as all zeros and ignored when written.

The vector base register (VBR) is 32 bits wide and holds a full 32-bit address. All operations involving the VBR are long word operations regardless of whether it is the source or destination operand.

The alternate function code registers (SFC and DFC) are three bits wide and contain the function code values placed on FC0 – FC2 during the operand read or write of a MOVES instruction. All transfers to or from the alternate function code registers are 32 bits although the upper 29 bits will be read as zeroes and ignored when written.

Data Organization in Memory

The data types supported by the SCN68010 are: bit data, integer data of 8, 16, or 32 bits, 32-bit addresses and binary coded decimal data. Each of these data types is put in memory, as shown in figure 4. The numbers indicate the order in which the data would be accessed from the processor.

Bytes are individually addressable with the high order byte having an even address the same as the word, as shown in figure 5. The low order byte has an odd address that is one count higher than the word address. Instructions and word or long word data are accessed only on word (even byte) boundaries. If a long word datum is located at address n (n even), then the low-order word of that datum is located at address n + 2.

Addressing

Instructions for the SCN68010 contain two kinds of information: the type of function to be performed and the location of the operand(s) on which to perform that function. The methods used to locate (address) the operand(s) are explained in the following paragraphs.

Instructions specify an operand location in one of three ways:

Register Specification – the number of the register is given in the register field of their instruction.

Effective Address – use of the different effective addressing modes.

Implicit Reference – the definition of certain instructions implies the use of specific registers.

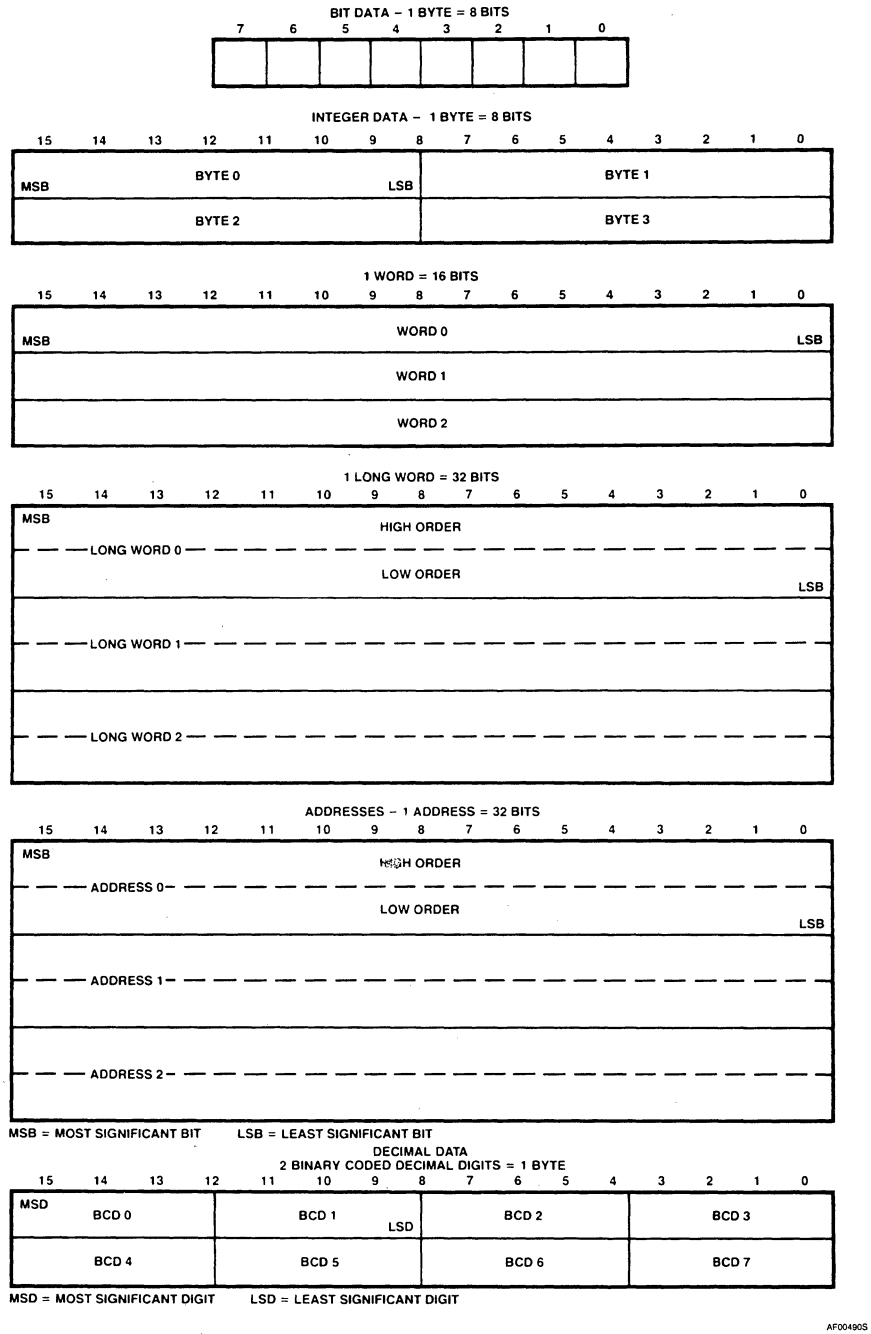


Figure 4. Memory Data Organization

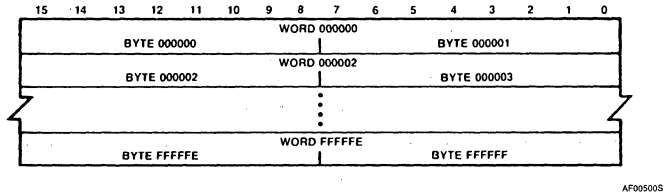


Figure 5. Word Organization in Memory

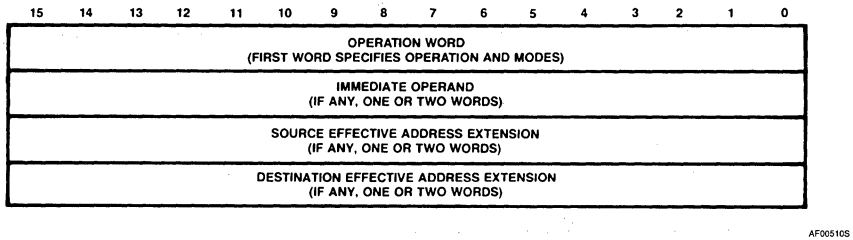


Figure 6. Instruction Operation Word General Format

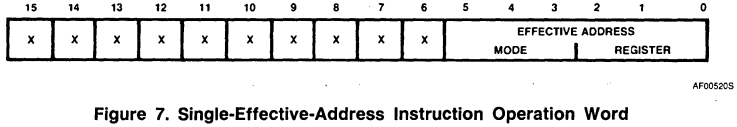


Figure 7. Single-Effective-Address Instruction Operation Word

Instruction Format

Instructions are from one to five words in length as shown in figure 6. The length of the instruction and the operation to be performed is specified by the first word of the instruction which is called the operation word. The remaining words further specify the operands. These words are either immediate operands or extensions to the effective address mode specified in the operation word.

Program/Data References

The SCN68010 separates memory references into two classes: program references and data references. Program references, as the name implies, are references to that section of memory that contains the program being executed. Data references refer to that section of memory that contains data. Generally, operand reads are from the data space. All operand writes are to the data space.

Register Specification

The register field within an instruction specifies the register to be used. Other fields within the instruction specify whether the register selected is an address or data register and how the register is to be used.

Effective Address

Most instructions specify the location of an operand by using the effective address field in the operation word. For example, figure 7 shows the general format of the single-effective-address instruction operation word. The effective address is composed of two 3-bit fields: the mode field and the register field. The value in the mode field selects the different address modes. The register field contains the number of a register.

The effective address field may require additional information to fully specify the operand. This additional information, called the effective address extension, is contained in the following word or words and is considered part of the instruction, as shown in figure 6. The effective address modes are grouped into three categories: register direct, memory addressing, and special.

Register Direct Modes

These effective addressing modes specify that the operand is in one of sixteen general purpose registers or one of four control registers.

Data Register Direct — The operand is in the data register specified by the effective address register field.

Address Register Direct — The operand is in the address register specified by the effective address register field.

Memory Address Modes

These effective addressing modes specify that the operand is in memory and provide the specific address of the operand.

Address Register Indirect — The address of the operand is in the address register specified by the register field. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Address Register Indirect with Post Increment — The address of the operand is in the address register specified by the register field. After the operand address is used, it is incremented by one, two, or four depending upon whether the size of the operand is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is incremented by two rather than one to keep the stack pointer on a

word boundary. The reference is classified as a data reference.

Address Register Indirect with Predecrement — The address of the operand is in the address register specified by the register field. Before the operand address is used, it is decremented by one, two, or four depending upon whether the operand size is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is decremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.

Address Register Indirect with Displacement — This addressing mode requires one word of extension. The address of the operand is the sum of the address in the address register and the sign-extended 16-bit displacement integer in the extension word. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Address Register Indirect with Index — This addressing mode requires one word of extension. The address of the operand is the sum of the address in the address register, the sign-extended displacement integer in the low order eight bits of the extension word, and the contents of the index register. The index may be specified as the sign extended low-order word or the long word in the index register. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Special Address Modes

The special address modes use the effective address register field to specify the special addressing mode instead of a register number.

Absolute Short Address — This addressing mode requires one word of extension. The address of the operand is in the extension word. The 16-bit address is sign extended before it is used. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Absolute Long Address — This addressing mode requires two words of extension. The address of the operand is developed by the concatenation of the extension words. The high order part of the address is the first extension word; the low order part of the address is the second extension word. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Program Counter with Displacement — This addressing mode requires one word of extension. The address of the operand is the sum of the address in the program counter and the sign-extended 16-bit displacement integer in the extension word. The value in the program counter is the address of the extension word. The reference is classified as a program reference.

Table 4. EFFECTIVE ADDRESS ENCODING SUMMARY

ADDRESSING MODE	MODE	REGISTER
Data register direct	000	Register number
Address register direct	001	Register number
Address register indirect	010	Register number
Address register indirect with postincrement	011	Register number
Address register indirect with predecrement	100	Register number
Address register indirect with displacement	101	Register number
Address register indirect with index	110	Register number
Absolute short	111	000
Absolute long	111	001
Program counter with displacement	111	010
Program counter with index	111	011
Immediate	111	100

Program Counter with Index — This addressing mode requires one word of extension. The address is the sum of the address in the program counter, the sign-extended displacement integer in the lower eight bits of the extension word, and the contents of the index register. The index may be specified as the sign extended low-order word or the long word in the index register. The value in the program counter is the address of the extension word. The reference is classified as a program reference.

Immediate Data — This addressing mode requires either one or two words of extension depending on the size of the operation.

Byte Operation – operand is in the low order byte of extension word

Word Operation – operand is in the extension word

Long Word – Operation operand is in the two extension words, high order 16 bits are in the first extension word, low order 16 bits are in the second extension word.

Implicit Reference — Some instructions make implicit reference to the program counter (PC), the system stack pointer (SP), the supervisor stack pointer (SSP), the user stack pointer (USP), the status register (SR), the condition code register (CCR), the vector base register (VBR), or the alternate function code registers (SFC or DFC).

A selected set of instructions may reference the status register by means of the effective address field. These are:

ANDI to CCR
ANDI to SR
EORI to CCR
EORI to SR
ORI to CCR
ORI to SR
MOVE to CCR
MOVE to SR
MOVE from SR

Effective Address Encoding Summary

Table 4 is a summary of the effective addressing modes discussed in the previous paragraphs.

System Stack

The system stack is used implicitly by many instructions; user stacks and queues may be created and maintained through the addressing modes. Address register seven (A7) is the system stack pointer (SP). The system stack pointer is either the supervisor stack pointer (SSP) or the user stack pointer (USP), depending on the state of the S bit in the status register. If the S bit indicates supervisor state, the SSP is the active system stack pointer and the USP cannot be referenced as an address register. If the S bit indicates user state, the USP is the active system stack pointer, and the SSP cannot be referenced. Each system stack fills from high memory to low memory.

INSTRUCTION SET SUMMARY

This section contains an overview of the form and structure of the SCN68010 instruction set. The instructions form a set of tools that include all the machine functions to perform the following operations:

Data movement
Integer arithmetic
Logical
Shift and rotate
Bit manipulation
Binary code decimal
Program control
System control

The complete range of instruction capabilities combined with the flexible addressing modes

described previously provide a very flexible base for program development.

Data Movement Operations

The basic method of data acquisition (transfer and storage) is provided by the move (MOVE) instruction. The move instruction and the effective addressing modes allow both address and data manipulation. Data movement instructions allow byte, word, and long word operands to be transferred from memory to memory, memory to register, register to memory, and register to register. Address movement instructions allow word and long word operand transfers and ensure that only legal address manipulations are executed. In addition to the general move instruction there are several special data movement instructions: move multiple registers (MOVEM), move peripheral data (MOVEP), exchange registers (EXG), load effective address (LEA), push effective address (PEA), link stack (LINK), unlink stack (UNLK), move quick (MOVEQ), move control register (MOVECS), and move alternate address space (MOVES). Table 5 is a summary of the data movement operations.

Integer Arithmetic Operations

The arithmetic operations include the four basic operations of add (ADD), subtract (SUB), multiply (MUL), and divide (DIV) as well as arithmetic compare (CMP), clear (CLR), and negate (NEG). The add and subtract instructions are available for both address and data operations, with data operations accepting all operand sizes. Address operations are limited to legal address size operands (16 or 32 bits). Data, address, and memory compare operations are also available. The clear and negate instructions may be used on all sizes of data operands.

The multiply and divide operations are available for signed and unsigned operands using word multiply to produce a long word product, and a long word dividend with word divisor to produce a word quotient with a word remainder.

Multiprecision and mixed size arithmetic can be accomplished using a set of extended instructions. These instructions are: add extended (ADDX), subtract extended (SUBX), sign extend (EXT), and negate binary with extend (NEGX).

Table 5. DATA MOVEMENT OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
EXG	32	Rx ↔ Ry
LEA	32	EA → An
LINK	–	(An) → –(SP) (SP) → An (SP) + displacement → SP
MOVE	8, 16, 32	(EA)s → EAd
MOVEC	32	(Rn) → Cr (Cr) → Rn
MOVEM	16, 32	(EA) → An, Dn An, Dn → EA
MOVES	8, 16, 32	(EA) → Rn (Rn) → EA
MOVEP	16, 32	d(An) → Dn Dn → d(An)
MOVEQ	8	#xxx → Dn
PEA	32	EA → –(SP)
SWAP	32	Dn[31:16] ↔ Dn[15:0]
UNLK	–	An → Sp (SP) + → An

NOTES:
s = source
d = destination
[ ] = bit numbers
–( ) = indirect with predecrement
( ) + = indirect with postdecrement
= immediate data

Table 6. INTEGER ARITHMETIC OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
ADD	8, 16, 32 16, 32	(Dn) + (EA) → Dn (EA) + (Dn) → EA (EA) + #xxx → EA (An) + (EA) → An
ADDX	8, 16, 32 16, 32	(Dx) + (Dy) + X → Dx –(Ax) + –(Ay) + X → (Ax)
CLR	8, 16, 32	0 → (EA)
CMP	8, 16, 32 16, 32	(Dn) – (EA) (EA) – #xxx (Ax) + –(Ay) + (An) – (EA)
DIVS	32÷16	(Dn)/(EA) → Dn
DIVU	32÷16	(Dn)/(EA) → Dn
EXT	8 → 16 16 → 32	$(Dn)_8 \rightarrow Dn_{16}$ $(Dn)_{16} \rightarrow Dn_{32}$
MULS	16 x 16 → 32	(Dn) x (EA) → Dn
MULU	16 x 16 → 32	(Dn) x (EA) → Dn
NEG	8, 16, 32	0 – (EA) → EA
NEGX	8, 16, 32	0 – (EA) – X → EA
SUB	8, 16, 32 16, 32	(Dn) – (EA) → Dn (EA) – Dn → EA (EA) – #xxx → EA (An) – (EA) → An
SUBX	8, 16, 32	(Dx) – (Dy) – X → Dx –(Ax) – –(Ay) – X → (Ax)
TAS	8	[EA] – 0, 1 → EA[7]
TST	8, 16, 32	(EA) – 0

NOTES:
[ ] = bit number
= immediate data
–( ) = indirect with predecrement
( ) + = indirect with postdecrement

A test operand (TST) instruction that will set the condition codes as a result of a compare of the operand with zero is also available. Test and set (TAS) is a synchronization instruction useful in multiprocessor systems. Table 6 is a summary of the integer arithmetic operations.

Logical Operations

Logical operation instructions AND, OR, EOR, and NOT are available for all sizes of integer data operands. A similar set of immediate instructions (ANDI, ORI, and EORI) provide these logical operations with all sizes of immediate data. Table 7 is a summary of the logical operations.

Table 7. LOGICAL OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
AND	8, 16, 32	(Dn) ∧ (EA) → Dn (EA) ∧ (Dn) → EA (EA) ∧ #xxx → EA
OR	8, 16, 32	(Dn) ∨ (EA) → Dn (EA) ∨ (Dn) → EA (EA) ∨ #xxx → EA
EOR	8, 16, 32	(EA) ⊕ (Dy) → EA (EA) ⊕ #xxx → EA
NOT	8, 16, 32	~(EA) → EA

NOTES:
= immediate data
~ = invert
∧ = logical AND
∨ = logical OR
⊕ = logical exclusive OR

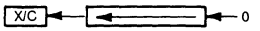
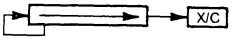
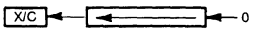
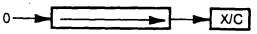
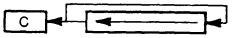
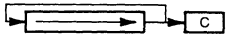
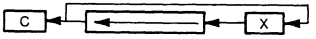
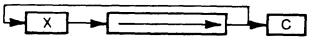
Shift and Rotate Operations

Shift operations in both directions are provided by the arithmetic shift instructions ASR and ASL and logical shift instructions LSR and LSL. The rotate instructions (with and without extend) available are ROXR, ROXL, ROR, and ROL. All shift and rotate operations can be performed in either registers or memory. Register shifts and rotates support all operand sizes and allow a shift count specified in a data register.

Memory shifts and rotates are for word operands only and allow only single-bit shifts or rotates.

Table 8 is a summary of the shift and rotate operations.

Table 8. SHIFT AND ROTATE OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
ASL	8, 16, 32	
ASR	8, 16, 32	
LSL	8, 16, 32	
LSR	8, 16, 32	
ROL	8, 16, 32	
ROR	8, 16, 32	
ROXL	8, 16, 32	
ROXR	8, 16, 32	

Bit Manipulation Operations

Bit manipulation operations are accomplished using the following instructions: bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). Table 9 is a summary of the bit manipulation operations. (Z is bit 2 of the status register.)

Table 9. BIT MANIPULATION OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
BTST	8, 32	~bit of (EA) → Z
BSET	8, 32	~bit of (EA) → Z 1 → bit of EA
BCLR	8, 32	~bit of (EA) → Z 0 → bit of EA
BCHG	8, 32	~bit of (EA) → Z ~bit of (EA) → bit of EA

NOTE:
~ = invert

Binary Coded Decimal Operations

Multiprecision arithmetic operations on binary coded decimal numbers are accomplished using the following instructions: add decimal with extend (ABCD), subtract decimal with extend (SBCD), and negate decimal with extend (NBCD). Table 10 is a summary of the binary coded decimal-operations.

Table 10. BINARY CODED DECIMAL OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
ABCD	8	$(Dx)_{10} + (Dy)_{10} + X \rightarrow Dx$ $-(Ax)_{10} + -(Ay)_{10} + X \rightarrow (Ax)$
SBCD	8	$(Dx)_{10} - (Dy)_{10} - X \rightarrow Dx$ $-(Ax)_{10} - -(Ay)_{10} - X \rightarrow (Ax)$
NBCD	8	$0 - (EA)_{10} - X \rightarrow (EA)$

NOTES:
– = indirect with predecrement.
+ = indirect with postdecrement

Program Control Operations

Program control operations are accomplished using a series of conditional and unconditional branch instructions and return instructions. These instructions are summarized in table 11.

The conditional instructions provide setting and branching for the following conditions:

CC — carry clear
CS — carry set
EQ — equal
F — never true
GE — greater or equal
GT — greater than
HI — high
LE — less or equal
LS — low or same
LT — less than
MI — minus
NE — not equal
PL — plus
T — always true
VC — no overflow
VS — overflow

Table 11. PROGRAM CONTROL OPERATIONS

INSTRUCTION	OPERATION
Conditional	
B_{CC}	Branch conditionally (14 conditions) 8- and 16-bit displacement
DB_{CC}	Test condition, decrement, and branch 16-bit displacement
S_{CC}	Set byte conditionally (16 conditions)
Unconditional	
BRA	Branch always 8- and 16-bit displacement
BSR	Branch to subroutine 8- and 16-bit displacement
JMP	Jump
JSR	Jump to subroutine
Returns	
RTD	Return from subroutine and deallocate stack
RTR	Return and restore condition codes
RTS	Return from subroutine

System Control Operations

System control operations are accomplished by using privileged instructions, trap generating instructions, and instructions that use or modify the condition code register. These instructions are summarized in table 12.

Table 12. SYSTEM CONTROL OPERATIONS

INSTRUCTION	OPERATION
Privileged	
ANDI to SR	Logical AND to status register
EORI to SR	Logical EOR to status register
MOVE EA to SR	Load new status register
MOVE SR to EA	Store status register
MOVE USP	Move user stack pointer
MOVEC	Move control register
MOVES	Move alternate address space
ORI to SR	Logical OR to status register
RESET	Reset external devices
RTE	Return from exception
STOP	Stop program execution
Trap Generating	
CHK	Check data register against upper bounds
TRAP	Trap
TRAPV	Trap on overflow
Condition Code Register	
ANDI to CCR	Logical AND to condition codes
EORI to CCR	Logical EOR to condition codes
MOVE EA to CCR	Load new condition codes
MOVE CCR to EA	Store condition codes
ORI to CCR	Logical OR to condition codes

SIGNAL AND BUS OPERATION DESCRIPTION

This section contains a brief description of the input and output signals. A discussion of bus operation during the various machine cycles and operations is also given.

NOTE

The terms **assertion** and **negation** will be used extensively. This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true, independent of whether that level is represented by a high or low voltage. The term negate or negation is used to indicate that a signal is inactive or false.

Signal Description

The input and output signals can be functionally organized into the groups shown in figure 8. The following paragraphs provide a brief description of the signals and a reference (if applicable) to other paragraphs that contain more detail about the function being performed.

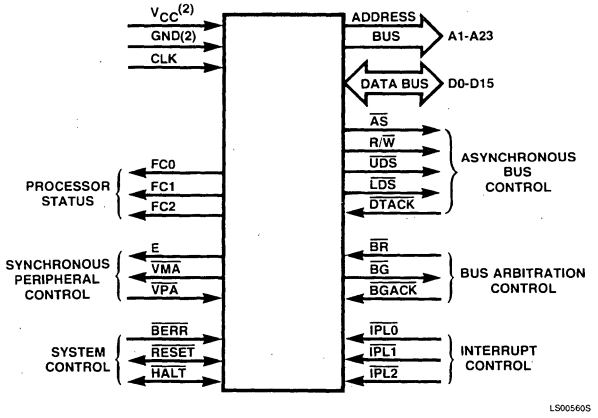


Figure 8. Input and Output Signals

Address Bus (A1 through A23)
This 23-bit, unidirectional, three-state bus is capable of addressing 8 megawords of data. It provides the address for bus operation during all cycles except CPU space cycles.

Data Bus (D0 through D15)
This 16-bit, bidirectional, three-state bus is the general purpose data path. It can transmit and accept data in either word or byte length.

Asynchronous Bus Control
Asynchronous data transfers are handled using the following control signals: address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are explained in the following paragraphs.

Address Strobe ($\overline{AS}$) — This signal indicates that there is a valid address on the address bus.

Read/Write (R/$\overline{W}$) — This signal defines the data bus transfer as a read or write cycle. The R/$\overline{W}$ signal also works in conjunction with the data strobes as explained in the following paragraph.

Upper and Lower Data Strobe ($\overline{UDS}$, $\overline{LDS}$) — These signals control the flow of data on the data bus, as shown in table 13. When the R/$\overline{W}$ line is high, the processor will read from the data bus as indicated. When the R/$\overline{W}$ line is low, the processor will write to the data bus as shown.

Data Transfer Acknowledge ($\overline{DTACK}$) — This input indicates that the data transfer is completed. When the processor recognizes $\overline{DTACK}$ during a read cycle, data is latched one clock cycle later and the bus cycle terminated. When $\overline{DTACK}$ is recognized during a write cycle, the bus cycle is terminated. Refer to **Asynchronous Versus Synchronous Operation.**

Bus Arbitration Control
The three signals, bus request, bus grant, and bus grant acknowledge, form a bus arbitration circuit to determine which device will be the bus master device.

Bus Request ($\overline{BR}$) — This input is wire ORed with all other devices that could be bus masters. This input indicates to the processor that some other device desires to become the bus master.

Bus Grant ($\overline{BG}$) — This output indicates to all other potential bus master devices that the processor will release bus control at the end of the current bus cycle.

Bus Grant Acknowledge ($\overline{BGACK}$) — This input indicates that some other device has become the bus master. This signal should not be asserted until the following four conditions are met:

1. a bus grant has been received,
2. address strobe is inactive which indicates that the microprocessor is not using the bus,
3. data transfer acknowledge is inactive which indicates that neither memory or peripherals are using the bus, and
4. bus grant acknowledge is inactive which indicates that no other device is still claiming bus mastership.

Table 13. DATA STROBE CONTROL OF DATA BUS

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8 – D15	D0 – D7
High	High	–	No valid data	No valid data
Low	Low	High	Valid data bits 8 – 15	Valid data bits 0 – 7
High	Low	High	No valid data	Valid data bits 0 – 7
Low	High	High	Valid data bits 8 – 15	No valid data
Low	Low	Low	Valid data bits 8 – 15	Valid data bits 0 – 7
High	Low	Low	Valid data bits 0 – 7*	Valid data bits 0 – 7
Low	High	Low	Valid data bits 8 – 15	Valid data bits 8 – 15*

*These conditions are a result of current implementation and may not appear on future devices.

Interrupt Control ($\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$)
These input pins indicate the encoded priority level of the device requesting an interrupt. Level seven is the highest priority while level zero indicates that no interrupts are requested. Level seven cannot be masked. The least significant bit is $\overline{IPL0}$ and the most significant bit is $\overline{IPL2}$. These lines must remain stable until the processor signals interrupt acknowledge (FC0 – FC2 are all high, A4 – A23 are all high) to insure that the interrupt is recognized.

System Control
The system control inputs are used to either reset or halt the processor and to indicate to the processor that bus errors have occurred. The three system control inputs are explained in the following paragraphs.

Bus Error ($\overline{BERR}$) — This input informs the processor that there is a problem with the cycle currently being executed. Problems may be a result of:

1. nonresponding devices,
2. interrupt vector number acquisition failure,
3. illegal access request as determined by a memory management unit, or
4. other application dependent errors.

The bus error signal interacts with the halt signal to determine if the current bus cycle should be reexecuted or if exception processing should be performed.

Refer to **Bus Error and Halt Operation** for additional information about the interaction of the $\overline{BERR}$ and $\overline{HALT}$ signals.

Reset ($\overline{RESET}$) — This bidirectional signal line acts to reset (start a system initialization sequence) the processor in response to an external reset signal. An internally generated reset (result of a reset instruction) causes all external devices to be reset and the internal state of the processor is not affected. A total system reset (processor and external devices) is the result of external $\overline{HALT}$ and $\overline{RESET}$ signals applied at the same time. Refer to **Reset Operation** for further information.

Halt ($\overline{HALT}$) — when this bidirectional line is driven by an external device, it will cause the processor to stop at the completion of the current bus cycle. When the processor has been halted using this input, all control signals are inactive and all three-state lines are put in their high-impedance state (refer to table 15). Refer to **Bus Error and Halt Operation** for additional information about the interaction between the $\overline{HALT}$ and $\overline{BERR}$ signals.

When the processor has stopped executing instructions, due to a double bus fault condition (refer to **Double Bus Faults),** the $\overline{HALT}$ line is driven by the processor to indicate to external devices that the processor has stopped.

Peripheral Control
These control signals are used to allow the interfacing of synchronous peripheral devices with the asynchronous SCN68010. These signals are explained in the following paragraphs.

Enable (E) — This signal is the standard enable signal common to all synchronous type peripheral devices. The period for this output is ten SCN68010 clock periods (six

clocks low, four clocks high). Enable is generated by an internal ring counter which may come up in any state (i.e., at power on, it is impossible to guarantee phase relationship of E to CLK). E is a free-running clock and runs regardless of the state of the bus on the MPU.

Valid Peripheral Address ($\overline{VPA}$) — This input indicates that the device addressed is a synchronous family device and that data transfer should be synchronized with the enable (E) signal. This input also indicates that the processor should use automatic vectoring for an interrupt. Refer to **Interface with Synchronous Peripherals.**

Valid Memory Address ($\overline{VMA}$) — This output is used to indicate to synchronous peripheral devices that there is a valid address on the address bus and the processor is synchronized to enable (E). This signal only responds to a valid peripheral address ($\overline{VPA}$) input which indicates that the peripheral is a synchronous family device.

Processor Status (FC0, FC1, FC2).
These function code outputs indicate the state (user or supervisor) and the address space currently being accessed, as shown in table 14. The information indicated by the function code outputs is valid whenever address strobe ($\overline{AS}$) is active.

Clock (CLK)
The clock input is a TTL-compatible signal that is internally buffered for development of the internal clocks needed by the processor. The clock input should not be gated off at any time and the clock signal must conform to minimum and maximum pulse width times.

Signal Summary
Table 15 is a summary of all the signals discussed in the previous paragraphs.

Table 14. FUNCTION CODE OUTPUTS

FUNCTION CODE OUTPUT			CYCLE TYPE
FC2	FC1	FC0	
0	0	0	Undefined, reserved*
0	0	1	User data space
0	1	0	User program space
0	1	1	Undefined, reserved*
1	0	0	Undefined, reserved*
1	0	1	Supervisor data space
1	1	0	Supervisor program space
1	1	1	CPU space

* Address space 3 is reserved for user definition, while 0 and 4 are reserved for future use by Signetics.

Table 15. SIGNAL SUMMARY

SIGNAL NAME	MNEMONIC	INPUT/OUTPUT	ACTIVE STATE	Hi-Z	
				on $\overline{HALT}$	on $\overline{BGACK}$
Address bus	A1 – A23	Output	High	Yes	Yes
Data bus	D0 – D15	Input/output	High	Yes	Yes
Address strobe	$\overline{AS}$	Output	Low	No	Yes
Read/write	R/$\overline{W}$	Output	Read-High Write-Low	No No	Yes Yes
Upper and lower data strobes	$\overline{UDS}$, $\overline{LDS}$	Output	Low	No	Yes
Data transfer acknowledge	$\overline{DTACK}$	Input	Low	–	–
Bus request	$\overline{BR}$	Input	Low	–	–
Bus grant	$\overline{BG}$	Output	Low	No	No
Bus grant acknowledge	$\overline{BGACK}$	Input	Low	–	–
Interrupt priority level	$\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$	Input	Low	–	–
Bus error	$\overline{BERR}$	Input	Low	–	–
Reset	$\overline{RESET}$	Input/Output	Low	No*	No*
Halt	$\overline{HALT}$	Input/Output	Low	No*	No*
Enable	E	Output	High	No	No
Valid memory address	$\overline{VMA}$	Output	Low	No	Yes
Valid peripheral address	$\overline{VPA}$	Input	Low	–	–
Function code output	FC0, FC1, FC2	Output	High	No	Yes
Clock	CLK	Input	High	–	–
Power input	V_{CC}	Input	–	–	–
Ground	GND	Input	–	–	–

*Open drain

Bus Operation

The following paragraphs explain control signal and bus operation during data transfer operations, bus arbitration, bus error and halt conditions, and reset operation.

Data Transfer Operations

Transfer of data between devices involves the following signals:

1. address bus A1 through A23,
2. data bus D0 through D15, and
3. control signals.

The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cycles, the bus master assumes responsibility for deskewing all signals it issues at both the start and end of a cycle. In addition, the bus master is responsible for deskewing the acknowledge and data signals from the slave device.

The following paragraphs explain the read, write, and read-modify-write cycles. The indivisible read-modify-write cycle is the method used by the SCN68010 for interlocked multiprocessor communications.

Read Cycle — During a read cycle, the processor receives data from the memory or a peripheral device. The processor reads bytes of data in all cases. If the instruction specifies a word (or long word) operation, the processor reads both upper and lower bytes simultaneously by asserting both upper and lower data strobes. When the instruction specifies byte operation, the processor uses an internal A0 bit to determine which byte to read and then issues the data strobe required for that byte. For byte operations, when the A0 bit equals zero, the upper data strobe is issued. When the A0 bit equals one, the lower data strobe is issued. When the data is received, the processor correctly positions it internally. If $\overline{DTACK}$, $\overline{BERR}$, or $\overline{VPA}$ is not asserted for the required set-up time before the falling edge of S4, a wait cycle will be inserted in the bus cycle and $\overline{DTACK}$ will be sampled again on the falling edge of each wait cycle. The SCN68010 will continue to insert wait cycles until $\overline{DTACK}$, $\overline{BERR}$, or $\overline{VPA}$ is recognized.

A word read cycle flowchart is given in figure 9. A byte read cycle flowchart is given in figure 10. Read cycle timing is given in figure 11. Figure 12 details word and byte read cycle operations.

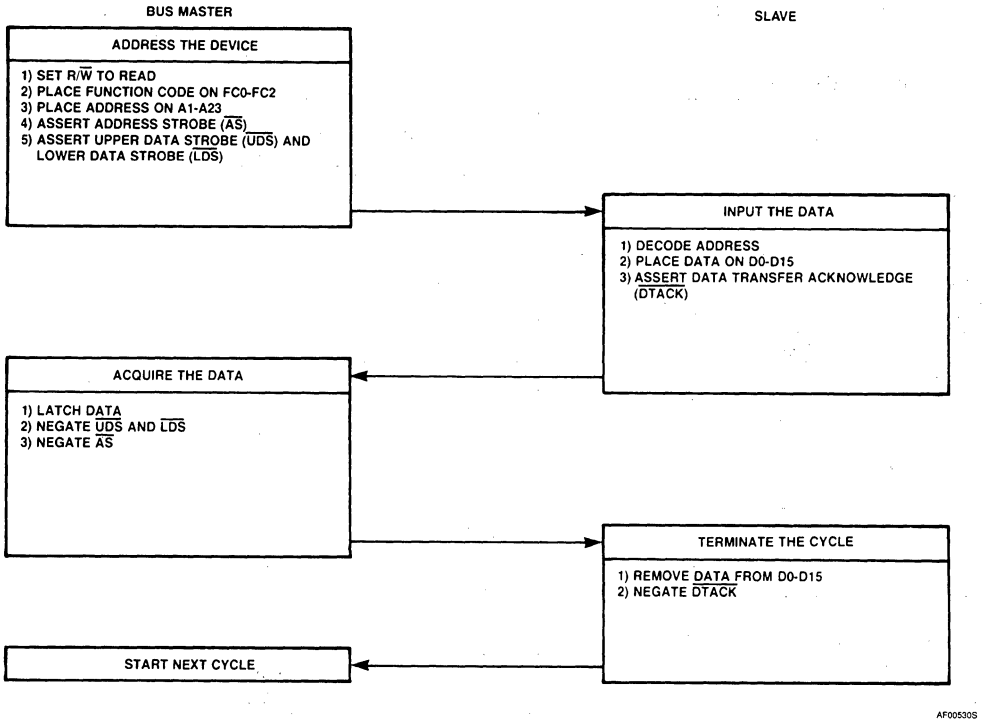


Figure 9. Word Read Cycle Flowchart

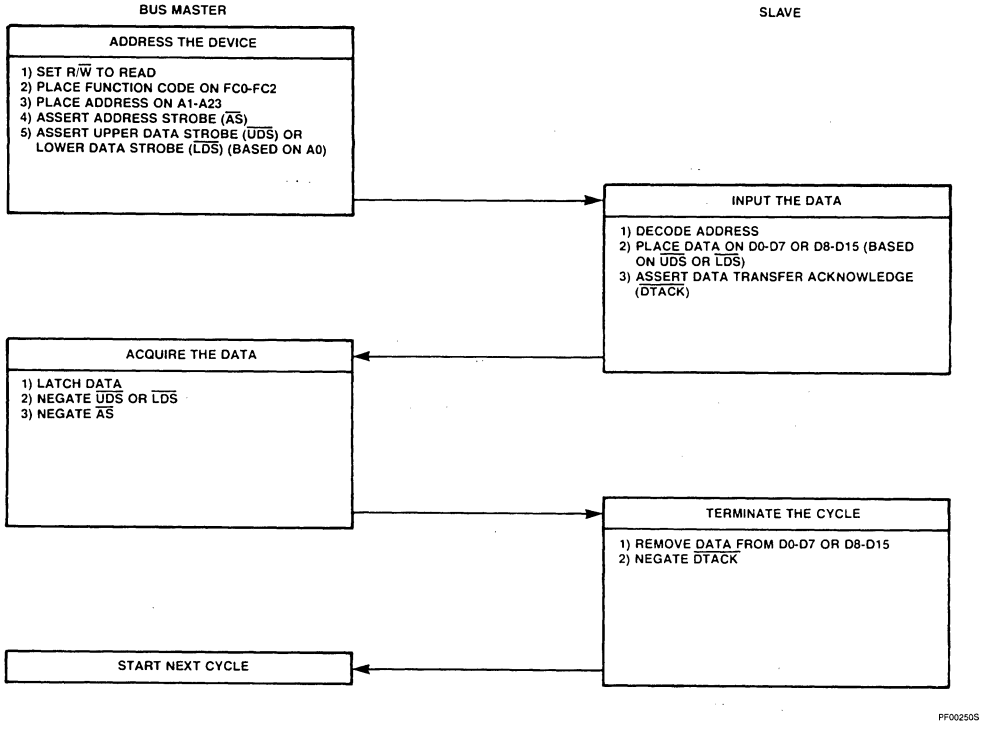


Figure 10. Byte Read Cycle Flowchart

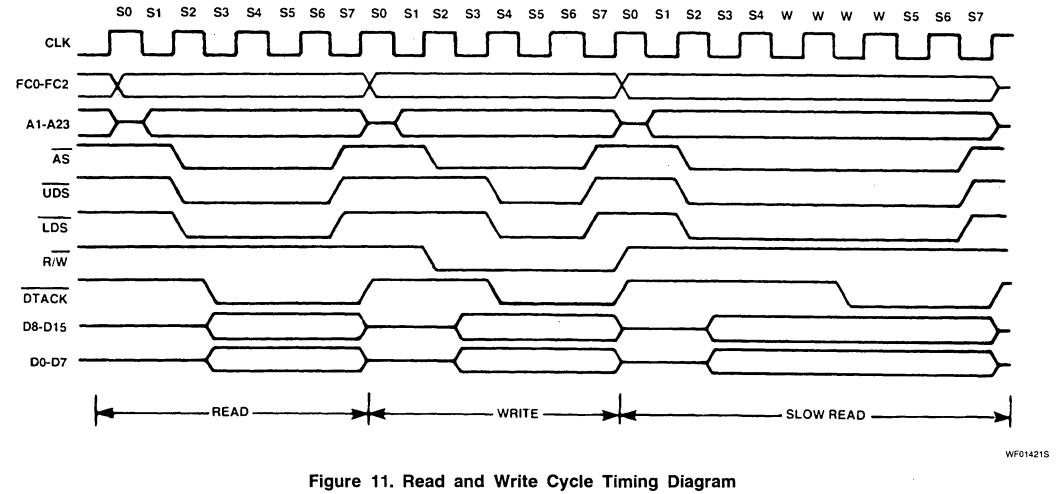


Figure 11. Read and Write Cycle Timing Diagram

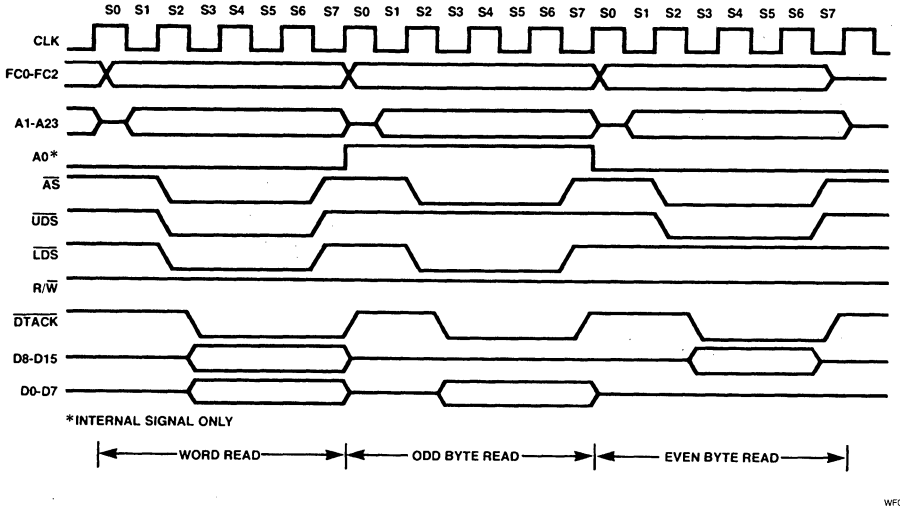


Figure 12. Word and Byte Read Cycle Timing Diagram

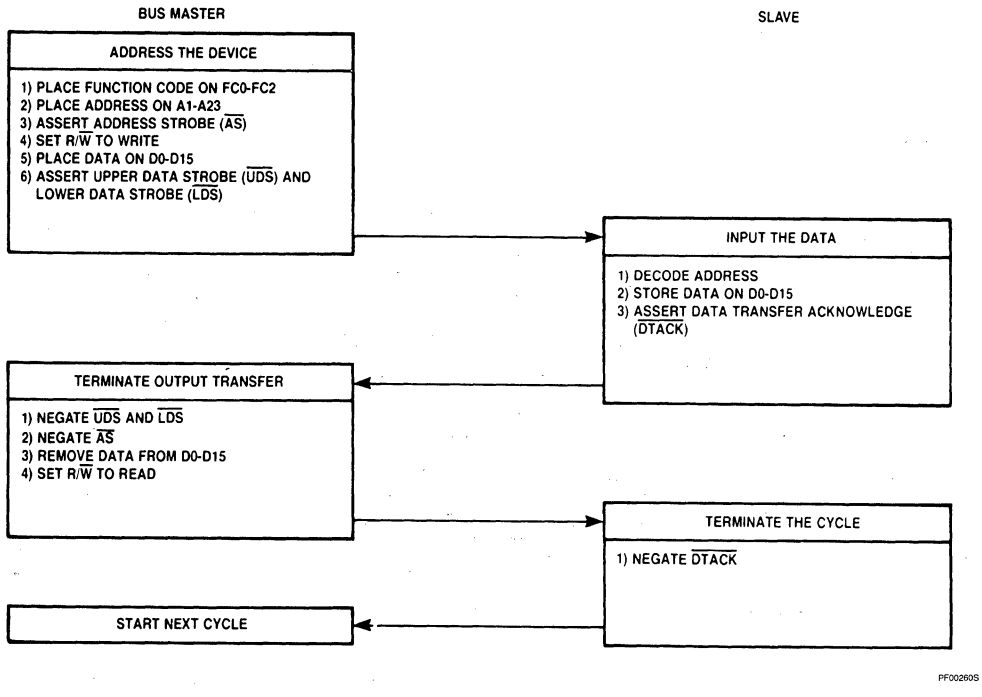


Figure 13. Word Write Cycle Flowchart

Write Cycle — During a write cycle, the processor sends data to either the memory or a peripheral device. The processor writes bytes of data in all cases. If the instruction specifies a word operation, the processor writes both bytes. When the instruction specifies a byte operation, the processor uses an internal A0 bit to determine which byte to write and then issues the data strobe required for that byte. For byte operations, when the A0 bit equals zero, the upper data strobe is issued. When the A0 bit equals one, the lower data strobe is issued. A word write flowchart is given in figure 13. A byte write cycle flowchart is given in figure 14. Write cycle timing is given in Figure 11. Figure 15 details word and byte write cycle operation.

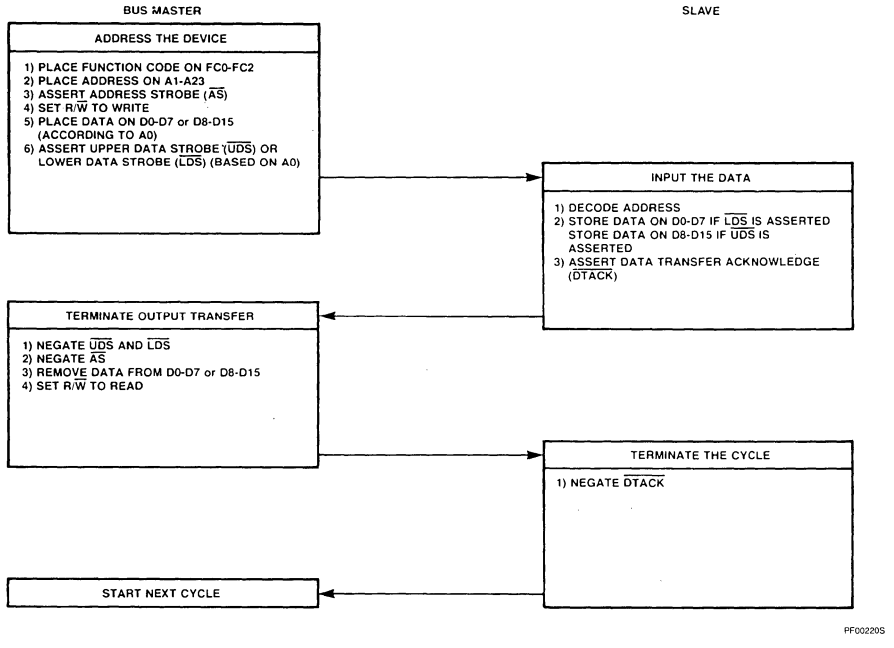


Figure 14. Byte Write Cycle Flowchart

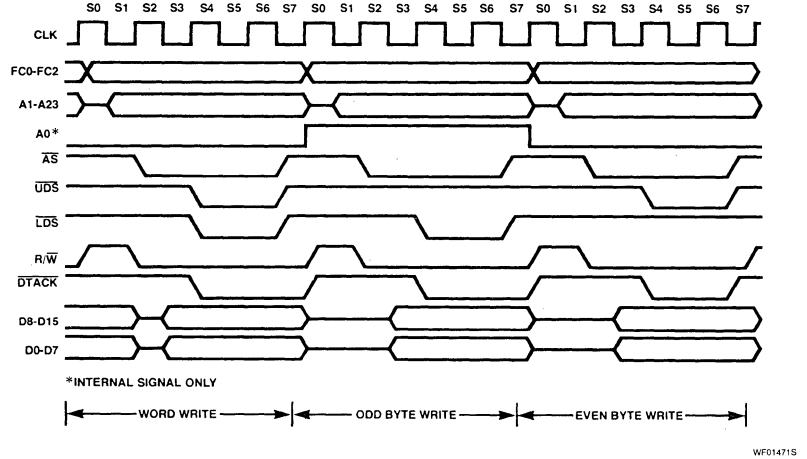


Figure 15. Word and Byte Write Cycle Timing Diagram

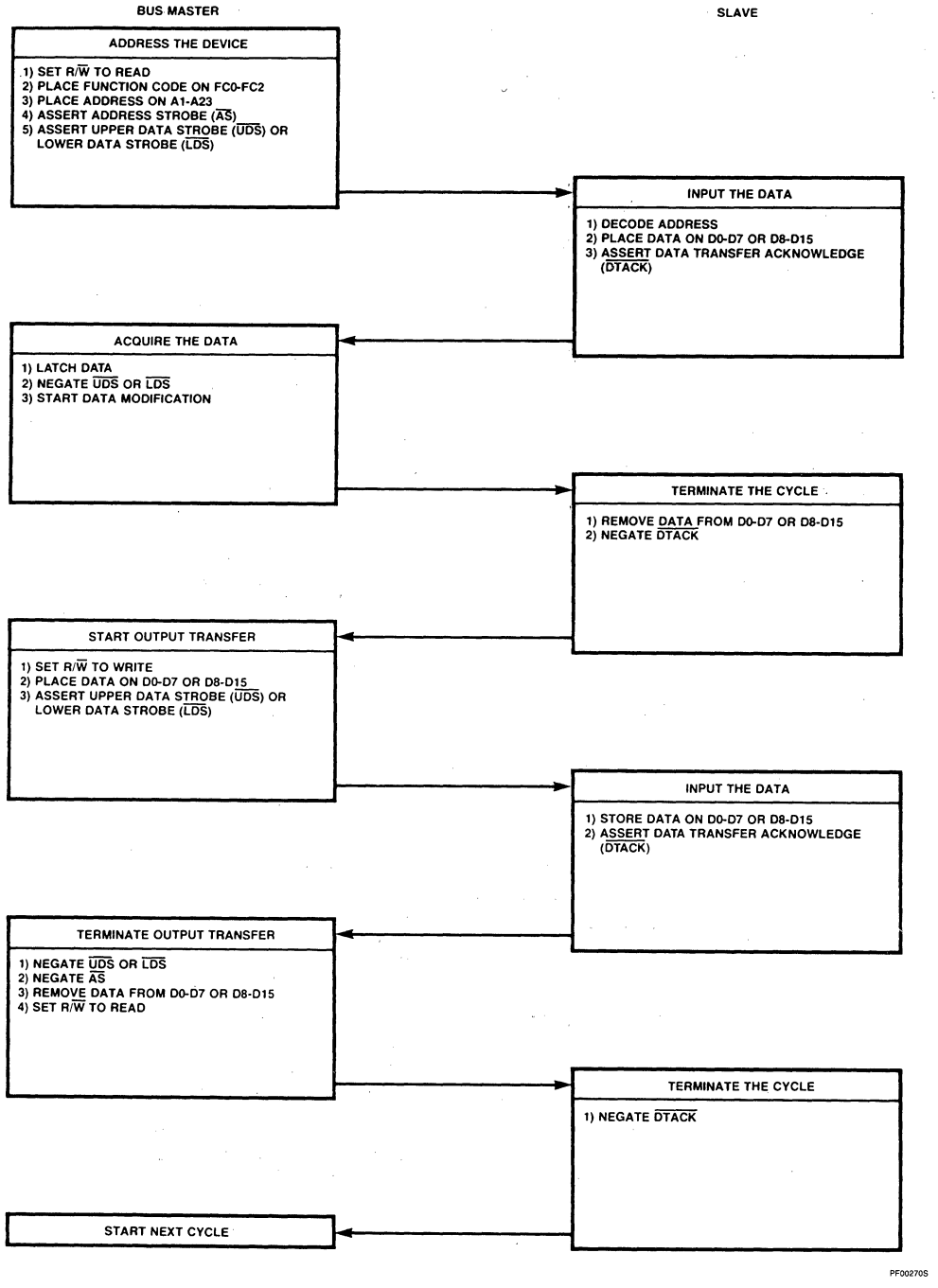


Figure 16. Read-Modify-Write Cycle Flowchart

Read-Modify-Write Cycle — The read-modify-write cycle performs a read, modifies the data in the arithmetic-logic unit, and writes the data back to the same address. In the SCN68010, this cycle is indivisible in that the address strobe is asserted throughout the entire cycle. The test and set (TAS) instruction uses this cycle to provide meaningful communication between processors in a multiple processor environment. This instruction is the only instruction that uses the read-modify-write cycle and since the test and set instruction only operates on bytes, all read-modify-write cycles are byte operations. A read-modify-write flowchart is given in figure 16 and a timing diagram is given in figure 17.

Wait cycles will be inserted between S4 and S5 on the read portion of the bus cycle and between S16 and S17 on the write portion of the cycle if $\overline{DTACK}$, $\overline{BERR}$, or $\overline{VPA}$ is not asserted for the required set-up time prior to the falling edge of S4 and S16 respectively.

CPU Space Cycle

During a CPU space cycle, the SCN68010 reads a peripheral device vector number or indicates a breakpoint instruction. If the cycle is to read a vector number it is referred to as an interrupt acknowledge cycle. A CPU space cycle is indicated when the function codes are all high. The address bus then defines what type of CPU space cycle is being executed. The SCN68010 defines two types of CPU space cycles, the interrupt acknowledge cycle, and the breakpoint cycle.

The interrupt acknowledge cycle on an S68000 family compatible processor is defined as a CPU space cycle with the most significant address lines high; on the SCN68010 this means that A4 – A23 will be high. The level of the interrupt being acknowledged is encoded on address lines A1 – A3. An interrupt acknowledge cycle is terminated in the same manner as a normal read cycle. The processor expects a peripheral device to respond to an interrupt acknowledge cycle with a vector number that will be used to transfer control to an interrupt handler routine. See **Interrupts** for further discussion of the interrupt acknowledge cycle.

The breakpoint read cycle is executed by the SCN68010 in response to a breakpoint illegal instruction. A breakpoint cycle on the SCN68010 is defined as a CPU space cycle with all of the address lines low. The processor does not accept or send any data during this cycle. The breakpoint cycle may be terminated by $\overline{DTACK}$, $\overline{BERR}$, or $\overline{VPA}$. See **Illegal and Unimplemented Instructions** for further discussion of breakpoints.

Since all members of the S68000 Family do not implement A20 – A31, these lines do not need to be decoded for CPU space functions. Only A16 – A19 are used to distinguish between different CPU space cycle types. The SCN68010 only uses the $0 and $F CPU space types as shown in figure 18; however, all unused encodings of bits A16 – A19 are reserved by Signetics for future extensions of the CPU space functions.

Bus Arbitration

Bus arbitration is a technique used by master-type devices to request, be granted, and acknowledge bus mastership. In its simplest form, it consists of the following:

1. asserting a bus mastership request,
2. receiving a grant that the bus is available at the end of the current cycle, and
3. acknowledging that mastership has been assumed.

Figure 19 is a flowchart showing the detail involved in a request from a single device. Figure 20 is a timing diagram for the same operation. This technique allows processing of bus requests during data transfer cycles.

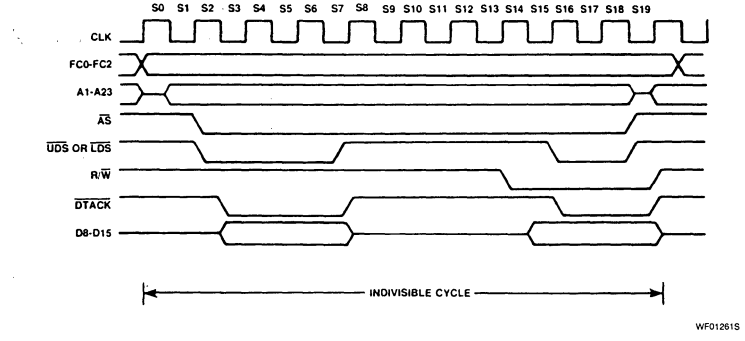


Figure 17. Read-Modify-Write Cycle Timing Diagram

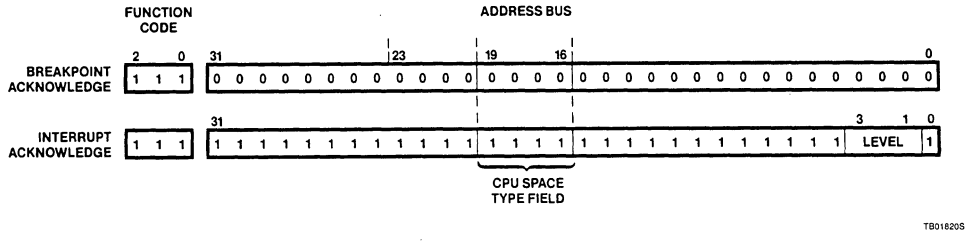


Figure 18. SCN68010 CPU-Space Address Encoding

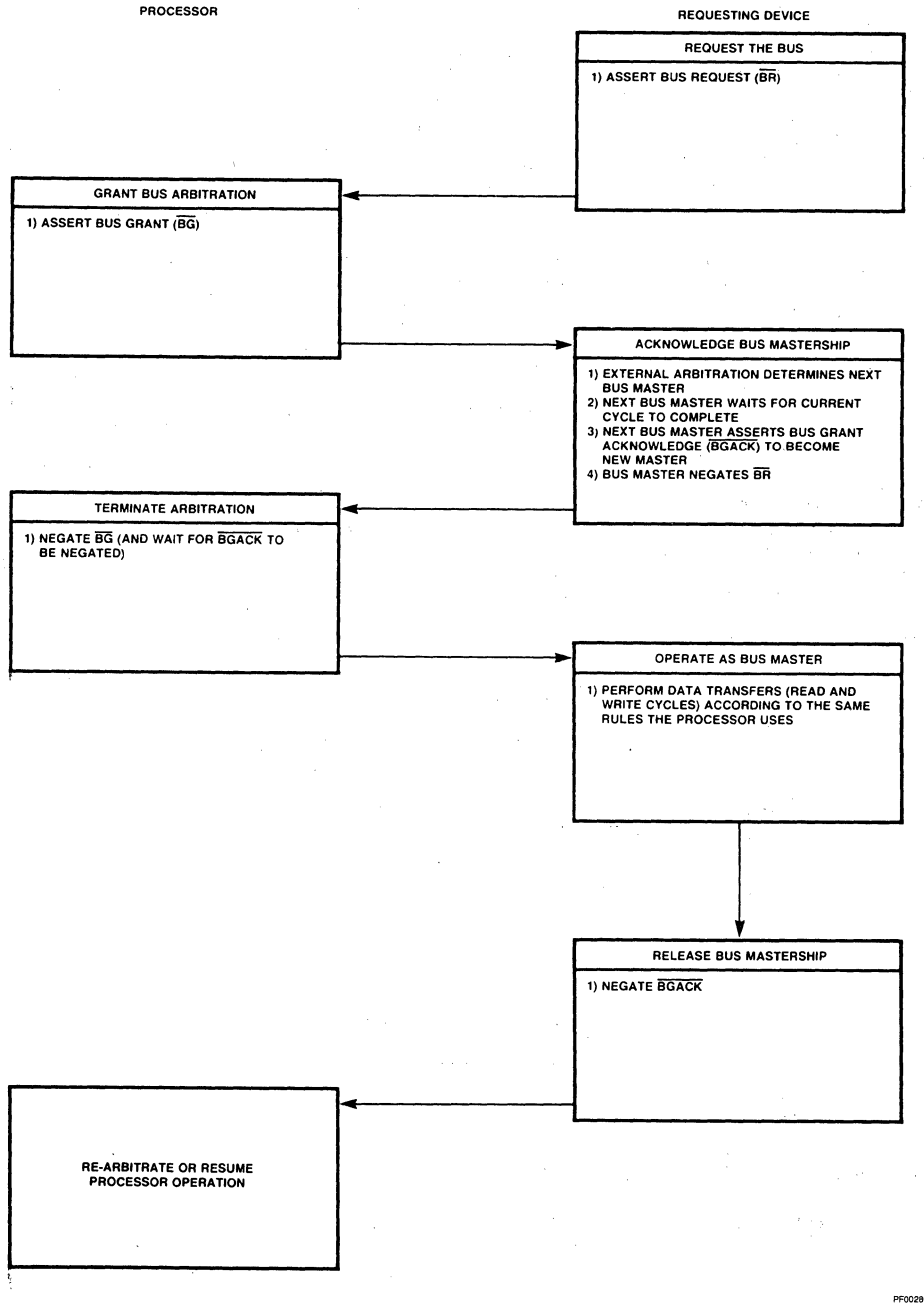


Figure 19. Bus Arbitration Cycle Flowchart

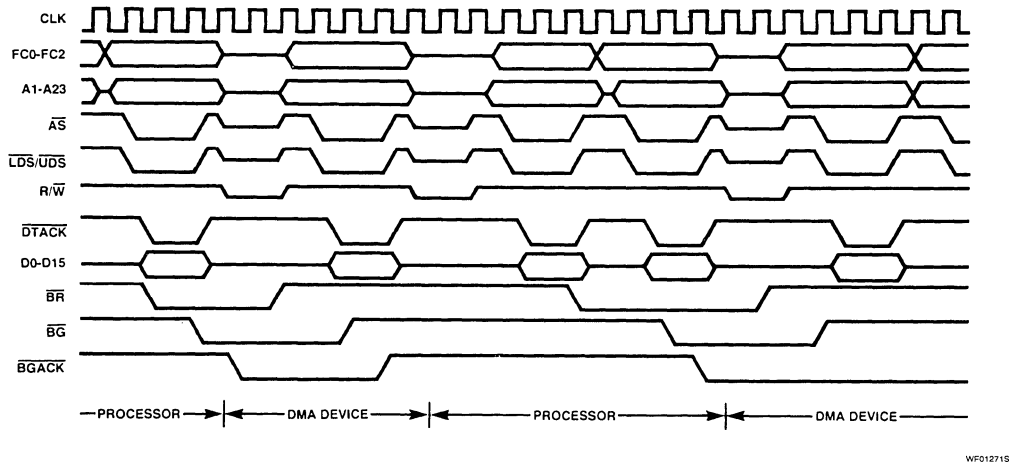


Figure 20. Bus Arbitration Cycle Timing Diagram

The timing diagram shows that the bus request is negated at the time that an acknowledge is asserted. This type of operation would be true for a system consisting of the processor and one device capable of bus mastership. In systems having a number of devices capable of bus mastership, the bus request line from each device is wire ORed to the processor. In this system, it is easy to see that there could be more than one bus request being made. The timing diagram shows that the bus grant signal is negated a few clock cycles after the transition of the acknowledge ($\overline{BGACK}$) signal.

However, if bus requests are still pending, the processor will assert another bus grant within a few clock cycles after it was negated. This additional assertion of bus grant allows external arbitration circuitry to select the next bus master before the current bus master has completed its requirements. The following paragraphs provide additional information about the three steps in the arbitration process.

Requesting the Bus — External devices capable of becoming bus masters request the bus by asserting the bus request ($\overline{BR}$) signal. This is a wire-ORed signal (although it need not be constructed from open-collector devices) that indicates to the processor that some external device requires control of the external bus. The processor is effectively at a lower bus priority level than the external device and will relinquish the bus after it has completed the last bus cycle it has started.

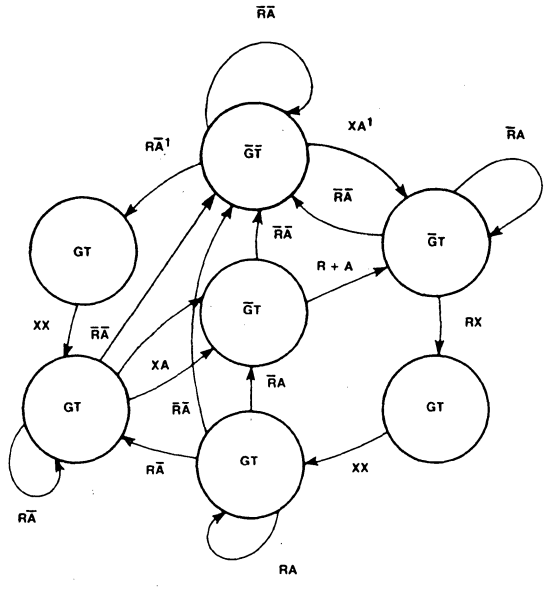
When no acknowledge is received before the bus request signal goes inactive, the processor will continue processing when it detects that the bus request is inactive. This allows ordinary processing to continue if the arbitration circuitry responded to noise inadvertently.

Receiving the Bus Grant — The processor asserts bus grant ($\overline{BG}$) as soon as possible. Normally this is immediately after internal synchronization. The only exception to this occurs when the processor has made an internal decision to execute the next bus cycle but has not progressed far enough into the cycle to have asserted the address strobe ($\overline{AS}$) signal. In this case, bus grant will be delayed until $\overline{AS}$ is asserted to indicate to external devices that a bus cycle is being executed.

The bus grant signal may be routed through a daisy-chained network or through a specific priority-encoded network. The processor is not affected by the external method of arbitration as long as the protocol is obeyed.

Acknowledgement of Mastership — Upon receiving a bus grant, the requesting device waits until address strobe, data transfer acknowledge, and bus grant acknowledge are negated before issuing its own $\overline{BGACK}$. The negation of the $\overline{AS}$ indicates that the previous master has completed its cycle; the negation of $\overline{BGACK}$ indicates that the previous master has released the bus. (While address strobe is asserted, no device is allowed to "break into" a cycle.) The negation of $\overline{DTACK}$ indicates the previous slave has terminated its connection to the previous master. Note that in some applications data transfer acknowledge might not enter into this function. General purpose devices would then be connected such that they were only dependent on address strobe. When bus grant acknowledge is issued, the device is a bus master until it negates bus grant acknowledge. Bus grant acknowledge should not be negated until after the bus cycle(s) is (are) completed. Bus mastership is terminated at the negation of bus grant acknowledge.

The bus request from the granted device should be negated after bus grant acknowledge is asserted. If a bus request is still pending, another bus grant will be asserted within a few clocks of the negation of the bus grant. Refer to **Bus Arbitration Control.** Note that the processor does not perform any external bus cycles before it re-asserts bus grant.



NOTES:
R = Bus request internal
A = Bus grant acknowledge internal
G = Bus grant
T = Three-state control to bus control logic[2]
X = Don't care
1. State machine will not change if the bus is S0 or S1. Refer to Bus Arbitration Control.
2. The address bus will be placed in the high-impedance state if T is asserted and $\overline{AS}$ is negated.

Figure 21. SCN68010 Bus Arbitration Unit State Diagram

Bus Arbitration Control

The bus arbitration control unit in the SCN68010 is implemented with a finite state machine. A state diagram of this machine is shown in figure 21. All asynchronous signals to the SCN68010 are synchronized before they are used internally. This synchronization is accomplished in a maximum of one cycle of the system clock, assuming that the asynchronous input set-up time (#47) has been met (see figure 22). The input signal is sampled on the falling edge of the clock and is valid internally after the next rising edge.

As shown in figure 21, input signals labeled R and A are internally synchronized on the bus request and bus grant acknowledge pins respectively. The bus grant output is labeled G and the internal three-state control signal T. If T is true, the address, data, and control buses are placed in a high-impedance state when $\overline{AS}$ is negated. All signals are shown in positive logic (active high) regardless of their true active voltage level.

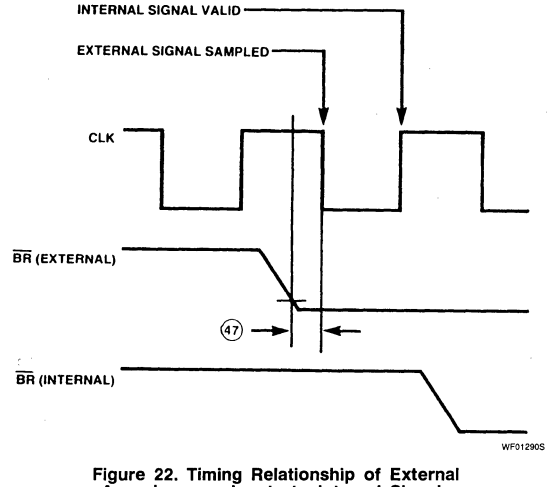


Figure 22. Timing Relationship of External Asynchronous Inputs to Internal Signals

State changes (valid outputs) occur on the next rising edge after the internal signal is valid.

A timing diagram of the bus arbitration sequence during a processor bus cycle is shown in figure 23. The bus arbitration sequence while the bus is inactive (i.e., executing internal operations such as a multiply instruction) is shown in figure 24.

If a bus request is made at a time when the MPU has already begun a bus cycle but $\overline{AS}$ has not been asserted (bus state S0), $\overline{BG}$ will not be asserted on the next rising edge. Instead, $\overline{BG}$ will be delayed until the second rising edge following its internal assertion. This sequence is shown in figure 25.

Bus Error and Halt Operation

In a bus architecture that requires a handshake from an external device, the possibility exists that the handshake might not occur. Since different systems will require a different maximum response time, a bus error input is provided. External circuitry must be used to determine the duration between address strobe and data transfer acknowledge before issuing a bus error signal. When a bus error or/and halt signal is received, the processor will initiate a bus error exception sequence or try to re-run the bus cycle.

In addition to a bus timeout indicator, the bus error input is used to indicate a page fault in a virtual memory system. When an external memory management unit detects an invalid access, a bus error is signaled to suspend execution of the current instruction.

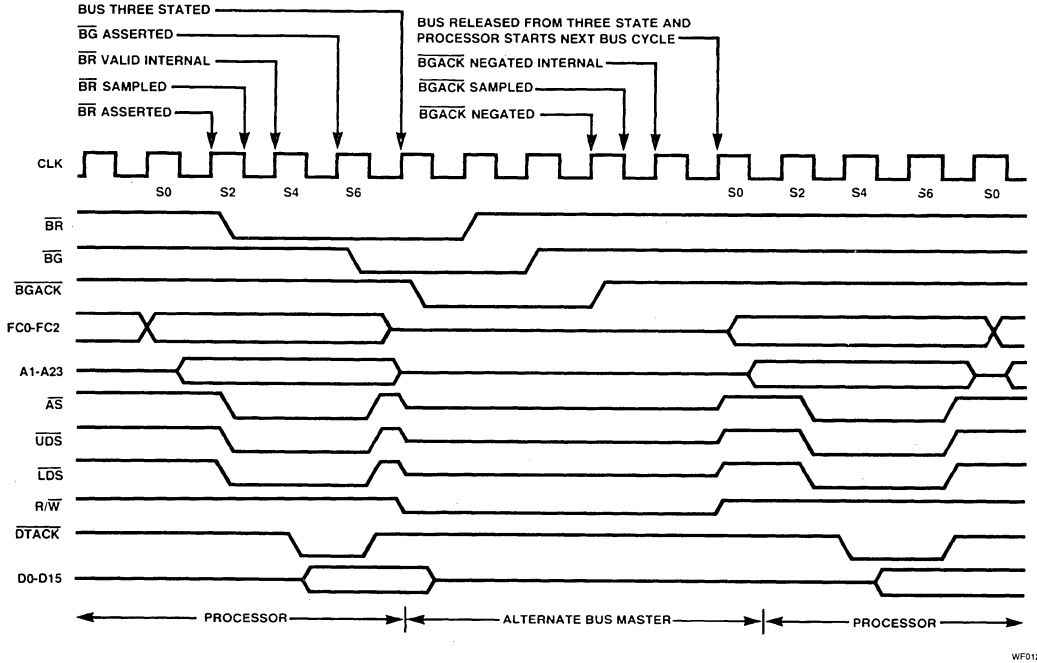


Figure 23. Bus Arbitration Timing Diagram — Processor Active

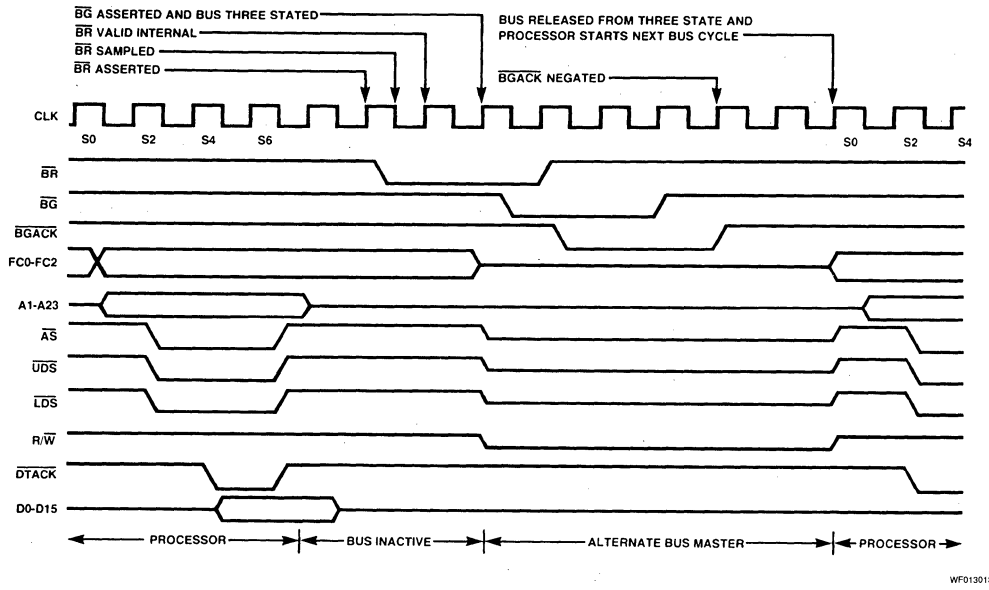


Figure 24. Bus Arbitration Timing Diagram — Bus Inactive

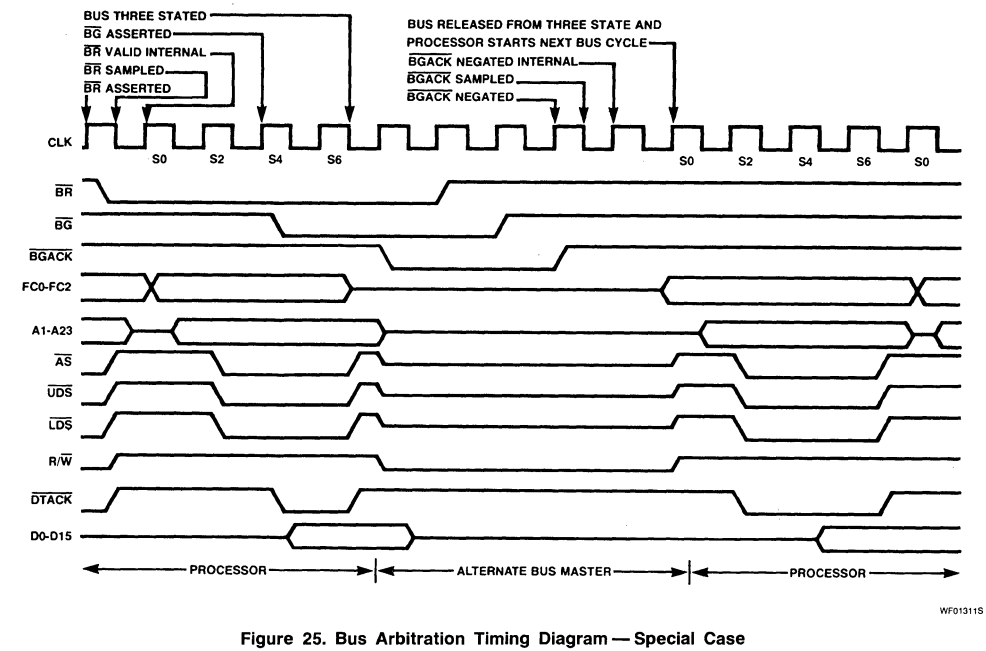


Figure 25. Bus Arbitration Timing Diagram — Special Case

Bus Error Operation — When the bus error signal is used to terminate a bus cycle, the SCN68010 will enter exception processing immediately following the bus cycle. The bus error signal is recognized in either of the following cases:

1. $\overline{DTACK}$ and $\overline{HALT}$ are negated and $\overline{BERR}$ is asserted.
2. $\overline{HALT}$ and $\overline{BERR}$ are negated and $\overline{DTACK}$ is asserted. $\overline{BERR}$ is then asserted within one clock cycle.

When the bus error condition is recognized, the current bus cycle will be terminated in S9 for a read cycle, a write cycle, or the read portion of a read-modify-write cycle and in S21 of the write portion of a read-modify-write cycle. As long as $\overline{BERR}$ remains asserted, the data and address buses will be in the high-impedance state. Figures 26 and 27 show the timing diagrams for both types of bus error signals.

2

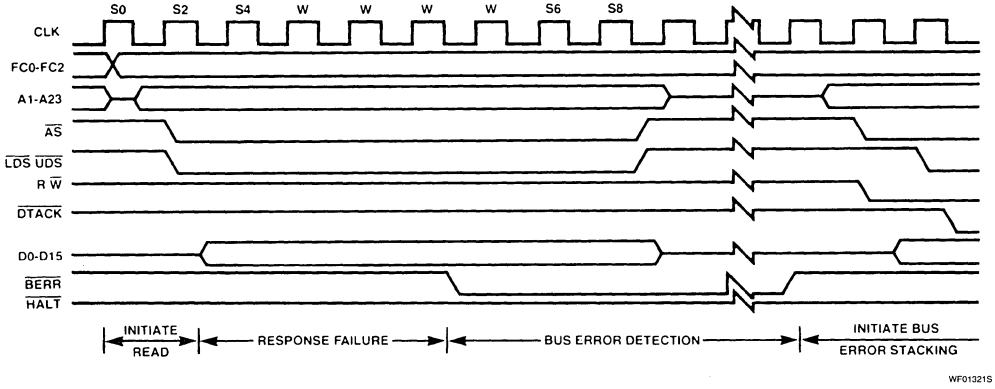


Figure 26. Bus Error Timing Diagram

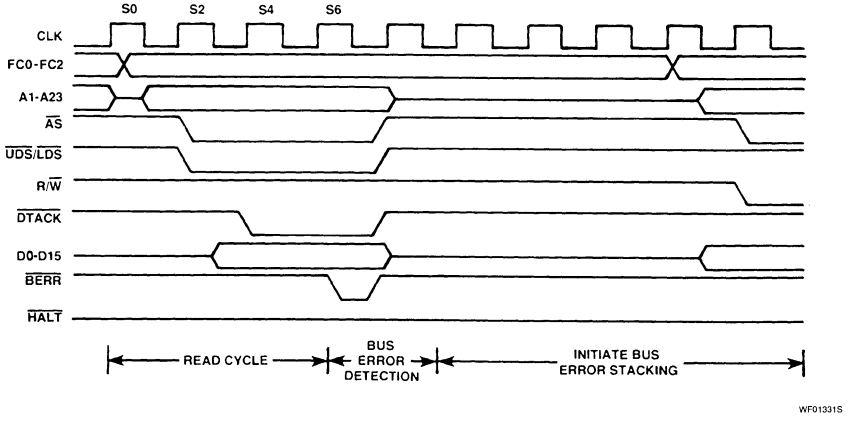


Figure 27. Delayed Bus Error Timing Diagram

After the aborted bus cycle is terminated and $\overline{BERR}$ is negated, the SCN68010 enters exception processing for the bus error exception. During the exception processing sequence, the following information is placed on the supervisor stack:

1. Status register
2. Program counter (two words, may be up to five words past the instruction being executed)
3. Frame format and vector offset
4. Internal register information, 22 words

Note that the first four words of information are identical to the information stacked by any other exception such as an interrupt or TRAP instruction. The additional information is used by the SCN68010 to continue the execution of the suspended instruction when it is reloaded by an RTE instruction. See **Bus Error** for further details.

After the SCN68010 has placed the above information on the stack, the bus error exception vector is read from vector table entry number two (offset $08) and placed in the program counter. The processor then resumes instruction execution.

NOTE
If a read-modify-write instruction is terminated with a bus error and later continued with an RTE instruction, the processor will re-run the entire cycle whether the bus error occurred on the read or the write portion of the cycle.

Re-Run Operation — When, during a bus cycle, the processor receives a bus error signal and the halt pin is being driven by an external device, the processor enters the re-run sequence. A delayed re-run signal may be used similarly to the delayed bus error signal described above. Figures 28 and 29 are timing diagrams for both methods of re-running the bus cycle.

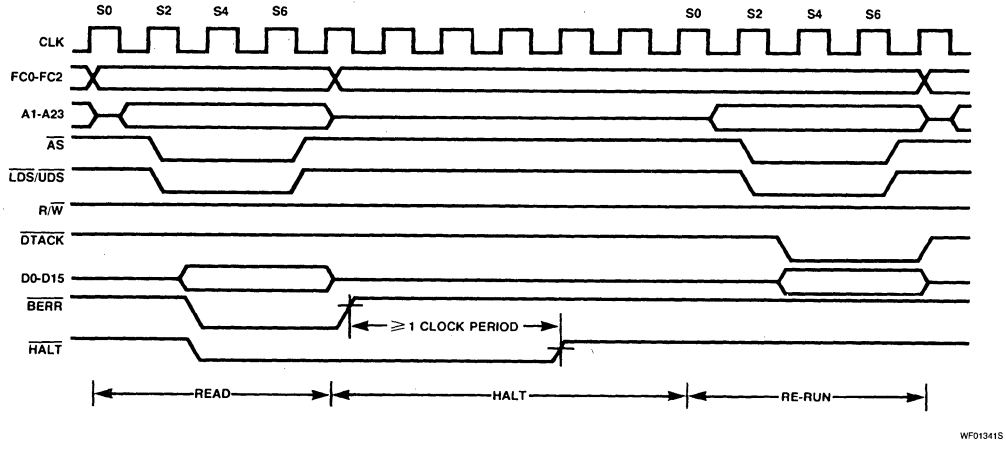


Figure 28. Re-Run Bus Cycle Timing Diagram

The processor terminates the bus cycle, then puts the address and data lines in the high-impedance state. The processor remains "halted", and will not run another bus cycle until the halt signal is removed by external logic. Then the processor will re-run the previous cycle using the same function codes, the same data (for a write operation), and the same address. The bus error signal should be removed at least one clock cycle before the halt signal is removed.

NOTE
The processor will not re-run a read-modify-write cycle. This restriction is made to guarantee that the entire cycle runs correctly and that the write operation of a test-and-set operation is performed without ever releasing $\overline{AS}$. If $\overline{BERR}$ and $\overline{HALT}$ are asserted during a read-modify-write bus cycle, a bus error operation results.

Halt Operation — The halt input signal to the SCN68010 performs a halt/run/single-step function in a similar fashion to the synchronous halt function. The halt and run modes are somewhat self explanatory in that when the halt signal is constantly active the processor "halts" (does nothing) and when the halt signal is constantly inactive the processor "runs" (does something).

This single-step mode is derived from correctly timed transitions on the halt signal input. It forces the processor to execute a single bus cycle by entering the run mode until the processor starts a bus cycle then changing to the halt mode. Thus, the single-step mode allows the user to proceed through (and therefore debug) processor operations one bus cycle at a time.

Figure 30 details the timing required for correct single-step operations. Some care must be exercised to avoid harmful interactions between the bus error signal and the halt pin when using the single-cycle mode as a debugging tool. This is also true of interactions between the halt and reset lines since these can reset the machine.

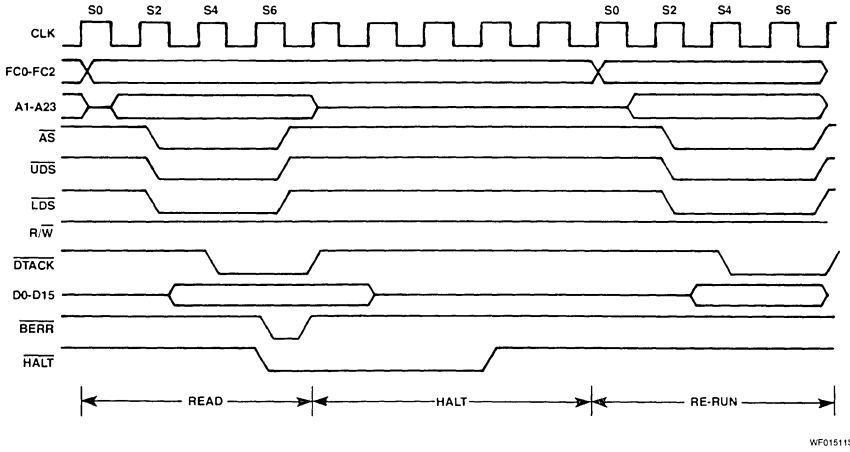


Figure 29. Delayed Re-Run Bus Cycle Timing Diagram

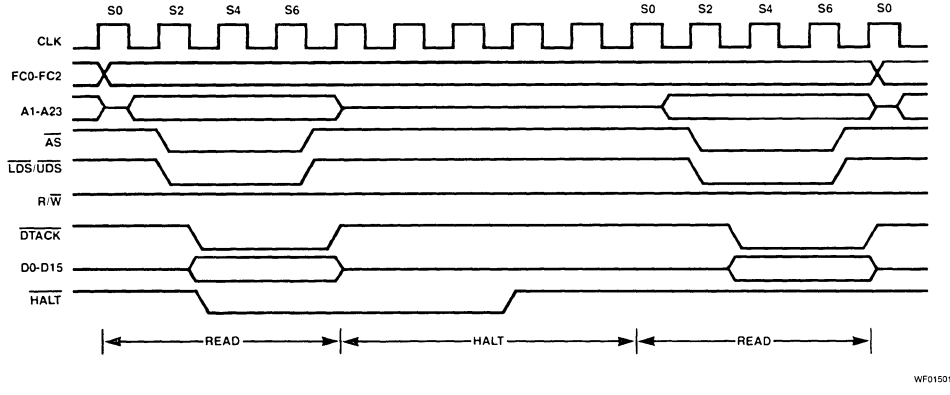


Figure 30. Halt Processor Timing Diagram

When the processor completes a bus cycle after recognizing that the halt signal is active, most three-state signals are put in the high-impedance state, these include:

1. address lines, and
2. data lines.

This is required for correct performance of the re-run bus cycle operation.

While the processor is honoring the halt request, bus arbitration performs as usual. That is, halting has no effect on bus arbitration. It is the bus arbitration function that removes the control signals from the bus.

The halt function and the hardware trace capability allow the hardware debugger to trace single bus cycles or single instructions at a time. These processor capabilities, along with a software debugging package, give total debugging flexibility.

Double Bus Faults — When a bus error exception occurs, the processor will attempt to stack several words containing information about the state of the machine. If a bus error exception occurs during the stacking operation, there have been two bus errors in a row. This is commonly referred to as a double bus fault. When a double bus fault occurs, the processor will halt and drive the $\overline{HALT}$ line low. Once a bus error exception has occurred, any bus error exception occurring before the execution of the next instruction constitutes a double bus fault.

Note that a bus cycle which is re-run does not constitute a bus error exception and does not contribute to a double bus fault. Note also that this means that as long as the external hardware requests it, the processor will continue to re-run the same bus cycle.

The bus error pin also has an effect on processor operation after the processor receives an external reset input. The processor reads the vector table after a reset to determine the address to start program execution. If a bus error occurs while reading the vector table (or at any time before the first instruction is executed), the processor reacts as if a double bus fault has occurred and it halts. Only an external reset will start a halted processor.

Reset Operation

The reset signal is a bidirectional signal that allows either the processor or an external device to reset the system. Figure 31 is a timing diagram for the reset operation. Both the halt and reset lines must be asserted to ensure total reset of the processor in all cases.

When the reset and halt lines are driven by an external device, it is recognized as an entire system reset, including the processor. The processor responds by reading the reset vector table entry (vector number zero, address $000000) and loads it into the supervisor stack pointer (SSP). Vector table entry number one at address $000004 is read next and loaded into the program counter. The processor initializes the status register to an interrupt level of seven and the vector base register to $00000000. No other registers are affected by the reset sequence.

When a reset instruction is executed, the processor drives the reset pin for 124 clock periods. In this case, the processor is trying to reset the rest of the system. Therefore, there is no effect on the internal state of the processor. All of the processor's internal registers and the status register are unaffected by the execution of a reset instruction. All external devices connected to the reset line should be reset at the completion of the reset instruction.

Asserting the $\overline{RESET}$ and $\overline{HALT}$ lines for ten clock cycles will cause a processor reset, except when V_{CC} is initially applied to the processor. In this case, an external reset must be applied for at least 100 milliseconds.

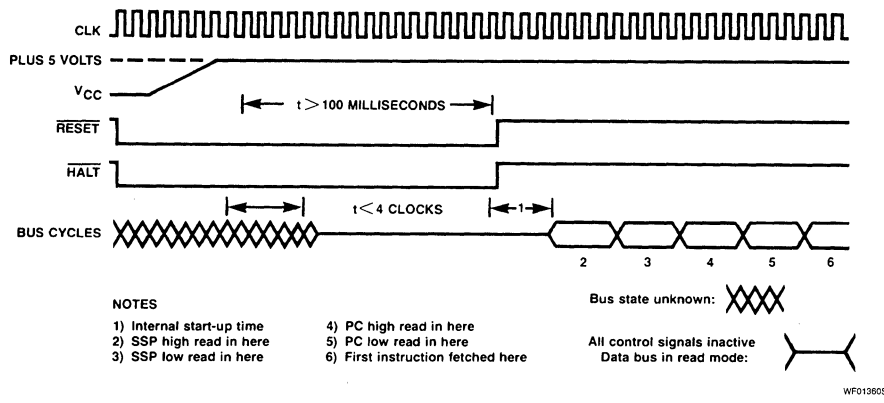


Figure 31. Reset Operation Timing Diagram

The Relationship of $\overline{DTACK}$, $\overline{BERR}$, and $\overline{HALT}$

In order to properly control termination of a bus cycle for a re-run or a bus error condition, $\overline{DTACK}$, $\overline{BERR}$, and $\overline{HALT}$ should be asserted and negated on the rising edge of the SCN68010 clock. This will assure that when two signals are asserted simultaneously, the required set-up time (#47) for both of them will be met during the same bus state. This, or some equivalent precaution, should be designed external to the SCN68010.

The preferred bus cycle terminations may be summarized as follows (case numbers refer to table 16):

Normal Termination:
$\overline{DTACK}$ is asserted, $\overline{BERR}$ and $\overline{HALT}$ remain negated (case 1).

Halt Termination:
$\overline{HALT}$ is asserted at same time, or before $\overline{DTACK}$ and $\overline{BERR}$ remains negated (case 2).

Bus Error Termination:
$\overline{BERR}$ is asserted in lieu of, at the same time, or before $\overline{DTACK}$ (case 3) or after $\overline{DTACK}$ (case 4) and $\overline{HALT}$ remains negated; $\overline{BERR}$ is negated at the same time or after $\overline{DTACK}$.

Re-Run Termination:
$\overline{HALT}$ and $\overline{BERR}$ are asserted in lieu of, at the same time, or before $\overline{DTACK}$ (case 5) or after $\overline{DTACK}$ (case 6); $\overline{BERR}$ is negated at the same time or after $\overline{DTACK}$, $\overline{HALT}$ must be held at least one cycle after $\overline{BERR}$.

Table 16 details the resulting bus cycle termination under various combinations of control signal sequences. The negation of these same control signals under several conditions is shown in table 17. ($\overline{DTACK}$ is assumed to be negated normally in all cases; for best results, both $\overline{DTACK}$ and $\overline{BERR}$ should be negated when address strobe is negated).

EXAMPLE A:
A system uses a watch-dog timer to terminate accesses to unpopulated address space. The timer asserts $\overline{BERR}$ after time out (case 3).

EXAMPLE B:
A system uses error detection on RAM contents. Designer may:

a) Delay $\overline{DTACK}$ until data verified, and return $\overline{BERR}$ and $\overline{HALT}$ simultaneously to re-run error cycle (case 5), or if valid, return $\overline{DTACK}$ (case 1).
b) Delay $\overline{DTACK}$ until data verified, and return $\overline{BERR}$ at same time as $\overline{DTACK}$ if data in error (case 3).
c) Return $\overline{DTACK}$ prior to data verification, as described in the next section. If data is invalid, $\overline{BERR}$ is asserted on next clock cycle (case 4).
d) Return $\overline{DTACK}$ prior to data verification, if data is invalid assert $\overline{BERR}$ and $\overline{HALT}$ on next clock cycle (case 6). The memory controller may then correct the RAM prior to or during the re-run.

Table 16. $\overline{DTACK}$, $\overline{BERR}$, AND $\overline{HALT}$ ASSERTION RESULTS

CASE NO.	CONTROL SIGNAL	ASSERTED ON RISING EDGE OF STATE		RESULT
		N	N + 2	
1	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	S NA X	Normal cycle terminate and continue.
2	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA A/S	S NA S	Normal cycle terminate and halt. Continue when $\overline{HALT}$ removed.
3	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	X A NA	X S NA	Terminate and take bus error trap.
4	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	X A NA	Terminate and take bus error trap.
5	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	X A A/S	X S S	Terminate and re-run when $\overline{HALT}$ removed.
6	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	X A A	Terminate and re-run when $\overline{HALT}$ removed.

NOTES:
N – the number of the current even bus state (e.g., S4, S6, etc.)
A – signal is asserted in this bus state
NA – signal is not asserted in this state
X – don't care
S – signal was asserted in previous state and remains asserted in this state

Table 17. $\overline{BERR}$ AND $\overline{HALT}$ NEGATION RESULTS

CONDITIONS OF TERMINATION IN TABLE 16	CONTROL SIGNAL	NEGATED ON RISING EDGE OF STATE		RESULTS – NEXT CYCLE
		N	N+2	
Bus Error	$\overline{BERR}$ $\overline{HALT}$	• or • • or •		Takes bus error trap.
Re-run	$\overline{BERR}$ $\overline{HALT}$	• or • •		Illegal sequence; usually traps to vector number 0.
Re-run	$\overline{BERR}$ $\overline{HALT}$	• 	 •	Re-runs the bus cycle.
Normal	$\overline{BERR}$ $\overline{HALT}$	• • or	 •	May lengthen next cycle.

• = Signal is negated in this bus state.

Asynchronous Versus Synchronous Operation

Asynchronous Operation

To achieve clock frequency independence at a system level, the SCN68010 can be used in an asynchronous manner. This entails using only the bus handshake lines ($\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, $\overline{DTACK}$, $\overline{BERR}$, $\overline{HALT}$ and $\overline{VPA}$) to control the data transfer. Using this method, $\overline{AS}$ signals the start of a bus cycle and the data strobes are used as a condition for valid data on a write cycle. The slave device (memory or peripheral) then responds by placing the requested data on the data bus for a read cycle or latching data on a write cycle and asserting the data transfer acknowledge signal ($\overline{DTACK}$) to terminate the bus cycle. If no slave responds or the access is invalid, external control logic asserts the $\overline{BERR}$, or $\overline{BERR}$

and $\overline{HALT}$, signal to abort or re-run the bus cycle.

The $\overline{DTACK}$ signal is allowed to be asserted before the data from a slave device is valid on a read cycle. The length of time that $\overline{DTACK}$ may precede data is given as parameter #31 (See **AC Electrical Characteristics** for # references) and it must be met in any asynchronous system to insure that valid data is latched into the processor. Notice that there is no maximum time specified from the assertion of $\overline{AS}$ to the assertion of $\overline{DTACK}$. This is because the MPU will insert wait cycles of one clock period each until $\overline{DTACK}$ is recognized.

The $\overline{BERR}$ signal is allowed to be asserted after the $\overline{DTACK}$ signal is asserted. $\overline{BERR}$ must be asserted within the time given as parameter #48 after $\overline{DTACK}$ is asserted in any asynchronous system to insure proper operation. If this maximum delay time is violated, the processor may exhibit erratic behavior.

Synchronous Operation

To allow for those systems which use the system clock as a signal to generate $\overline{DTACK}$ and other asynchronous inputs, the asynchronous input set-up time is given as parameter #47. If this set-up is met on an input, such as $\overline{DTACK}$, the processor is guaranteed to recognize that signal on the next falling edge of the system clock. However, the converse is not true — if the input signal does not meet the set-up time it is not guaranteed not to be recognized. In addition, if $\overline{DTACK}$ is recognized on a falling edge, valid data will be latched into the processor (on a read cycle) on the next falling edge provided that the data meets the set-up time given as parameter #27. Given this, parameter #31 may be ignored. Note that if $\overline{DTACK}$ is asserted, with the required set-up time, before the falling edge of S4, no wait states will be incurred and the bus cycle will run at its maximum speed of four clock periods.

In order to assure proper operation in a synchronous system when $\overline{BERR}$ is asserted after $\overline{DTACK}$, $\overline{BERR}$ must meet the set-up time parameter #27A prior to the falling edge of the clock one clock cycle after $\overline{DTACK}$ was recognized. This set-up time is critical to proper operation, and the SCN68010 may exhibit erratic behavior if it is violated.

NOTE

During an active bus cycle, $\overline{VPA}$ and $\overline{BERR}$ are sampled on every falling edge of the clock starting with S0. $\overline{DTACK}$ is sampled on every falling edge of the clock starting with S4 and data is latched on the falling edge of S6 during a read. The bus cycle will then be terminated in S7 except when $\overline{BERR}$ is asserted in the absence of $\overline{DTACK}$, in which case it will terminate one clock cycle later in S9.

PROCESSING STATES

This section describes the actions of the SCN68010 which are outside the normal processing associated with the execution of instructions. The functions of the bits in the supervisor portion of the status register are covered: the supervisor/user bit, the trace enable bit, and the processor interrupt priority mask. Finally, the sequence of memory references and actions taken by the processor on exception conditions are detailed.

The SCN68010 is always in one of three processing states: normal, exception, or halted. The normal processing state is that associated with instruction execution; the memory references are to fetch instructions and operands, and to store results. Two special cases of the normal state are the stopped state, which the processor enters when a STOP instruction is executed, and the loop mode, which the processor may enter when a DBcc instruction is executed. In the stopped state, no further memory references are made and in the loop mode only operand references are made.

The exception processing state is associated with interrupts, trap instructions, tracing and other exceptional conditions. The exception may be internally generated by an instruction or by an unusual condition arising during the execution of an instruction. Externally, exception processing can be forced by an interrupt, by a bus error, or by a reset. Exception processing is designed to provide an efficient context switch so that the processor may handle unusual conditions.

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts. Only an external reset can restart a halted processor. Note that a processor in the stopped state is not in the halted state, nor vice versa.

Privilege States

The processor operates in one of two states of privilege: the "supervisor" state or the "user" state. The privilege state determines which operations are legal, are used to choose between the supervisor stack pointer and the user stack pointer in instruction references, and may be used by an external memory management device to control and translate accesses.

The privilege state is a mechanism for providing security in a computer system. Programs should access only their own code and data areas, and ought to be restricted from accessing information which they do not need and must not modify.

The privilege mechanism provides security by allowing most programs to execute in user state. In this state, the accesses are controlled, and the effects on other parts of the system are limited. The operating system executes in the supervisor state, has access to all resources, and performs the overhead tasks for the user programs.

Supervisor State

The supervisor state is the higher state of privilege. For instruction execution, the supervisor state is determined by the S bit of the status register; if the S bit is asserted (high), the processor is in the supervisor state. All instructions can be executed in the supervisor state. The bus cycles generated by instructions executed in the supervisor state are classified as supervisor references. While the processor is in the supervisor privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly access the supervisor stack pointer.

All exception processing is done in the supervisor state, regardless of the previous setting of the S bit. The bus cycles generated during exception processing are classified as supervisor references. All stacking operations during exception processing use the supervisor stack pointer.

User State

The user state is the lower state of privilege. For instruction execution, the user state is determined by the S bit of the status register; if the S bit is negated (low), the processor is executing instructions in the user state.

Most instructions execute the same in user state as in the supervisor state. However, some instructions which have important system effects are made privileged. User programs are not permitted to execute the STOP instruction, or the RESET instruction. To ensure that a user program cannot enter the supervisor state except in a controlled manner, the instructions which modify the whole status register are privileged. To aid in debugging programs which are to be used as operating systems, the move from status register (MOVE from SR), move to/from user stack pointer (MOVE USP), move to/from control register (MOVEC), and move alternate address space (MOVES) instructions are also privileged.

The bus cycles generated by an instruction executed in the user state are classified as user state references. This allows an external memory management device to translate the address and to control access to protected portions of the address space. While the processor is in the user privilege state, those

Table 18. BUS CYCLE CLASSIFICATION

FUNCTION CODE OUTPUT			REFERENCE CLASS
FC2	FC1	FC0	
0	0	0	Unassigned, reserved*
0	0	1	User data
0	1	0	User program
0	1	1	Unassigned, reserved*
1	0	0	Unassigned, reserved*
1	0	1	Supervisor data
1	1	0	Supervisor program
1	1	1	CPU space

* Address space 3 is reserved for user definition, while 0 and 4 are reserved for future use by Signetics.

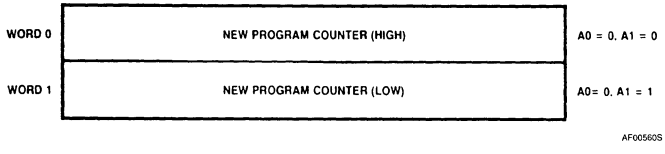


Figure 32. Format of Vector Table Entries

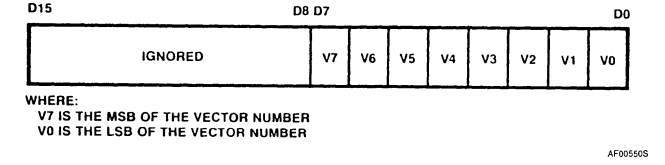


Figure 33. Vector Number Format

instructions which use either the system stack pointer implicitly or address register seven explicitly, access the user stack pointer.

Privilege State Changes

Once the processor is in the user state and executing instructions, only exception processing can change the privilege state. During exception processing, the previous setting of the S bit of the status register is saved and the S bit is asserted, putting the processor in the supervisor state. Therefore, when instruction execution resumes at the address specified to process the exception, the processor is in the supervisor privilege state.

Reference Classification

When the processor makes a reference, it classifies the kind of reference being made, using the encoding on the three function code output lines. This allows external translation of addresses, control of access, and differentiation of special processor state, such as interrupt acknowledge. Table 18 lists the classification of references.

Exception Processing

Before discussing the details of interrupts, traps, and tracing, a general description of exception processing is in order. The processing of an exception occurs in four steps, with variations for different exception causes. During the first step, a temporary copy of the status register is made and the status register is set for exception processing. In the second step the exception vector is determined and the third step is the saving of the current processor context. In the fourth step a new context is obtained and the processor resumes instruction processing.

Exception Vectors

Exception vectors are memory locations from which the processor fetches the address of a routine which will handle that exception. All exception vectors are two words in length (figure 32), except for the reset vector, which is four words. All exception vectors lie in the supervisor data space, except for the reset vector which is in the supervisor program space. A vector number is an 8-bit number which, when multiplied by four, gives the offset of an exception vector. Vector numbers are generated internally or externally, depending on the cause of the exception. In the case of interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (figure 33) to the processor on data bus lines D0 through D7. The processor translates the vector number into a full 32-bit offset which is added to the contents of the vector base register to generate the address used to fetch the vector, as shown in figure 34. The memory layout for exception vectors is given in table 19.

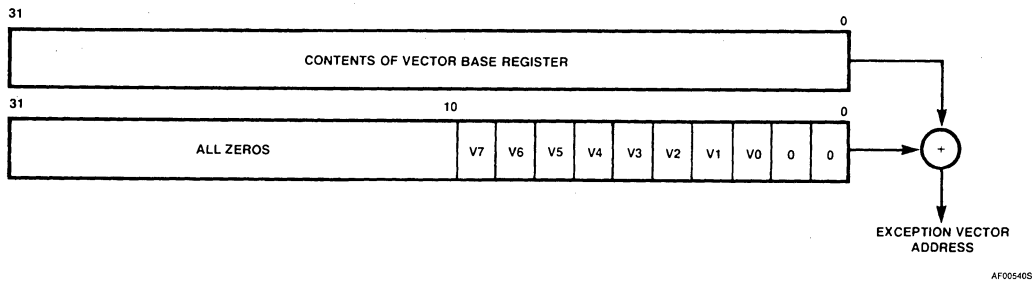


Figure 34. Exception Vector Address Calculation

As shown in table 19, the memory layout is 512 words long (1024 bytes). It starts at offset 0 and proceeds through offset 1023. This provides 255 unique vectors; some of these are reserved for TRAPs and other system functions. Of the 255, there are 192 reserved for user interrupt vectors. However, there is no protection on the first 63 entries, so externally generated interrupt vector numbers may reference any of the exception vectors at the discretion of the system designer.

Exception Stack Frame

Exception processing saves the most volatile portion of the current processor context on the top of the supervisor stack. This context is organized in a format called the exception stack frame. This information always includes the status register and program counter of the processor when the exception occurred. In order to support generic handlers, the processor also places the vector offset in the exception stack frame. The format code field allows the RTE (return from exception) instruction to identify what information is on the stack so that it may be properly restored. The general form of the exception stack frame is illustrated in figure 35. Table 20 lists the SCN68010 stack frame codes. Although some formats are peculiar to a particular S68000 family processor, the format 0000 is always legal, and indicates that just the first four words of the frame are present.

Table 20. SCN68010 FORMAT CODES

FORMAT CODE	STACKED INFORMATION
0000	SCN68010 short format (4 words)
1000	SCN68010 long format (29 words)
All others	Unassigned, reserved

Table 19. EXCEPTION VECTOR TABLE

VECTOR NUMBER(S)	OFFSET			ASSIGNMENT
	DEC	HEX	SPACE	
0	0	000	SP	Reset: initial SSP
—	4	004	SP	Reset: initial PC
2	8	008	SD	Bus error
3	12	00C	SD	Address error
4	16	010	SD	Illegal instruction
5	20	014	SD	Zero divide
6	24	018	SD	CHK instruction
7	28	01C	SD	TRAPV instruction
8	32	020	SD	Privilege violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 emulator
11	44	02C	SD	Line 1111 emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14	56	038	SD	Format error
15	60	03C	SD	Uninitialized interrupt vector
16 – 23*	64	04C	SD	(Unassigned, reserved)
	95	05F		—
24	96	060	SD	Spurious interrupt
25	100	064	SD	Level 1 interrupt autovector
26	104	068	SD	Level 2 interrupt autovector
27	108	06C	SD	Level 3 interrupt autovector
28	112	070	SD	Level 4 interrupt autovector
29	116	074	SD	Level 5 interrupt autovector
30	120	078	SD	Level 6 interrupt autovector
31	124	07C	SD	Level 7 interrupt autovector
32 – 47	128	080	SD	TRAP instruction vectors
	191	0BF		—
48 – 63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		—
64 – 255	256	100	SD	User interrupt vectors

*Vector numbers 12, 13, 16 through 23, and 48 through 63 are reserved for future enhancements by Signetics. No user peripheral devices should be assigned these numbers.

Kinds of Exceptions

Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts, bus error, and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset inputs are used for access control and processor restart. The internally generated exceptions come from instructions, or from address errors or tracing. The trap (TRAP), trap on overflow (TRAPV), check data register against upper bounds (CHK), and divide (DIV) instructions all can generate exceptions as part of their instruction execution. In addition, illegal instructions, word or long word fetches from odd addresses, and privilege violations cause exceptions. Tracing behaves like a very high-priority internally-generated interrupt after each instruction execution.

Exception Processing Sequence

Exception processing occurs in four identifiable steps. In the first step, an internal copy is made of the status register. After the copy is made, the S bit is asserted, putting the processor into the supervisor privilege state. Also, the T bit is negated which will allow the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor fetch classified as an interrupt acknowledge. For all other exceptions, internal logic provides the vector number. This vector number is then used to generate the offset of the exception vector and is added to the vector base register.

The third step is to save the current processor status, except for the reset exception. The exception stack frame is created at the top of the supervisor stack. The current program counter value, the saved copy of the status register, and the format/offset word are written into the stack frame. The program counter value stacked usually points to the next unexecuted instruction; however, for bus error and address error, the value stacked for the program counter is unpredictable, and may be incremented by up to five words from the address of the instruction which caused the error. Group 1 and 2 exceptions (see **Multiple Exceptions**) use a short format exception stack frame (format = 0000). Additional information defining the current context is stacked for the bus error and address error exceptions.

The last step is the same for all exceptions. The new program counter value is fetched from the exception vector table. The processor then resumes instruction execution. The instruction at the address given in the exception vector is fetched, and normal instruction decoding and execution is started.

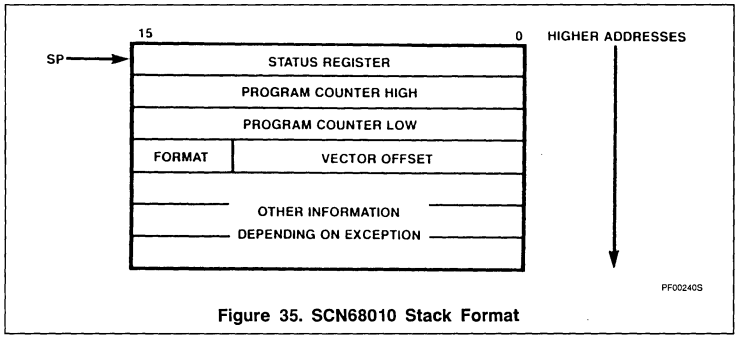


Figure 35. SCN68010 Stack Format

Multiple Exceptions

These paragraphs describe the processing which occurs when multiple exceptions arise simultaneously. Exceptions can be grouped according to their occurrence and priority. The group 0 exceptions are reset, bus error, and address error. These exceptions cause the instruction currently being executed to be aborted and the exception processing to commence within two clock cycles. The group 1 exceptions are trace and interrupt, as well as the privilege violations and illegal instructions. These exceptions allow the current instruction to execute to completion, but pre-empt the execution of the next instruction by forcing exception processing to occur (privilege violations and illegal instructions are detected when they are the next instruction to be executed). The group 2 exceptions occur as part of the normal processing of instructions. The TRAP, TRAPV, CHK, and zero divide exceptions are in this group. For these exceptions, the normal execution of an instruction may lead to exception processing.

Group 0 exceptions have highest priority, while group 2 exceptions have lowest priority. Within group 0, reset has highest priority, followed by address error and then bus error. Within group 1, trace has priority over external interrupts, which in turn takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, there is no priority relation within group 2.

The priority relation between two exceptions determines which is taken, or taken first, if the conditions for both arise simultaneously. Therefore, if a bus error occurs during a TRAP instruction, the bus error takes precedence, and the TRAP instruction processing is suspended. In another example, if an interrupt request occurs during the execution of an instruction while the T bit is asserted, the trace exception has priority, and is processed first. Before instruction processing resumes, however, the interrupt exception is also processed, and instruction processing commences finally in the interrupt handler routine. A summary of exception grouping and priority is given in table 21.

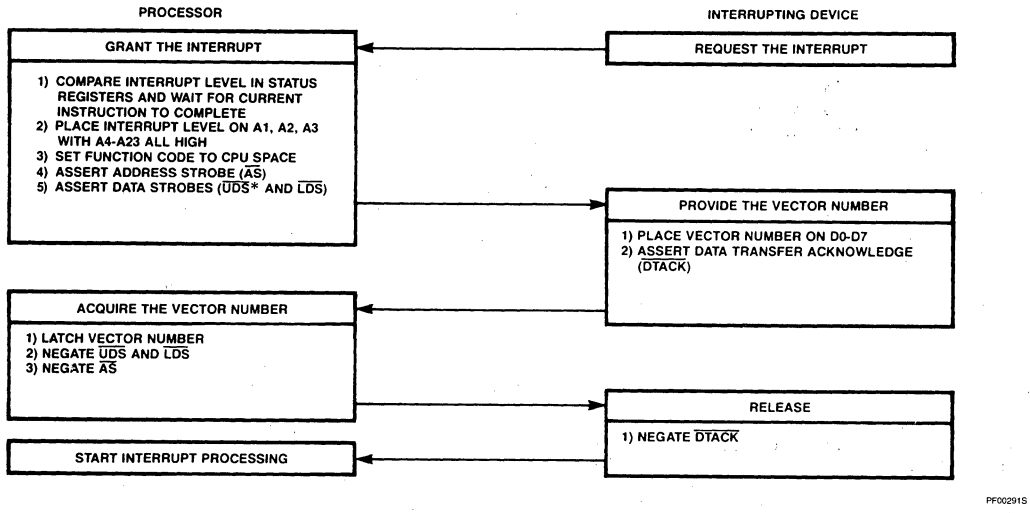
Exception Processing in Detail

Exceptions have a number of sources and each exception has processing which is peculiar to it. The following paragraphs detail the sources of exceptions, how each arises, and how each is processed.

Reset

The reset input provides the highest exception level. The processing of the reset signal is designed for system initiation, and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state, the trace state is forced off, and the processor interrupt priority mask is set to level seven. The vector base register is set to $00000000 and the vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents , in particular the supervisor stack pointer, neither the program counter nor the status register is saved. The address contained in the first two words of the reset exception vector is fetched as the initial supervisor stack pointer, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, instruction execution is started at the address in the program counter. The power-up/restart code should be pointed to by the initial program counter.

The reset instruction does not cause loading of the reset vector, but does assert the reset line to reset external devices. This allows the software to reset the system to a known state and then continue processing with the next instruction.



*Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D8 through D15 at this time.

Figure 36. Vector Acquisition Flowchart

Interrupts

Seven levels of interrupt priorities are provided. Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. Interrupt priority levels are numbered from one to seven, level seven being the highest priority. The status register contains a 3-bit mask which indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

An interrupt request is made to the processor by encoding the interrupt request level on the interrupt request lines; a zero indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts may cause exception processing to start at the end of an instruction depending on the current processor priority level. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception processing is postponed. (The recognition of level seven is slightly different, as explained in the following paragraph.)

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. A copy of the status register is saved, the privilege state is set to supervisor state, tracing is suppressed, and the processor priority level is set to the level of the interrupt being acknowledged. The processor fetches the vector number from the interrupting device, classifying the reference as an interrupt acknowledge and displaying the level number of the interrupt being acknowledged on the address bus. If external logic requests automatic vectoring, the processor internally generates a vector number which is determined by the interrupt level number. If external logic indicates a bus error, the interrupt is taken to be spurious, and the generated vector number references the spurious interrupt vector. The processor then proceeds with the usual exception processing, saving the format/offset word, program counter, and status register on the supervisor stack. The offset value in the format/offset word is the externally supplied or internally generated vector number multiplied by four. The format will be all zeros. The saved value of the program counter is the address of the instruction which would have been executed had the interrupt not been present. The content of the interrupt vector whose vector number was previously obtained is fetched and loaded into the program counter, and normal instruction execution commences in the interrupt handling routine. A flowchart for the interrupt acknowledge sequence is given in figure 36, a timing diagram is given in figure 37, and the interrupt processing sequence is shown in figure 38.

Table 21. EXCEPTION GROUPING AND PRIORITY

GROUP	EXCEPTION	PROCESSING
0	Reset address error bus error	Exception processing begins within two clock cycles
1	Trace interrupt illegal privilege	Exception processing begins before the next instruction
2	TRAP, TRAPV, CHK, zero divide format error	Exception processing is started by normal instruction execution

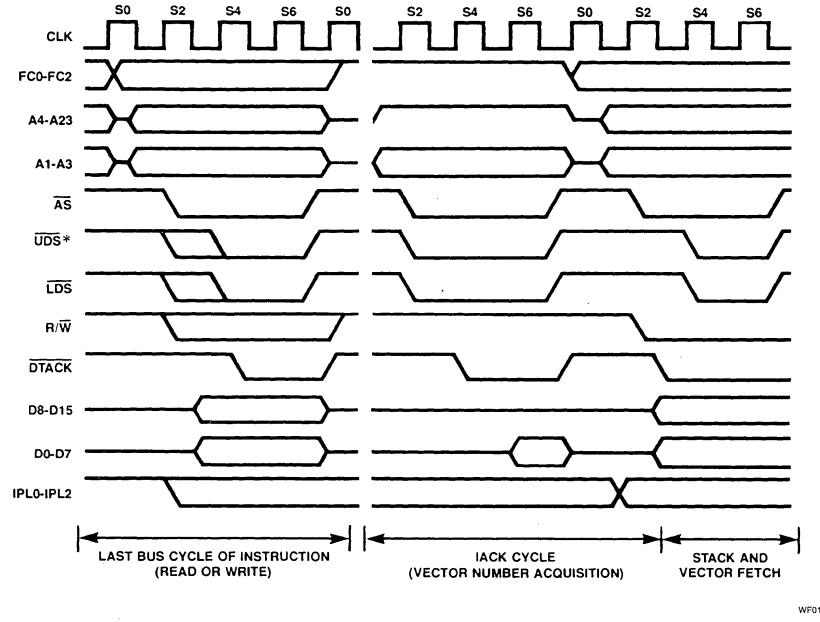


Figure 37. Interrupt Acknowledge Cycle Timing Diagram

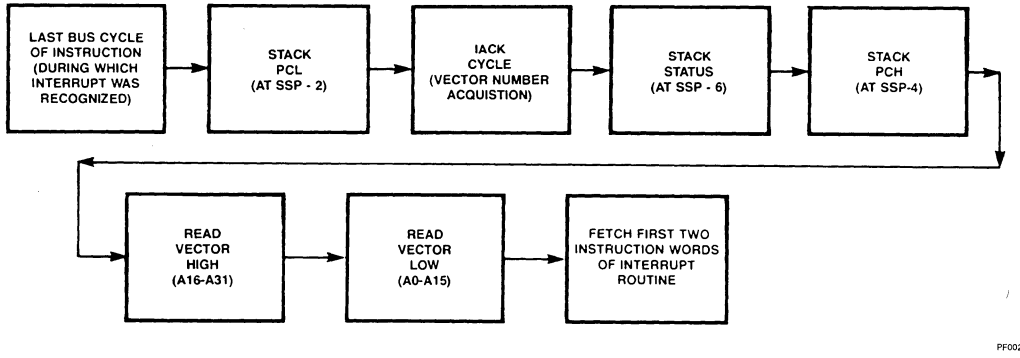


Figure 38. Interrupt Processing Sequence

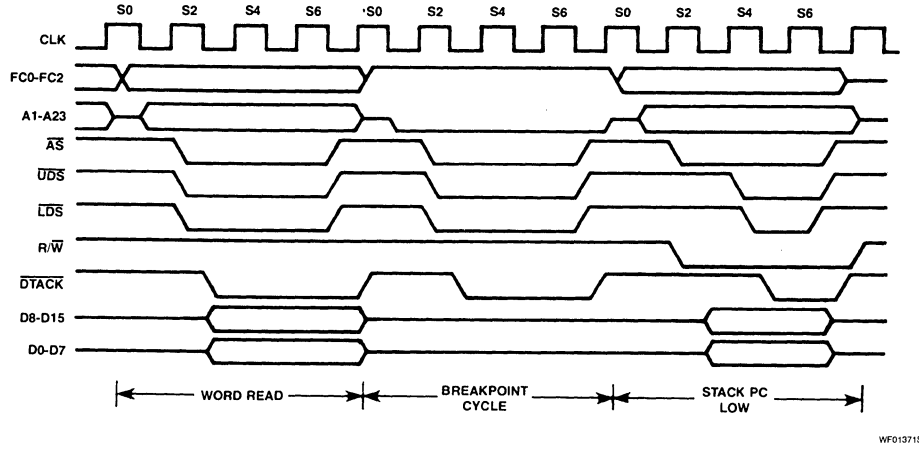


Figure 39. Breakpoint Cycle Timing Diagram

Priority level seven is a special case. Level seven interrupts cannot be inhibited by the interrupt priority mask, thus providing a "non-maskable interrupt" capability. An interrupt is generated each time the interrupt request level changes from some lower level to level seven. Note that a level seven interrupt may still be caused by the level comparison if the request level is a seven and the processor priority is set to a lower level by an instruction.

Uninitialized Interrupt

An interrupting device asserts $\overline{VPA}$, $\overline{BERR}$, or provides an interrupt vector number and asserts $\overline{DTACK}$ during an interrupt acknowledge cycle by the SCN68010. If the vector register has not been initialized, the responding S68000 family peripheral will provide vector number 15, the uninitialized interrupt vector. This provides a uniform way to recover from a programming error.

Spurious Interrupt

If during the interrupt acknowledge cycle no device responds by asserting $\overline{DTACK}$ or $\overline{VPA}$, $\overline{BERR}$ should be asserted to terminate the vector acquisition. The processor separates the processing of this error from bus error by forming a short format exception stack and fetching the spurious interrupt vector instead of the bus error vector. The processor then proceeds with the usual exception processing.

Instruction Traps

Traps are exceptions caused by instructions. They arise either from processor recognition of abnormal conditions during instruction execution, or from use of instructions whose normal behavior is trapping.

Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception and is useful for implementing supervisor calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a runtime error, which may be an arithmetic overflow or a subscript out of bounds.

The signed divide (DIVS) and unsigned divide (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

Illegal and Unimplemented Instructions

Illegal instruction is the term used to refer to any of the word bit patterns which are not the bit patterns of the first word of a legal SCN68010 instruction. During instruction execution, if such an instruction is fetched, an illegal instruction exception occurs. Signetics reserves the right to define instructions whose opcodes may be any of the illegal instructions. Three bit patterns will always force an illegal instruction trap on all S68000 family compatible microprocessors. They are: $4AFA, $4AFB, and $4AFC. Two of the patterns, $4AFA and $4AFB, are reserved for Signetics system products. The third pattern, $4AFC, is reserved for customer use.

In addition to the previously defined illegal instruction opcodes, the SCN68010 defines eight breakpoint illegal instructions with the bit patterns $4848 – $484F. These instructions cause the processor to enter illegal instruction exception processing as usual, but a breakpoint bus cycle is executed before the stacking operations are performed as shown in figure 39. The processor does not accept or send any data during this cycle. Whether the breakpoint cycle is terminated with a $\overline{DTACK}$, $\overline{BERR}$, or $\overline{VPA}$ signal, the processor will continue with the illegal instruction processing. The purpose of this cycle is to provide a software breakpoint that will signal external hardware when it is executed.

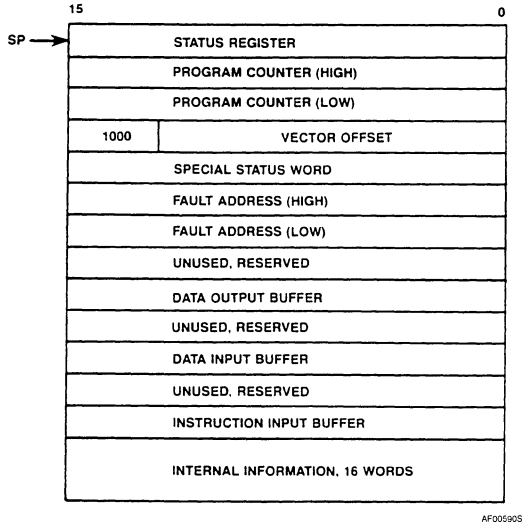


Figure 40. Exception Stack Order (Bus and Address Error)

Word patterns with bits 12 – 15 equaling 1010 or 1111 are distinguished as unimplemented instructions and separate exception vectors are given to these patterns to permit efficient emulation. This facility allows the operating system to detect program errors, or to emulate unimplemented instructions, such as the 68881 floating-point coprocessor instructions, in software.

Privilege Violations

In order to provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user state will cause an exception. The privileged instructions are:

AND immediate to SR	MOVE USP
EOR immediate to SR	OR immediate to SR
MOVE to SR	RESET
MOVE from SR	RTE
MOVEC	STOP
MOVES	

Tracing

To aid in program development, the SCN68010 includes a facility to allow instruction-by-instruction tracing. In the trace state, after each instruction is executed an exception is forced, allowing a debugging program to monitor the execution of the program under test.

The trace facility uses the T bit in the supervisor portion of the status register. If the T bit is negated (off), tracing is disabled, and instruction execution proceeds from instruction to instruction as normal. If the T bit is asserted (on) at the beginning of the execution of an instruction, a trace exception will be generated as the execution of that instruction is completed. If the instruction is not executed, either because an interrupt is taken, or the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus error, or address error exception. If the instruction is indeed executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. If, during the execution of the instruction an exception is forced by that instruction, the forced exception is processed before the trace exception.

As an extreme illustration of the above rules, consider the arrival of an interrupt during the execution of a TRAP instruction while tracing is enabled. First the trap exception is processed, then the trace exception, and finally the interrupt exception. Instruction execution resumes in the interrupt handler routine.

Bus Error

Bus error exceptions occur when external logic terminates a bus cycle with a bus error signal. Whether the processor was doing instruction or exception processing, that processing is terminated, and the processor immediately begins exception processing. However, if a bus error occurs during exception processing for a bus error, address error, or reset, the processor detects a double bus fault and halts. When exception processing is completed, instruction execution continues at the address contained in exception vector table entry two, at offset $008.

Exception processing for a bus error follows a slightly different sequence than the sequence for group 1 and 2 exceptions. In addition to the four steps executed during exception processing for all other exceptions, 22 words of additional information are placed on the stack. This additional information describes the internal state of the processor at the time of the bus error and is reloaded by the RTE instruction to continue the instruction that caused the error. Figure 40 shows the order of the stacked information.

The value of the saved program counter does not necessarily point to the instruction that was executing when the bus error occurred, but may be advanced by up to five words. This is due to the prefetch mechanism on the SCN68010 that always fetches a new instruction word as each previously fetched instruction word is used (see **Instruction Prefetch**). However, enough information is placed on the stack for the bus error exception handler routine to determine why the bus fault occurred. This additional information includes the address that was being accessed, the function codes for the access, whether it was a read or a write, and what internal register was included in the transfer. The fault address can be used by an operating system to determine what virtual memory location is needed so that the requested data can be brought into physical memory. The RTE instruction is then used to reload the processor's internal state at the time of the fault, the faulted bus cycle will then be re-run and the suspended instruction completed. If the faulted bus cycle was a read-modify-write, the entire cycle will be re-run whether the fault occurred during the read or the write operation.

An alternate method of handling a bus error is to complete the faulted access in software. In order to use this method, use of the special status word, the instruction input buffer, the data input buffer, and the data output buffer image is required. The format of the special status word is shown in figure 41. If the bus cycle was a write, the data output buffer image should be written to the fault address location using the function code contained in the special status word. If the cycle was a read, the data at the fault address location should be written to the images of the data input buffer, instruction input buffer, or both according to the DF and IF bits.[1] In addition,

[1]If the faulted access was a byte operation, the data should be moved from or to the least-significant byte of the data output or input buffer images unless the HB bit is set. This condition will only occur if a MOVEP instruction caused the fault during the transfer of bits 8 – 15 of a word or long word or bits 24 – 31 of a long word.

for read-modify-write cycles, the status register image must be properly set to reflect the read data if the fault occurred during the read portion of the cycle and the write operation (i.e., setting the most significant bit of the memory location) must also be performed. This is because the entire read-modify-write cycle is assumed to have been completed by software. Once the cycle has been completed by software, the RR bit in the special status word is set to indicate to the processor that it should not re-run the cycle when the RTE instruction is executed. If the re-run flag is set when an RTE instruction executes, the SCN68010 still reads all of the information from the stack.

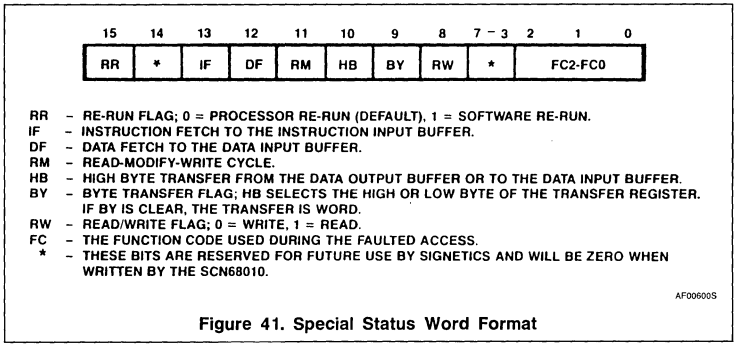


Figure 41. Special Status Word Format

Address Error

Address error exceptions occur when the processor attempts to access a word or long word operand or an instruction at an odd address. The effect is much like an internally generated bus error, so that the bus cycle is aborted, and the processor begins exception processing. After exception processing commences, the sequence is the same as that for bus error including the information that is stacked, except that the vector offset refers to the address error exception vector. If an address error occurs during exception processing for a bus error, address error, or reset, the processor detects a double bus fault and halts.

As shown in figure 42, an address error will execute a short bus cycle followed by exception processing. This short bus cycle is similar to a normal read or write cycle, except that the data strobes are not asserted and no external signals are used to terminate the cycle. During an address error bus cycle, AS is asserted to indicate that the SCN68010 will drive the address bus (thus allowing for proper operation in a multiple bus master system). Note that data strobes are not asserted allowing for address error detection and memory protection.

Since the address error exception stacks the same information that is stacked by a bus error exception, it is possible to use the RTE instruction to continue execution of the suspended instruction. However, if the software re-run flag is not set, the fault address will be used when the cycle is re-run and another address error exception will occur. Therefore, the user must be certain that the proper corrections have been made to the stack image and user registers before attempting to continue the instruction. With proper software handling, the address error exception handler could emulate word or long word accesses to odd addresses if desired.

Return From Exception

In addition to returning from any exception handler routine, the RTE instruction is used to resume the execution of a suspended instruction by restoring all of the temporary register and control information stored during a bus error and returning to the normal processing state. For the RTE instruction to execute properly, the stack must contain valid and accessible data. The RTE instruction checks for data validity in two ways; first, by checking the format/offset word for a valid stack format code, and second, if the format code indicates the long stack format, the long stack data is checked for validity as it is loaded into the processor. In addition, the data is checked for accessibility when the processor starts reading the long data. Because of these checks, the RTE instruction executes as follows:

1. Determine the stack format. This step is the same for any stack format and consists of reading the status register, program counter, and format/offset word. If the format code indicates a short stack format, execution continues at the new program counter address. If the format code is not one of the SCN68010 defined stack format codes, exception processing starts for a format error.
2. Determine data validity. For a long stack format, the SCN68010 will begin to read the remaining stack data, checking for validity of the data. The only word checked for validity is the first of the 16 internal information words (SP + 26)

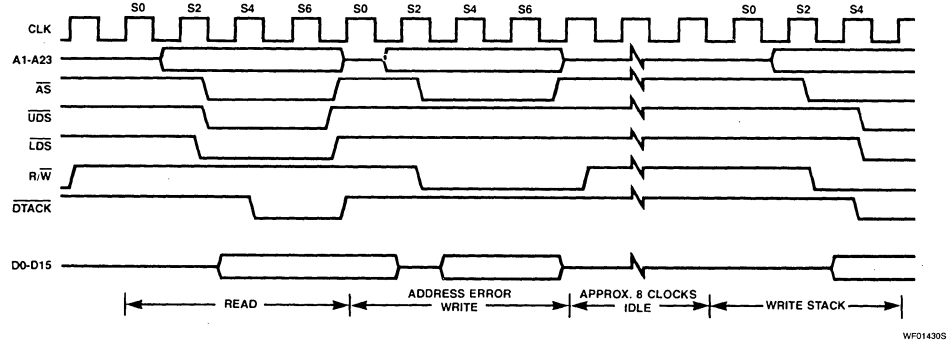


Figure 42. Address Error Timing Diagram

shown in figure 40. This word contains a processor version number in bits 10 through 13, that must match the version number of the SCN68010 that is attempting to read the data. This validity check is used to insure that in dual processor systems, the data will be properly interpreted by the RTE instruction if the two processors are of different versions. If the version number is incorrect for this processor, the RTE instruction will be aborted and exception processing will begin for a format error exception. Since the stack pointer is not updated until the RTE instruction has successfully read all of the stack data, a format error occurring at this point will not stack new data over the previous bus error stack information.

3. Determine data accessibility. If the long stack data is valid, the SCN68010 performs a read from the last word (SP + 56) of the long stack to determine data accessibility. If this read is terminated normally, the processor assumes that the remaining words on the stack frame are also accessible. If a bus error is signaled before or during this read, a bus error exception is taken as usual. After this read, the processor must be able to load the remaining data without receiving a bus error; therefore, if a bus error occurs on any of the remaining stack reads, the SCN68010 treats this as a double bus fault and enters the halted state.

INTERFACE WITH SYNCHRONOUS PERIPHERALS

To interface the synchronous peripherals with the asynchronous SCN68010, the processor modifies its bus cycle to meet the synchronous cycle requirements whenever a synchronous device address is detected. This is possible since both processors use memory mapped I/O. Figure 43 is a flowchart of the interface operation between the processor and synchronous devices.

Data Transfer Operation

Three signals on the processor provide the synchronous interface. They are: enable (E), valid memory address ($\overline{VMA}$), and valid peripheral address ($\overline{VPA}$). Enable corresponds to the E or phase 2 signal in existing synchronous systems. The bus frequency is one tenth of the incoming SCN68010 clock frequency. The timing of E allows 1 megahertz peripherals to be used with an 8 megahertz SCN68010. Enable has a 60/40 duty cycle; that is, it is low for six input clocks and high for four input clocks. This duty cycle allows the processor to do successive $\overline{VPA}$ accesses on successive E pulses.

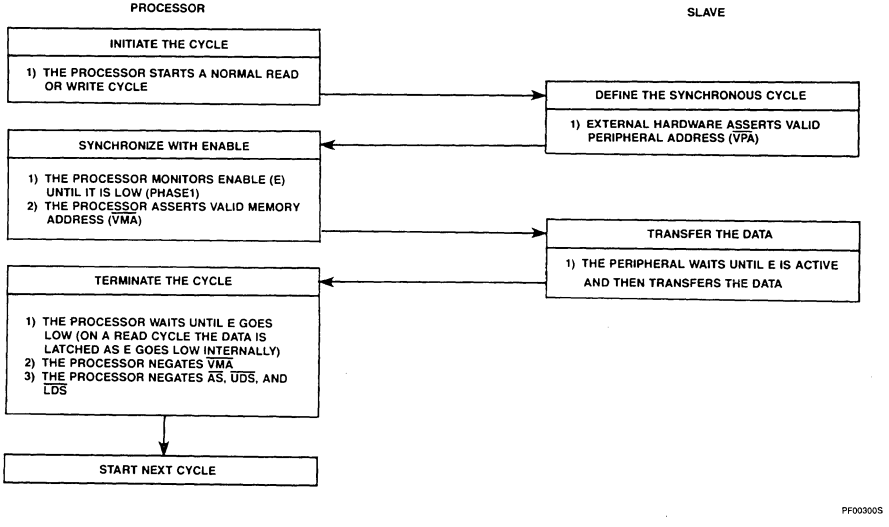


Figure 43. Synchronous Interfacing Flowchart

Synchronous cycle timing is given in figure 44. At state zero (S0) in the cycle, the address bus is in the high-impedance state. A function code is asserted on the function code output lines. One-half clock later, in state 1, the address bus is released from the high-impedance state.

During state 2, the address strobe ($\overline{AS}$) is asserted to indicate that there is a valid address on the address bus. If the bus cycle is a read cycle, the upper and/or lower data strobes are also asserted in state 2. If the bus cycle is a write cycle, the read/write (R/$\overline{W}$) signal is switched to low (write) during state 2. One-half clock later, in state 3, the write data is placed on the data bus, and in state 4 the data strobes are issued to indicate valid data on the data bus. The processor now inserts wait states until it recognizes the assertion of $\overline{VPA}$.

The $\overline{VPA}$ input signals the processor that the address on the bus is the address of a synchronous device (or an area reserved for synchronous devices) and that the bus should conform to the phase 2 transfer characteristics of the synchronous bus. Valid peripheral address is derived by decoding the address bus, conditioned by the address strobe. Chip select for the synchronous peripherals should be derived by decoding the address bus conditioned by $\overline{VMA}$.

After recognition of $\overline{VPA}$, the processor assures that the enable (E) is low, by waiting if necessary, and subsequently asserts $\overline{VMA}$ two clock cycles before E goes high. $\overline{VMA}$ is then used as part of the chip select equation of the peripheral. This ensures that the synchronous peripherals are selected and deselected at the correct time. The peripheral now runs its cycle during the high portion of the E signal. Figures 44 and 45 depict the best and worst case synchronous cycle timing; this cycle length is dependent strictly upon when $\overline{VPA}$ is asserted in relationship to the E clock.

If we assume that external circuitry asserts $\overline{VPA}$ as soon as possible after the assertion of $\overline{AS}$, then $\overline{VPA}$ will be recognized as being asserted on the falling edge of S4. In this case, no "extra" wait cycles will be inserted prior to the recognition of $\overline{VPA}$ asserted and only the wait cycles inserted to synchronize with the E clock will determine the total length of the cycle. In any case, the synchronization delay will be some integral number of clock cycles within the following two extremes:

1. Best Case — $\overline{VPA}$ is recognized as being asserted on the falling edge three clock cycles before E rises (or three clock cycles after E falls).
2. Worst Case — $\overline{VPA}$ is recognized as being asserted on the falling edge two clock cycles before E rises (or four clock cycles after E falls).

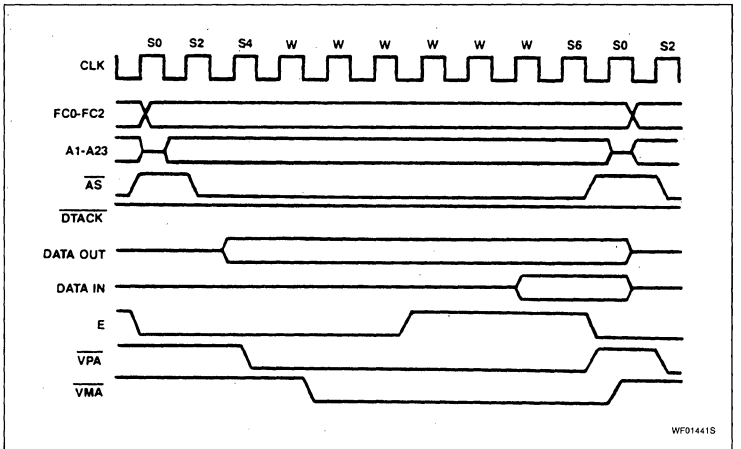


Figure 44. SCN68010 to Synchronous Peripheral Timing Diagram — Best Case

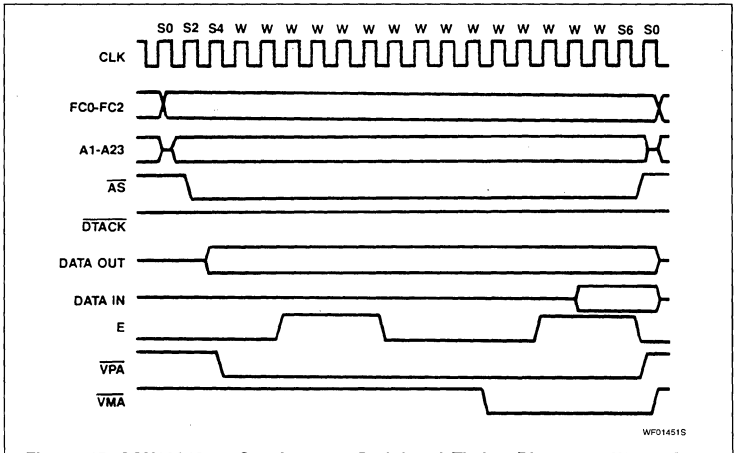


Figure 45. SCN68010 to Synchronous Peripheral Timing Diagram — Worst Case

During a read cycle, the processor latches the peripheral data in state 6. For all cycles, the processor negates the address and data strobes one-half clock cycle later in state 7 and the enable signal goes low at this time. Another half clock later, the address bus is put in the high-impedance state. During a write cycle, the data bus is put in the high-impedance state and the read/write signal is switched high. The peripheral logic must remove $\overline{VPA}$ within one clock after the address strobe is negated.

$\overline{DTACK}$ should not be asserted while $\overline{VPA}$ is asserted. Notice that the SCN68010 $\overline{VMA}$ is active low, contrasted with the active high synchronous VMA. This allows the processor to put its buses in the high-impedance state on DMA requests without inadvertently selecting the peripherals.

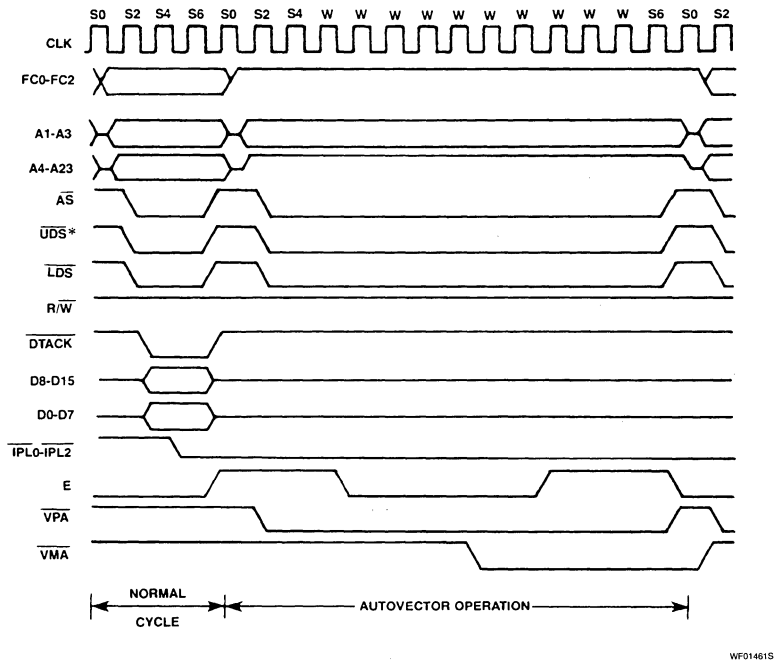


Figure 46. Autovector Operation Timing Diagram

Interrupt Interface Operation

During an interrupt acknowledge cycle while the processor is fetching the vector, if $\overline{VPA}$ is asserted the SCN68010 will assert $\overline{VMA}$ and complete a normal synchronous read cycle as shown in figure 46. The processor will then use an internally generated vector that is a function of the interrupt being serviced. This process is known as autovectoring. The seven autovectors are vector numbers 25 through 31 (decimal).

Autovectoring operates in the same fashion (but is not restricted to) the synchronous interrupt sequence. The basic difference is that there are six normal interrupt vectors and one NMI type vector. As with both the synchronous and the SCN68010's normal vectored interrupt, the interrupt service routine can be located anywhere in the address space. This is due to the fact that while the vector numbers are fixed, the contents of the vector table entries are assigned by the user.

Since $\overline{VMA}$ is asserted during autovectoring, the synchronous peripheral address decoding should prevent unintended accesses.

INSTRUCTION SET AND EXECUTION TIMES

Instruction Set

The following paragraphs provide information about the addressing categories and instruction set of the SCN68010.

Addressing Categories

Effective address modes may be categorized by the ways in which they may be used. The following classifications will be used in the instruction definitions:

Data — If an effective address mode may be used to refer to data operands, it is considered a data addressing effective address mode.

Memory — If an effective address mode may be used to refer to memory operands, it is considered a memory addressing effective address mode.

Alterable — If an effective address mode may be used to refer to alterable (writeable) operands, it is considered an alterable addressing effective address mode.

Control — If an effective address mode may be used to refer to memory operands without an associated size, it is considered a control addressing effective address mode.

Table 22. EFFECTIVE ADDRESSING MODE CATEGORIES

EFFECTIVE ADDRESS MODES	MODE	REGISTER	DATA	ADDRESSING CATEGORIES		
				Memory	Control	Alterable
Dn	000	Register number	X	–	–	X
An	001	Register number	–	–	–	X
(An)	010	Register number	X	X	X	X
(An)+	011	Register number	X	X	–	X
–(An)	100	Register number	X	X	–	X
d(An)	101	Register number	X	X	X	X
d(An, ix)	110	Register number	X	X	X	X
xxx.W	111	000	X	X	X	X
xxx.L	111	001	X	X	X	X
d(PC)	111	010	X	X	X	–
d(PC, ix)	111	011	X	X	X	–
#xxx	111	100	X	X	–	–

These categories may be combined, so that additional, more restrictive, classifications may be defined. For example, the instruction descriptions use such classifications as alterable memory or data alterable. The former refers to those addressing modes which are both alterable and memory addresses, and the latter refers to addressing modes which are both data and alterable. Table 22 shows the various categories to which each of the effective address modes belong. Table 23 is the instruction set summary.

2

Table 23. INSTRUCTION SET

MNEMONIC	DESCRIPTION	OPERATION	CONDITION CODES				
			X	N	Z	V	C
ABCD	Add decimal with extend	(Destination)$_{10}$ + (Source)$_{10}$ + X → Destination	*	U	*	U	*
ADD	Add binary	(Destination) + (Source) → Destination	*	*	*	*	*
ADDA	Add address	(Destination) + (Source) → Destination	–	–	–	–	–
ADDI	Add immediate	(Destination) + Immediate Data → Destination	*	*	*	*	*
ADDQ	Add quick	(Destination) + Immediate Data → Destination	*	*	*	*	*
ADDX	Add extended	(Destination) + (Source) + X= Destination	*	*	*	*	*
AND	AND logical	(Destination) ∧ Source → Destination	–	*	*	0	0
ANDI	AND immediate	(Destination) ∧ Immediate Data → Destination	–	*	*	0	0
ANDI to CCR	AND immediate to condition codes	(Source) ∧ CCR → CCR	*	*	*	*	*
ANDI to SR	AND immediate to status register	(Source) ∧ SR → SR	*	*	*	*	*
ASL, ASR	Arithmetic shift	(Destination) Shifted by < count > → Destination	*	*	*	*	*
B_{CC}	Branch conditionally	If $_{CC}$ the PC + d → PC	–	–	–	–	–
BCHG	Test a bit and change	~(< bit number >) OF Destination → Z ~(< bit number >) OF Destination → < bit number > OF Destination	–	–	*	–	–
BCLR	Test a bit and clear	~(< bit number >) OF Destination → Z 0 → < bit number > → OF Destination	–	–	*	–	–
BRA	Branch always	(PC) + displacement → PC	–	–	–	–	–
BSET	Test a bit and set	~(< bit number >) OF Destination → Z 1 → < bit number > OF Destination	–	–	*	–	–
BSR	Branch to subroutine	(PC) → –(SP); (PC) + d → PC	–	–	–	–	–
BTST	Test a bit	~(< bit number >) OF Destination → Z	–	–	*	–	–
CHK	Check register against bounds	If Dn < 0 or Dn > (< ea >) then TRAP	–	*	U	U	U
CLR	Clear and operand	0 → Destination	–	0	1	0	0
CMP	Compare	(Destination) – (Source)	–	*	*	*	*
CMPA	Compare address	(Destination) – (Source)	–	*	*	*	*
CMPI	Compare immediate	(Destination) – Immediate Data	–	*	*	*	*
CMPM	Compare memory	(Destination) – (Source)	–	*	*	*	*
DB_{CC}	Test condition, decrement and branch	if ~$_{CC}$ then Dn – 1 → Dn; if Dn ≠ –1 then PC + d → PC	–	–	–	–	–
DIVS	Signed divide	(Destination)/(Source) → Destination	–	*	*	*	0
DIVU	Unsigned divide	(Destination)/(Source) → Destination	–	*	*	*	0
EOR	Exclusive OR logical	(Destination) ⊕ (Source) → Destination	–	*	*	0	0
EORI	Exclusive OR immediate	(Destination) ⊕ Immediate Data → Destination	–	*	*	0	0
EORI to CCR	Exclusive OR immediate to condition codes	(Source) ⊕ CCR → CCR	*	*	*	*	*
EORI to SR	Exclusive OR immediate to status register	(Source) ⊕ SR → SR	*	*	*	*	*
EXG	Exchange register	(Rx) ↔ (Ry)	–	–	–	–	–
EXT	Sign extend	(Destination) Sign-Extended → Destination	–	*	*	0	0
JMP	Jump	(Destination) → PC	–	–	–	–	–
JSR	Jump to subroutine	(PC) → –(SP); Destination → PC	–	–	–	–	–
LEA	Load effective address	Destination → An	–	–	–	–	–
LINK	Link and allocate	(An) → –(SP); (SP) → An; (SP) + → SP	–	–	–	–	–
LSL, LSR	Logical shift	(Destination) Shifted by < count > → Destination	*	*	*	0	*
MOVE	Move data from source to destination	(Source) → Destination	–	*	*	0	0
MOVE to CCR	Move to condition code	(Source) → CCR	*	*	*	*	*
MOVE from CCR	Move from condition codes	(CCR) → Destination	–	–	–	–	–
MOVE to SR	Move to the status register	(Source) → SR	*	*	*	*	*
MOVE from SR	Move from the status register	(SR) → Destination	–	–	–	–	–
MOVE USP	Move user stack pointer	(USP) → An; (An)=Ar USP	–	–	–	–	–
MOVEA	Move address	(Source) → Destination	–	–	–	–	–
MOVEC	Move control register	(Cr) → Rn; (Rn) → Cr	–	–	–	–	–

Table 23. INSTRUCTION SET (Continued)

MNEMONIC	DESCRIPTION	OPERATION	CONDITION CODES				
			X	N	Z	V	C
MOVEM	Move multiple registers	(Registers) → Destination (Source) → Registers	–	–	–	–	–
MOVEP	Move peripheral data	(Source) → Destination	–	–	–	–	–
MOVEQ	Move quick	Immediate Data → Destination	–	*	*	0	0
MOVES	Move alternate address space	(Dn) → Destination; (Source) → Dn	–	–	–	–	–
MULS	Signed multiply	(Destination)X (Source) → Destination	–	*	*	0	0
MULU	Unsigned multiply	(Destination)X (Source) → Destination	–	*	*	0	0
NBCD	Negate decimal with extend	$0 - (\text{Destination})_{10} - X$ → Destination	*	U	*	U	*
NEG	Negate	0 – (Destination) → Destination	*	*	*	*	*
NEGX	Negate with extend	0 – (Destination) – X → Destination	*	*	*	*	*
NOP	No operation	–	–	–	–	–	–
NOT	Logical complement	~ (Destination) → Destination	–	*	*	0	0
OR	Inclusive OR logical	(Destination) v (Source) → Destination	–	*	*	0	0
ORI	Inclusive OR immediate	(Destination) v Immediate Data → Destination	–	*	*	0	0
ORI to CCR	Inclusive OR immediate to condition codes	(Source) v CCR → CCR	*	*	*	*	*
ORI to SR	Inclusive OR immediate to status register	(Source) v SR → SR	*	*	*	*	*
PEA	Push effective address	Destination → – (SP)	–	–	–	–	–
RESET	Reset external device	–	–	–	–	–	–
ROL, ROR	Rotate (without extend)	(Destination) Rotated by < count > → Destination	–	*	*	0	*
ROXL, ROXR	Rotate with extend	(Destination) Rotated by < count > → Destination	*	*	*	0	*
RTD	Return and deallocate stack	(SP)+ → PC; (SP) + d → SP	–	–	–	–	–
RTE	Return from exception	(SP) + → SR; (SP) + → PC	*	*	*	*	*
RTR	Return and restore condition codes	(SP) + → CC; (SP) + → PC	*	*	*	*	*
RTS	Return from subroutine	(SP) + → PC	–	–	–	–	–
SBCD	Subtract decimal with extend	$(\text{Destination})_{10} - (\text{Source})_{10} - X$ → Destination	*	U	*	U	*
S_{CC}	Set according to condition	If $_{CC}$ then 1's → Destination else 0's → Destination	–	–	–	–	–
STOP	Load status register and stop	Immediate Data → SR; STOP	*	*	*	*	*
SUB	Subtract binary	(Destination) – (Source) → Destination	*	*	*	*	*
SUBA	Subtract address	(Destination) – (Source) → Destination	–	–	–	–	–
SUBI	Subtract immediate	(Destination) – Immediate Data → Destination	*	*	*	*	*
SUBQ	Subtract quick	(Destination) – Immediate Data → Destination	*	*	*	*	*
SUBX	Subtract with extend	(Destination) – (Source) – X → Destination	*	*	*	*	*
SWAP	Swap register halves	Register [31:16] ↔ Register [15:0]	–	*	*	0	0
TAS	Test and set an operand	(Destination) Tested → CC; 1 → [7] OF Destination	–	*	*	0	0
TRAP	Trap	(PC) → – (SSP); (SR) → – (SSP); (Vector) → PC	–	–	–	–	–
TRAPV	Trap on overflow	If V set then TRAP	–	–	–	–	–
TST	Test and operand	(Destination) Tested → CC	–	*	*	0	0
UNLK	Unlink	(An) → SP; (SP) + → An	–	–	–	–	–

NOTES:

[ ] = bit number
⊕ logical exclusive OR
∧ logical AND
V logical OR
~ logical complement

* affected
– unaffected
0 cleared
1 set
U undefined

Instruction Prefetch

The SCN68010 uses a two-word tightly-coupled instruction prefetch mechanism to enhance performance. This mechanism is described in terms of the microcode operations involved. If the execution of an instruction is defined to begin when the microroutine for that instruction is entered, some features of the prefetch mechanism can be described.

1. When execution of an instruction begins, the operation word and the word following have already been fetched. The operation word is in the instruction decoder.
2. In the case of multi-word instructions, as each additional word of the instruction is used internally, a fetch is made to the instruction stream to replace it.
3. The last fetch for an instruction from the instruction stream is made when the operation word is discarded and decoding is started on the next instruction.
4. If the instruction is a single-word instruction causing a branch, the second word is not used. But because this word is fetched by the preceding instruction, it is impossible to avoid this superfluous fetch.
5. In the case of an interrupt or trace exception, both prefetched words are not used.
6. The program counter usually points to the last word fetched from the instruction stream.

Loop Mode Operation

The SCN68010 has several features that provide efficient execution of program loops. One of these features is the DBcc looping primitive instruction. The DBcc instruction operates on three operands, a loop counter, a branch condition, and a branch displacement. When the DBcc is executed in loop mode, the contents of the low order word of the register specified as the loop counter is decremented by one and compared to minus one. If equal to minus one, the result of the decrement is placed back into the count register and the next sequential instruction is executed, otherwise the condition code register is checked against the specified branch condition. If the condition is true, the result of the decrement is discarded and the next sequential instruction is executed. Finally, if the count register is not equal to minus one and the branch condition is false, the branch displacement is added to the program counter and instruction execution continues at that new address. Note that this is slightly different than non-looped execution; however, the results are the same.

```
        LEA      SOURCE, A0     Load a pointer to source data
        LEA      DEST, A1       Load a pointer to destination
        MOVE. W  #LENGTH, D0    Load the counter register
LOOP    MOVE. W  (A0)+, (A1)+   Loop to move the block of data
        DBEQ     D0, LOOP       Stop if data word is zero
```

Figure 47. DBcc Loop Program Example

An example of using the DBcc instruction in a simple loop for moving a block of data is shown in figure 47. In this program, the block of data 'LENGTH' words long at address 'SOURCE' is to be moved to address 'DEST' provided that none of the words moved are equal to zero. When the effect of instruction prefetch on this loop is examined it can be seen that the bus activity during the loop execution would be:

1. Fetch the MOVE.W instruction,
2. Fetch the DBEQ instruction,
3. Read the operand where A0 points,
4. Write the operand where A1 points,
5. Fetch the DBEQ branch displacement, and
6. If loop conditions are met, return to step 1.

During this loop, five bus cycles are executed; however, only two bus cycles perform the data movement. Since the SCN68010 has a two word prefetch queue in addition to a one word instruction decode register, it is evident that the three instruction fetches in this loop could be eliminated by placing the MOVE.W word in the instruction decode register and holding the DBEQ instruction and its branch displacement in the prefetch queue. The SCN68010 has the ability to do this by entering the loop mode of operation. During loop mode operation, all opcode fetches are suppressed and only operand reads and writes are performed until an exit loop condition is met.

Loop mode operation is transparent to the programmer, with only two conditions required for the SCN68010 to enter the loop mode. First, a DBcc instruction must be executed with both branch conditions met and a branch displacement of minus four; which indicates that the branch is to a one word instruction preceding the DBcc instruction. Second, when the processor fetches the instruction at the branch address, it is checked to determine whether it is one of the allowed looping instructions. If it is, the loop mode is entered. Thus, the single word looped instruction and the first word of the DBcc instruction will each be fetched twice when the loop is entered; but no instruction fetches will occur again until the DBcc loop conditions fail.

In addition to the normal termination conditions for a loop, there are several conditions that will cause the SCN68010 to exit loop mode operation. These conditions are interrupts, trace exceptions, reset errors, and bus errors. Interrupts are honored after each execution of the DBcc instruction, but not after the execution of the looped instruction. If an interrupt exception occurs, loop mode operation is terminated and can be restarted on return from the interrupt handler. If the T bit is set, trace exceptions will occur at the end of both the loop instruction and the DBcc instruction and thus loop mode operation is not available. Reset will abort all processing, including the loop mode. Bus errors during the loop mode will be treated the same as in normal processing; however, when the RTE instruction is used to continue the execution of the looped instruction, the three word loop will not be re-fetched.

The loopable instructions available on the SCN68010 are listed in table 24. These instructions may use the three address register indirect addressing modes to form one word looping instructions; (An), (An) + , and − (An).

Table 24. MC68010 LOOPABLE INSTRUCTIONS

OPCODES	APPLICABLE ADDRESSING MODES
MOVE [BWL]	(Ay) to (Ax) (Ay) to (Ax)+ (Ay) to -(Ax) (Ay)+ to (Ax) (Ay)+ to (Ax)+ (Ay)+ to -(Ax) -(Ay) to (Ax) -(Ay) to (Ax)+ -(Ay) to -(Ax) Ry to (Ax) Ry to (Ax)+
ADD [BWL] AND [BWL] CMP [BWL] OR [BWL] SUB [BWL]	(Ay) to Dx (Ay)+ to Dx -(Ay) to Dx
ADDA [WL] CMPA [WL] SUBA [WL]	(Ay) to Ax -(Ay) to Ax (Ay)+ to Ax
ADD [BWL] AND [BWL] EOR [BWL] OR [BWL] SUB [BWL]	Dx to (Ay) Dx to (Ay)+ Dx to -(Ay)
ABCD [B] ADDX [BWL] SBCD [B] SUBX [BWL]	-(Ay) to -(Ax)
CMP [BWL]	(Ay)+ to (Ax)+
CLR [BWL] NEG [BWL] NEGX [BWL] NOT [BWL] TST [BWL] NBCD [B]	(Ay) (Ay)+ -(Ay)
ASL [W] ASR [W] LSL [W] LSR [W] ROL [W] ROR [W] ROXL [W] ROXR [W]	(Ay) by #1 (Ay)+ by #1 -(Ay) by #1

NOTE:
[B, W, or L] indicate an operand size of byte, word, or long word.

Instruction Execution Times

The following paragraphs contain listings of the instruction execution times in terms of external clock (CLK) periods. In this timing data, it is assumed that both memory read and write cycle times are four clock periods. Any wait states caused by a longer memory cycle must be added to the total instruction time. The number of bus read and write cycles for each instruction is also included with the timing data. This data is enclosed in parenthesis following the execution periods and is shown as (r/w) where r is the number of read cycles and w is the number of write cycles.

NOTE
The number of clock periods includes instruction fetches and all applicable operand fetches and stores.

Operand Effective Address Calculation Times

Table 25 lists the number of clock periods required to compute an instruction's effective address. It includes fetching of any extension words, the address computation, and fetching of the memory operand if necessary. Several instructions do not need the operand at an effective address to be fetched and thus require fewer clock periods to calculate a given effective address than the instructions that do fetch the effective address operand. The number of bus read and write cycles is shown in parentheses as (r/w). Note there are no write cycles involved in processing the effective address.

Table 25. EFFECTIVE ADDRESS CALCULATION TIMES

ADDRESSING MODE		BYTE, WORD		LONG	
		Fetch	No Fetch	Fetch	No Fetch
	Register				
Dn	Data register direct	0(0/0)	-	0(0/0)	-
An	Address register direct	0(0/0)	-	0(0/0)	-
	Memory				
(An)	Address register indirect	4(1/0)	2(0/0)	8(2/0)	2(0/0)
(An)+	Address register indirect with postincrement	4(1/0)	4(0/0)	8(2/0)	4(0/0)
-(An)	Address register indirect with predecrement	6(1/0)	4(0/0)	10(2/0)	4(0/0)
d(An)	Address register indirect with displacement	8(2/0)	4(0/0)	12(3/0)	4(1/0)
d(An, ix)*	Address register indirect with index	10(2/0)	8(1/0)	14(3/0)	8(1/0)
xxx.W	Absolute short	8(2/0)	4(1/0)	12(3/0)	4(1/0)
xxx.L	Absolute long	12(3/0)	8(2/0)	16(4/0)	8(2/0)
d(PC)	Program counter with displacement	8(2/0)	-	12(3/0)	-
d(PC, ix)	Program counter with index	10(2/0)	-	14(3/0)	-
#xxx	Immediate	4(1/0)	-	8(2/0)	-

*The size of the index register (ix) does not affect execution time.

Table 26. MOVE BYTE AND WORD INSTRUCTION EXECUTION TIMES

SOURCE	DESTINATION								
	Dn	An	(An)	(An)+	-(An)	d(An)	d(An, ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
(An)	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
(An)+	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
-(An)	10(2/0)	10(2/0)	14(2/1)	14(2/1)	14(2/1)	18(3/1)	20(3/1)	18(3/1)	22(4/1)
d(An)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
d(An, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
xxx.W	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
xxx.L	16(4/0)	16(4/0)	20(4/1)	20(4/1)	20(4/1)	24(5/1)	26(5/1)	24(5/1)	28(6/1)
d(PC)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
d(PC, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
#xxx	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)

*The size of the index register (ix) does not affect execution time.

Move Instruction Execution Times

Tables 26, 27, 28, and 29 indicate the number of clock periods for the move instruction. This data includes instruction fetch, operand reads, and operand writes. The number of bus read and write cycles is shown in parenthesis as (r/w).

Table 27. MOVE BYTE AND WORD INSTRUCTION LOOP MODE EXECUTION TIMES

	LOOP CONTINUED			LOOP TERMINATED					
	Valid Count, cc False			Valid Count, cc True			Expired Count		
	Destination								
SOURCE	(An)	(An)+	-(An)	(An)	(An)+	-(An)	(An)	(An)+	-(An)
Dn	10(0/1)	10(0/1)	-	18(2/1)	18(2/1)	-	16(2/1)	16(2/1)	-
An*	10(0/1)	10(0/1)	-	18(2/1)	18(2/1)	-	16(2/1)	16(2/1)	-
(An)	14(1/1)	14(1/1)	16(1/1)	20(3/1)	20(3/1)	22(3/1)	18(3/1)	18(3/1)	20(3/1)
(An)+	14(1/1)	14(1/1)	16(1/1)	20(3/1)	20(3/1)	22(3/1)	18(3/1)	18(3/1)	20(3/1)
-(An)	16(1/1)	16(1/1)	18(1/1)	22(3/1)	22(3/1)	24(3/1)	20(3/1)	20(3/1)	22(3/1)

*Word only.

Table 28. MOVE LONG INSTRUCTION EXECUTION TIMES

SOURCE	DESTINATION								
	Dn	An	(An)	(An)+	-(An)	d(An)	d(An, ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	12(1/2)	12(1/2)	14(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An	4(1/0)	4(1/0)	12(1/2)	12(1/2)	14(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
(An)	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
(An)+	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
-(An)	14(3/0)	14(3/0)	22(3/2)	22(3/2)	22(3/2)	26(4/2)	28(4/2)	26(4/2)	30(5/2)
d(An)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
d(An, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
xxx.W	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
xxx.L	20(5/0)	20(5/0)	28(5/2)	28(5/2)	28(5/2)	32(6/2)	34(6/2)	32(6/2)	36(7/2)
d(PC)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(5/2)
d(PC, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
#xxx	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)

*The size of the index register (ix) does not affect execution time.

Table 29. MOVE LONG INSTRUCTION LOOP MODE EXECUTION TIMES

	LOOP CONTINUED			LOOP TERMINATED					
	Valid Count, cc False			Valid Count, cc True			Expired Count		
	Destination								
SOURCE	(An)	(An)+	-(An)	(An)	(An)+	-(An)	(An)	(An)+	-(An)
Dn	14(0/2)	14(0/2)	-	20(2/2)	20(2/2)	-	18(2/2)	18(2/2)	-
An	14(0/2)	14(0/2)	-	20(2/2)	20(2/2)	-	18(2/2)	18(2/2)	-
(An)	22(2/2)	22(2/2)	24(2/2)	28(4/2)	28(4/2)	30(4/2)	24(4/2)	24(4/2)	26(4/2)
(An)+	22(2/2)	22(2/2)	24(2/2)	28(4/2)	28(4/2)	30(4/2)	24(4/2)	24(4/2)	26(4/2)
-(An)	24(2/2)	24(2/2)	26(2/2)	30(4/2)	30(4/2)	32(4/2)	26(4/2)	26(4/2)	28(4/2)

Standard Instruction Execution Times

The number of clock periods shown in tables 30 and 31 indicate the time required to perform the operations, store the results, and read the next instruction. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In tables 30 and 31 the headings have the following meanings: An = address register operand, Dn = data register operand, ea = an operand specified by an effective address, and M = memory effective address operand.

Table 30. STANDARD INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	op <ea>, An***	op <ea>, Dn	op Dn, <M>
ADD	byte, word	8(1/0)+	4(1/0)+	8(1/1)+
	long	6(1/0)+	6(1/0)	12(1/2)+
AND	byte, word	-	4(1/0)+	8(1/1)+
	long	-	6(1/0)+	12(1/2)+
CMP	byte, word	6(1/0)+	4(1/0)+	-
	long	6(1/0)+	6(1/0)+	-
DIVS	-	-	122(1/0)+	-
DIVU	-	-	108(1/0)+	-
EOR	byte, word	-	4(1/0)+ **	18(1/1)+
	long	-	6(1/0)**	12(1/2)+
MULS	-	-	42(1/0)+ *	-
MULU	-	-	40(1/0)+	-
OR	byte, word	-	4(1/0)+	8(1/1)+
	long	-	6(1/0)+	12(1/2)+
SUB	byte, word	8(1/0)+	4(1/0)+	8(1/1)+
	long	6(1/0)+	6(1/0)+	12(1/2)+

NOTES:
+ add effective address calculation time
* indicates maximum value
** only available addressing mode is data register direct
*** word or long only

Table 31. STANDARD INSTRUCTION LOOP MODE EXECUTION TIMES

		LOOP CONTINUED			LOOP TERMINATED					
		Valid Count, cc False			Valid Count, cc True			Expired Count		
INSTRUCTION	SIZE	op <ea>, An*	op <ea>, Dn	op Dn, <ea>	op <ea>, An*	op <ea>, Dn	op Dn, <ea>	op <ea>, An*	op <ea>, Dn	op Dn, <ea>
ADD	byte, word	18(1/0)	16(1/0)	16(1/1)	24(3/0)	22(3/0)	22(3/1)	22(3/0)	20(3/0)	20(3/1)
	long	22(2/0)	22(2/0)	24(2/2)	28(4/0)	28(4/0)	30(4/2)	26(4/0)	26(4/0)	28(4/2)
AND	byte, word	–	16(1/0)	16(1/1)	–	22(3/0)	22(3/1)	–	20(3/0)	20(3/1)
	long	–	22(2/0)	24(2/2)	–	28(4/0)	30(4/2)	–	26(4/0)	28(4/2)
CMP	byte, word	12(1/0)	12(1/0)	–	18(3/0)	18(3/0)	–	16(3/0)	16(4/0)	–
	long	18(2/0)	18(2/0)	–	24(4/0)	24(4/0)	–	20(4/0)	20(4/0)	–
EOR	byte, word	–	–	16(1/0)	–	–	22(3/1)	–	–	20(3/1)
	long	–	–	24(2/2)	–	–	30(4/2)	–	–	28(4/2)
OR	byte, word	–	16(1/0)	16(1/0)	–	22(3/0)	22(3/1)	–	20(3/0)	20(3/1)
	long	–	22(2/0)	24(2/2)	–	28(4/0)	30(4/2)	–	26(4/0)	28(4/2)
SUB	byte, word	18(1/0)	16(1/0)	16(1/1)	24(3/0)	22(3/0)	22(3/1)	22(3/0)	20(3/0)	20(3/1)
	long	22(2/0)	20(2/0)	24(2/2)	28(4/0)	26(4/0)	(30(4/2)	26(4/0)	24(4/0)	28(4/2)

*Word or long only.
<ea> may be (An), +(An), or –(An) only. Add two clock periods to the table value if <ea> is –(An).

Immediate Instruction Execution Times

The number of clock periods shown in table 32 includes the time to fetch immediate operands, perform the operations, store the results, and read the next operation. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In table 32, the headings have the following meanings: # = immediate operand, Dn = data register operand, AN = address register operand, and M = memory operand.

Table 32. IMMEDIATE INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	op #, Dn	op #, An	op #, M
ADDI	byte, word	8(2/0)	–	12(2/1)+
	long	14(3/0)	–	20(3/2)+
ADDQ	byte, word	4(1/0)	4(1/0)*	8(1/1)+
	long	8(1/0)	8(1/0)	12(1/2)+
ANDI	byte, word	8(2/0)	–	12(2/1)+
	long	14(3/0)	–	20(3/1)+
CMPI	byte, word	8(2/0)	–	8(2/0)+
	long	12(3/0)	–	12(3/0)+
EORI	byte, word	8(2/0)	–	12(2/1)+
	long	14(3/0)	–	20(3/2)+
MOVEQ	long	4(1/0)	–	–
ORI	byte, word	8(2/0)	–	12(2/1)+
	long	14(3/0)	–	20(3/2)+
SUBI	byte, word	8(2/0)	–	12(2/1)+
	long	14(3/0)	–	20(3/2)+
SUBQ	byte, word	4(1/0)	4(1/0)*	8(1/1)+
	long	8(1/0)	8(1/0)	12(1/2)+

+ add effective address calculation time
* word only

Single Operand Instruction Execution Times

Tables 33, 34, and 35 indicate the number of clock periods for the single operand instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 33. SINGLE OPERAND INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	REGISTER	MEMORY
NBCD	byte	6(1/0)	8(1/1)+
NEG	byte, word	4(1/0)	8(1/1)+
	long	6(1/0)	12(1/2)+
NEGX	byte, word	4(1/0)	8(1/1)+
	long	6(1/0)	12(1/2)+
NOT	byte, word	4(1/0)	8(1/1)+
	long	6(1/0)	12(1/2)+
S_{CC}	byte, false	4(1/0)	8(1/1)+*
	byte, true	4(1/0)	8(1/1)+*
TAS	byte	4(1/0)	14(2/1)+*
TST	byte, word	4(1/0)	4(1/0)
	long	4(1/0)	4(1/0)+

+ add effective address calculation time
* Use non-fetching effective address calculation time.

Table 34. CLEAR INSTRUCTION EXECUTION TIMES

	SIZE	Dn	An	(An)	(An)+	-(An)	d(An)	(An, ix)*	xxx.W	xxx.L
CLR	byte, word	4(1/0)	–	8(1/1)	8(1/1)	10(1/1)	12(2/1)	16(2/1)	12(2/1)	16(3/1)
	long	6(1/0)	–	12(1/2)	12(1/2)	14(1/2)	16(2/2)	20(2/2)	16(2/2)	20(3/2)

* The size of the index register (ix) does not affect execution time.

Table 35. SINGLE OPERAND INSTRUCTION LOOP MODE EXECUTION TIMES

		LOOP CONTINUED			LOOP TERMINATED					
		Valid Count, cc False			Valid Count, cc True			Expired Count		
INSTRUCTION	SIZE	(An)	(An)+	-(An)	(An)	(An)+	-(An)	(An)	(An)+	-(An)
CLR	byte, word	10(0/1)	10(0/1)	12(0/1)	18(2/1)	18(2/1)	20(2/0)	16(2/1)	16(2/1)	18(2/1)
	long	14(0/2)	14(0/2)	16(0/2)	22(2/2)	22(2/2)	24(2/2)	20(2/2)	20(2/2)	22(2/2)
NBCD	byte	18(1/1)	18(1/1)	20(1/1)	24(3/1)	24(3/1)	26(3/1)	22(3/1)	22(3/1)	24(3/1)
NEG	byte, word	16(1/1)	16(1/1)	18(2/2)	22(3/1)	22(3/1)	24(3/1)	20(3/1)	20(3/1)	22(3/1)
	long	24(2/2)	24(2/2)	26(2/2)	30(4/2)	30(4/2)	32(4/2)	28(4/2)	28(4/2)	30(4/2)
NEGX	byte, word	16(1/1)	16(1/1)	18(2/2)	22(3/1)	22(3/1)	24(3/1)	20(3/1)	20(3/1)	22(3/1)
	long	24(2/2)	24(2/2)	26(2/2)	30(4/2)	30(4/2)	32(4/2)	28(4/2)	28(4/2)	30(4/2)
NOT	byte, word	16(1/1)	16(1/1)	18(2/2)	22(3/1)	22(3/1)	24(3/1)	20(3/1)	20(3/1)	22(3/1)
	long	24(2/2)	24(2/2)	26(2/2)	30(4/2)	30(4/2)	32(4/2)	28(4/2)	28(4/2)	30(4/2)
TST	byte, word	12(1/0)	12(1/0)	14(1/0)	18(3/0)	18(3/0)	20(3/0)	16(3/0)	16(3/0)	18(3/0)
	long	18(2/0)	18(2/0)	20(2/0)	24(4/0)	24(4/0)	26(4/0)	20(4/0)	20(4/0)	22(4/0)

Shift/Rotate Instruction Execution Times

Tables 36 and 37 indicate the number of clock periods for the shift and rotate instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 36. SHIFT/ROTATE INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	REGISTER	MEMORY*
ASR, ASL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–
LSR, LSL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–
ROR, ROL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–
ROXR, ROXL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–

+ add effective address calculation time
n is the shift or rotate count
* word only

Table 37. SHIFT/ROTATE INSTRUCTION LOOP MODE EXECUTION TIMES

		LOOP CONTINUED			LOOP TERMINATED					
		Valid Count, cc False			Valid Count, cc True			Expired Count		
INSTRUCTION	SIZE	(An)	(An)+	–(An)	(An)	(An)+	–(An)	(An)	(An)+	–(An)
ASR, ASL	word	18(1/1)	18(1/1)	20(1/1)	24(3/1)	24(3/1)	26(3/1)	22(3/1)	22(3/1)	24(3/1)
LSR, LSL	word	18(1/1)	18(1/1)	20(1/1)	24(3/1)	24(3/1)	26(3/1)	22(3/1)	22(3/1)	24(3/1)
ROR, ROL	word	18(1/1)	18(1/1)	20(1/1)	24(3/1)	24(3/1)	26(3/1)	22(3/1)	22(3/1)	24(3/1)
ROXR, ROXL	word	18(1/1)	18(1/1)	20(1/1)	24(3/1)	24(3/1)	26(3/1)	22(3/1)	22(3/1)	24(3/1)

Bit Manipulation Instruction Execution Times

Table 38 indicates the number of clock periods required for the bit manipulation instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 38. BIT MANIPULATION INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	DYNAMIC		STATIC	
		Register	Memory	Register	Memory
BCHG	byte	–	8(1/1)+	–	12(2/1)+
	long	8(1/0)*	–	12(2/0)*	–
BCLR	byte	–	10(1/1)+	–	14(2/1)+
	long	10(1/0)*	–	14(2/0)*	–
BSET	byte	–	8(1/0)+	–	12(2/1)+
	long	8(1/0)*	–	12(2/0)*	–
BTST	byte	–	4(1/0)+	–	8(2/0)+
	long	6(1/0)*	–	10(2/0)	–

+ add effective address calculation time
* indicates maximum value

Conditional Instruction Execution Times
Table 39 indicates the number of clock periods required for the conditional instructions. The number of bus read and write cycles is indicated in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times
Table 40 indicates the number of clock periods required for the jump, jump-to-subroutine, load effective address, push effective address, and move multiple registers instructions. The number of bus read and write cycles is shown in parenthesis as (r/w).

Table 39. CONDITIONAL INSTRUCTION EXECUTION TIMES

INSTRUCTION	DISPLACEMENT	BRANCH TAKEN	BRANCH NOT TAKEN
B_{CC}	byte	10(2/0)	6(1/0)
	word	10(2/0)	10(2/0)
BRA	byte	10(2/0)	–
	word	10(2/0)	–
BSR	byte	18(2/2)	–
	word	18(2/2)	–
DB_{CC}	CC true	–	10(2/0)
	CC false	10(2/0)	16(3/0)

Table 40. JMP, JSR, LEA, PEA, AND MOVEM INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	(An)	(An)+	-(An)	d(An)	d(An, ix)+	xxx.W	xxx.L	d(PC)	d(PC, ix)*
JMP	–	8(2/0)	–	–	10(2/0)	14(3/0)	10(2/0)	12(3/0)	10(2/0)	14(3/0)
JSR	–	16(2/2)	–	–	18(2/2)	22(2/2)	18(2/2)	20(3/2)	18(2/2)	22(2/2)
LEA	–	4(1/0)	–	–	8(2/0)	12(2/0)	8(2/0)	12(3/0)	8(2/0)	12(2/0)
PEA	–	12(1/2)	–	–	16(2/2)	20(2/2)	16(2/2)	20(3/2)	16(2/2)	20(2/2)
MOVEM M → R	word	12 + 4n (3 + n/0)	12 + 4n (3 + n/0)	–	16 + 4n (4 + n/0)	18 + 4n (4 + n/0)	16 + 4n (4 + n/0)	20 + 4n (5 + n/0)	16 + 4n (4 + n/0)	18 + 4n (4 + n/0)
	long	12 + 8n (3 + 2n/0)	12 + 8n (3 + 2n/0)	–	16 + 8n (4 + 2n/0)	18 + 8n (4 + 2n/0)	16 + 8n (4 + 2n/0)	20 + 8n (5 + 2n/0)	16 + 8n (4 + 2n/0)	18 + 8n (4 + 2n/0)
MOVEM R → M	word	8 + 4n (2/n)	–	8 + 4n (2/n)	12 + 4n (3/n)	14 + 4n (3/n)	12 + 4n (3/n)	16 + 4n (4/n)	– –	– –
	long	8 + 8n (2/2n)	– –	8 + 8n (2/2n)	12 + 8n (3/2n)	14 + 8n (3/2n)	12 + 8n (3/2n)	16 + 8n (4/2n)	– –	– –

n is the number of registers to move
* The size of the index register (ix) does not affect the instruction's execution time

Multi-Precision Instruction Execution Times
Table 41 indicates the number of clock periods for the multi-precision instructions. The number of clock periods includes the time to fetch both operands, perform the operations, store the results, and read the next instructions. The number of read and write cycles is shown in parenthesis as (r/w).

In table 41, the headings have the following meanings: Dn = data register operand and M = memory operand.

Table 41. MULTI-PRECISION INSTRUCTION EXECUTION TIMES

		NON-LOOPED	LOOP MODE		
			Continued	Terminated	
			Valid Count, cc False	Valid Count, ccTrue	Expired Count
INSTRUCTION	SIZE	op Dn, Dn	op M, M*		
ADDX	byte, word	4(1/0)	18(3/10)	22(2/1)	28(4/1)	26(4/1)
	long	6(1/0)	30(5/2)	32(4/2)	38(6/2)	36(6/2)
CMPM	byte, word	–	12(3/0)	14(2/0)	20(4/0)	18(4/0)
	long	–	20(5/0)	24(4/0)	30(6/0)	26(6/0)
SUBX	byte, word	4(1/0)	18(3/1)	22(2/1)	28(4/1)	26(4/1)
	long	6(1/0)	30(5/2)	32(4/2)	38(6/2)	36(6/2)
ABCD	byte	6(1/0)	18(3/1)	24(2/1)	30(4/1)	28(4/1)
SBCD	byte	6(1/0)	18(3/1)	24(2/1)	30(4/1)	28(4/1)

*Source and destination ea is (An)+ for CMPM and -(An) for all others.

Miscellaneous Instruction Execution Times

Table 42 indicates the number of clock periods for the following miscellaneous instructions. The number of bus read and write cycle is shown in parenthesis as (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

Exception Processing Execution Times

Table 43 indicates the number of clock periods for exception processing. The number of clock periods includes the time for all stacking, the vector fetch, and the fetch of the first two instruction words of the handler routine. The number of bus read and write cycles is shown in parenthesis as (r/w).

Table 43. EXCEPTION PROCESSING EXECUTION TIMES

EXCEPTION	
Address error	126(4/26)
Breakpoint instruction*	42(5/4)
Bus error	126(4/26)
CHK instruction**	44(5/4)+
Divide by zero	42(5/4)
Illegal instruction	38(4/4)
Interrupt*	46(5/4)
MOVEC, illegal cr**	46(5/4)
Privilege violation	38(4/4)
Reset***	40(6/0)
RTE, illegal format	50(7/4)
RTE, illegal revision	70(12/4)
Trace	38(4/4)
TRAP instruction	38(4/4)
TRAPV instruction	40(5/4)

+add effective address calculation time.
*The interrupt acknowledge and breakpoint cycles are assumed to take four clock periods.
**Indicates maximum value
***Indicates the time from when $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ are first sampled as negated to when instruction execution starts.

Table 42. MISCELLANEOUS INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	REGISTER	MEMORY	REGISTER → DESTINATION**	SOURCE** → REGISTER
ANDI to CCR	–	16(2/0)	–	–	–
ANDI to SR	–	16(2/0)	–	–	–
CHK	–	8(1/0)+	–	–	–
EORI to CCR	–	16(2/0)	–	–	–
EORI to SR	–	16(2/0)	–	–	–
EXG	–	6(1/0)	–	–	–
EXT	word	4(1/0)	–	–	–
	long	4(1/0)	–	–	–
LINK	–	16(2/2)	–	–	–
MOVE from CCR	–	4(1/0)	8(1/1)+*	–	–
MOVE to CCR	–	12(2/0)	12(2/0)+	–	–
MOVE from SR	–	4(1/0)	8(1/1)+*	–	–
MOVE to SR	–	12(2/0)	12(2/0)+	–	–
MOVE from USP	–	6(1/0)	–	–	–
MOVE to USP	–	6(1/0)	–	–	–
MOVEC	–	–	–	10(2/0)	12(2/0)
MOVEP	word	–	–	16(2/2)	16(4/0)
	long	–	–	24(2/4)	24(6/0)
MOVES	byte, word	–	–	16(2/1) + *	16(3/0) + *
	long	–	–	20(2/2) + *	20(4/0) + *
NOP	–	4(1/0)	–	–	–
ORI to CCR	–	16(2/0)	–	–	–
ORI to SR	–	16(2/0)	–	–	–
RESET	–	130(1/0)	–	–	–
RTD	–	16(4/0)	–	–	–
RTE	short	24(6/0)	–	–	–
	long, retry read	112(27/10)	–	–	–
	long, retry write	112(26/1)	–	–	–
	long, no retry	110(26/0)	–	–	–
RTR	–	20(5/0)	–	–	–
RTS	–	16(4/0)	–	–	–
STOP	–	4(0/0)	–	–	–
SWAP	–	4(1/0)	–	–	–
TRAPV	–	4(1/0)	–	–	–
UNLK	–	12(3/0)	–	–	–

+ add effective address calculation time.
* use non-fetching effective address calculation time.
** Source or destination is a memory location for the MOVEP and MOVES instruction and a control register for the MOVEC instruction.

ABSOLUTE MAXIMUM RATINGS

RATING	SYMBOL	VALUE	UNIT
Supply voltage	V_{CC}	−0.3 to +7.0	V
Input voltage	V_{in}	−0.3 to +7.0	V
Operating temperature range SCN68010 SCN68010 ceramic	T_A	T_L to T_H 0 to 70 −40 to 85	°C
Storage temperature	T_{stg}	−55 to 150	°C

NOTE:
This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

THERMAL CHARACTERISTICS

CHARACTERISTIC	SYMBOL	VALUE	SYMBOL	VALUE	RATING
Thermal Resistance (Still Air)	θ_{JA}		θ_{JC}		°C/W
Ceramic DIP		30		15*	
Pin grid array		33		15	
Plastic DIP		30		15*	
PLCC		45*		25*	

*Estimated

Table 44. MAXIMUM POWER DISSIPATION BY PACKAGE TYPE MODES

PACKAGE TYPE	TEMPERATURE (°C)	MAXIMUM POWER DISSIPATION (WATTS) PER FREQUENCY (MHz)		
		8 MHz	10 MHz	12.5 MHz
Ceramic DIP	0 to 70	1.50	1.50	1.75
	−40 to 85	1.65	1.65	–
Plastic DIP	0 to 70	1.50	1.50	–
	−40 to 85	1.65	1.65	–
PGA	0 to 70	1.50	1.50	1.75
	−40 to 85	1.65	1.65	–
PLCC	0 to 70	1.50	1.50	–

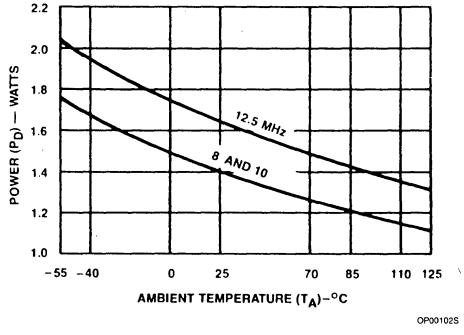


Figure 48. SCN68010 Power Dissipation (P_D) vs Ambient Temperature (T_A)

Power Considerations

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \theta_{JA}) \quad (1)$$

Where:

T_A = ambient temperature, °C
θ_{JA} = package thermal resistance, junction-to-ambient, °C/W
$P_D = P_{INT} + P_{I/O}$
$P_{INT} = I_{CC} \times V_{CC}$, watts — chip internal power
$P_{I/O}$ = power dissipation on input and output pins — user determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = T_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

The curve shown in figure 48 gives the graphic solution to these equations for the specification power dissipation of 1.50 and 1.75 watts over the ambient temperature range of −55°C to 125°C using a θ_{JA} of 45°C/W for the ceramic package.

The total thermal resistance of a package (θ_{JA}) can be separated into two components, θ_{JC} and θ_{CA}, representing the barrier to heat flow from the semiconductor junction to the package (case) surface (θ_{JC}) and from the case to the outside ambient (θ_{CA}). These terms are related by the equation:

$$\theta_{JA} = \theta_{JC} + \theta_{CA} \quad (4)$$

θ_{JC} is device related and cannot be influenced by the user. However, θ_{CA} is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling and thermal convention. Thus, good thermal management on the part of the user can significantly reduce θ_{CA} so that θ_{JA} approximately equals θ_{JC}. Substitution of θ_{JC} for θ_{JA} in equation (1) will result in a lower semiconductor junction temperature.

Values for thermal resistance presented in this data sheet, unless estimated, are provided for design purposes only. Thermal measurements are complex and dependent on procedure and set-up. User derived values for thermal resistance may differ.

2

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0Vdc ±5%; GND = 0Vdc; $T_A = T_L$ to T_H (see figures 49, 50, and 51)

CHARACTERISTIC		SYMBOL	MIN	MAX	UNIT
Input high voltage		V_{IH}	2.0	V_{CC}	V
Input low voltage		V_{IL}	GND – 0.3	0.8	V
Input leakage current @ 5.25V	$\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$, CLK, $\overline{IPL0}$ – $\overline{IPL2}$, $\overline{VPA}$	I_{IN}	–	2.5	μA
	$\overline{HALT}$, $\overline{RESET}$			20	
Three-state (off state) input current @ 2.4V/0.4V	$\overline{AS}$, A1 – A23, D0 – D15, FC0 – FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	I_{TSI}	–	20	μA
Output high voltage (I_{OH} = –400μA)	E*	V_{OH}	V_{CC} – 0.75	–	V
	E**, $\overline{AS}$, A1 – A23, $\overline{BG}$, D0 – D15, FC0 – FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$		2.4	–	
Output low voltage					
(I_{OL} = 1.6mA)	$\overline{HALT}$	V_{OL}	–	0.5	V
(I_{OL} = 3.2mA)	A1 – A23, $\overline{BG}$, FC0 – FC2		–	0.5	
(I_{OL} = 5.0mA)	$\overline{RESET}$		–	0.5	
(I_{OL} = 5.3mA)	E, $\overline{AS}$, D0 – D7, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$		–	0.5	
Power dissipation (see Power Considerations)***			P_D	–	–
Capacitance (V_{IN} = 0V, T_A = 25°C, frequency = 1MHz)****		C_{in}	–	20.0	pF

* With external pullup resistor of 1.1 kΩ.
** Without external pullup resistor.
*** During normal operation instantaneous V_{CC} current requirements may be as high as 1.5A.
****Capacitance is periodically sampled rather than 100% tested.

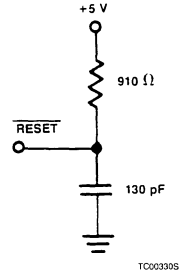


Figure 49. $\overline{RESET}$ Test Load

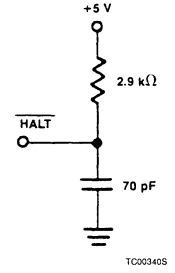


Figure 50. Test Loads

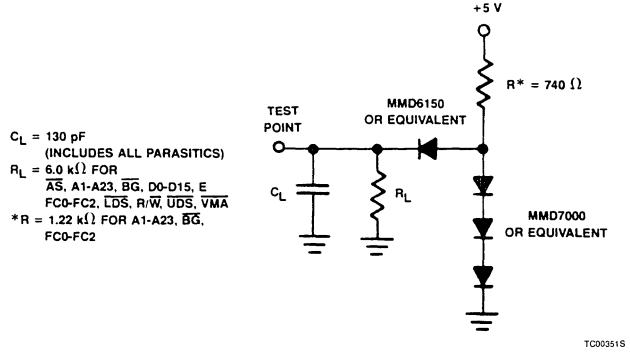


Figure 51. $\overline{HALT}$ Test Load

AC ELECTRICAL CHARACTERISTICS — Clock Input (See figure 52)

CHARACTERISTIC	SYMBOL	8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	
Frequency of operation	f	4.0	8.0	4.0	10.0	4.0	12.5	MHz
Cycle time	t_{cyc}	125	250	100	250	80	250	ns
Clock pulse width	t_{CL} t_{CH}	55 55	125 125	45 45	125 125	35 35	125 125	ns
Rise and fall times	t_{Cr} t_{Cf}	– –	10 10	– –	10 10		5 5	ns

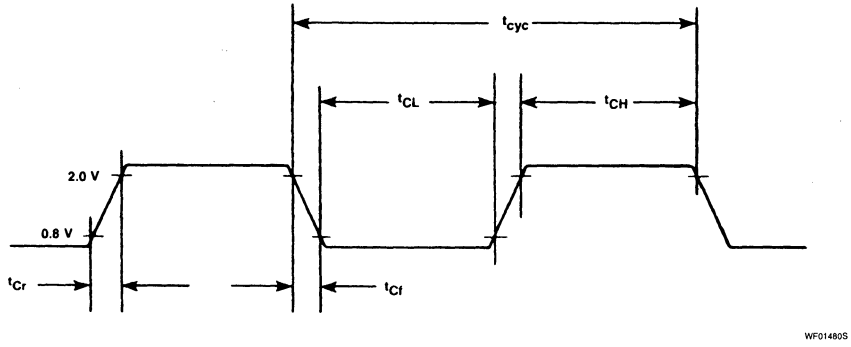


Figure 52. Clock Input Timing Diagram

2

AC ELECTRICAL CHARACTERISTICS — Read and Write Cycles (V_{CC} = 5.0Vdc ±5%; GND = 0Vdc; $T_A = T_L$ to T_H see figures 53 and 54)

NO.	CHARACTERISTIC	8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	
1	Clock period	125	500	100	250	80	250	ns
2	Clock width low	55	125	45	125	35	125	ns
3	Clock width high	55	125	45	125	35	125	ns
4	Clock fall time	–	10	–	10	–	5	ns
5	Clock rise time	–	10	–	10	–	5	ns
6	Clock low to address valid	–	70	–	55	–	55	ns
6A	Clock high to FC valid	–	70	–	55	–	55	ns
7	Clock high to address, data bus high impedance (maximum)	–	80	–	70	–	60	ns
8	Clock high to address, FC invalid (minimum)	0	–	0	–	0	–	ns
9[1]	Clock high to $\overline{AS}$, $\overline{DS}$ low	0	60	0	55	0	55	ns
11[2]	Address valid to $\overline{AS}$, $\overline{DS}$ low (read)/$\overline{AS}$ low (write)	30	–	20	–	0	–	ns
11A[2]	FC valid to $\overline{AS}$, $\overline{DS}$ low (read)/$\overline{AS}$ low (write)	60	–	50	–	40	–	ns
12[1]	Clock low to $\overline{AS}$, $\overline{DS}$ high	–	70	–	55	–	50	ns
13[2]	$\overline{AS}$, $\overline{DS}$ high to address/FC invalid	30	–	20	–	10	–	ns
14[2]	$\overline{AS}$, $\overline{DS}$ width low (read)/$\overline{AS}$ low (write)	240	–	195	–	160	–	ns
14A[2]	$\overline{DS}$ width low (write)	–	115	95	–	80	–	ns
15[2]	$\overline{AS}$, $\overline{DS}$ width high	150	–	105	–	65	–	ns
16	Clock high to control bus high impedance	–	80	–	70	–	60	ns
17[2]	$\overline{AS}$, $\overline{DS}$ high to R/$\overline{W}$ high (read)	40	–	20	–	10	–	ns
18[1]	Clock high to R/$\overline{W}$ high	0	70	0	60	0	60	ns
20[1]	Clock high to R/$\overline{W}$ low	–	70	–	60	–	60	ns
20A[2]	$\overline{AS}$ low to R/$\overline{W}$ valid (write)	–	20	–	20	–	20	ns
21[2]	Address valid to R/$\overline{W}$ low (write)	20	–	0	–	0	–	ns
21A[2]	FC valid to R/$\overline{W}$ low (write)	60	–	50	–	30	–	ns
22[2]	R/$\overline{W}$ low to $\overline{DS}$ low (write)	80	–	50	–	30	–	ns
23	Clock low to data out valid (write)	–	70	–	55	–	55	ns
25[2]	$\overline{AS}$, $\overline{DS}$ high to data out invalid (write)	30	–	20	–	15	–	ns
26[2]	Data out valid to $\overline{DS}$ low (write)	30	–	20	–	15	–	ns
27[5]	Data in to clock low (set-up time on read)	15	–	10	–	10	–	ns
27A	Late $\overline{BERR}$ low to clock low (set-up time)	45	–	45	–	45	–	ns
28[2]	$\overline{AS}$, $\overline{DS}$ high to $\overline{DTACK}$ high	0	245	0	190	0	150	ns
29	$\overline{AS}$, $\overline{DS}$ high to data invalid (hold time on read)	0	–	0	–	0	–	ns
30	$\overline{AS}$, $\overline{DS}$ high to $\overline{BERR}$ high	0	–	0	–	0	–	ns
31[2,5]	$\overline{DTACK}$ low to data valid (set-up time)	–	90	–	65	–	50	ns
32	$\overline{HALT}$ and $\overline{RESET}$ input transition time	0	200	0	200	0	200	ns
33	Clock high to $\overline{BG}$ low	–	70	–	60	–	50	ns
34	Clock high to $\overline{BG}$ high	–	70	–	60	–	50	ns
35	$\overline{BR}$ low to $\overline{BG}$ low	1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	clk. per.

AC ELECTRICAL CHARACTERISTICS (Continued)

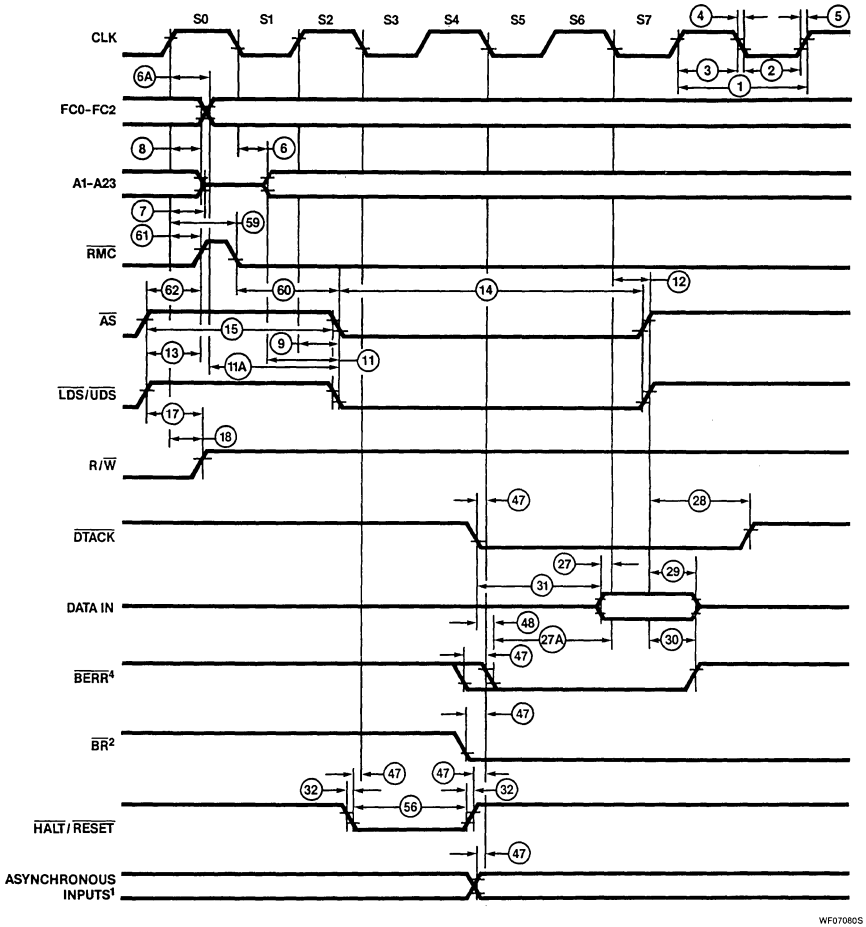
NO.	CHARACTERISTIC	8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	
36[6]	$\overline{BR}$ high to $\overline{BG}$ high	1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	clk. per.
37	$\overline{BGACK}$ low to $\overline{BG}$ high	1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	clk. per.
37A[7]	$\overline{BGACK}$ low to $\overline{BR}$ high	20	1.5 clocks	20	1.5 clocks	20	1.5 clocks	ns
38	$\overline{BG}$ low to control, address, data bus high impedance ($\overline{AS}$ high)	–	80	–	70	–	60	ns
39	$\overline{BG}$ width high	1.5	–	1.5	–	1.5	–	clk. per.
40	Clock low to $\overline{VMA}$ low	–	70	–	70	–	70	ns
41	Clock low to E transition	–	70	–	55	–	45	ns
42	E output rise and fall time	–	25	–	25	–	25	ns
43	$\overline{VMA}$ low to E high	200	–	150	–	90	–	ns
44	$\overline{AS}$, $\overline{DS}$ high to $\overline{VPA}$ high	0	120	0	90	0	70	ns
45	E low to control address bus invalid (address hold time)	30	–	10	–	10	–	ns
46	$\overline{BGACK}$ width	1.5	–	1.5	–	1.5	–	clk. per .
47[5]	Asynchronous input set-up time	20	–	20	–	20	–	ns
48[2,3]	$\overline{DTACK}$ low to $\overline{BERR}$ low	–	80	–	55	–	35	ns
49[8]	$\overline{AS}$, $\overline{DS}$ high to E low	–70	70	–55	55	–45	45	ns
50	E width high	450	–	350	–	280	–	ns
51	E width low	700	–	550	–	440	–	ns
53	Clock high to data out valid	0	–	0	–	0	–	ns
54	E low to data out valid	30	–	20	–	15	–	ns
55	R/$\overline{W}$ to data bus driven	30	–	20	–	10	–	ns
56[4]	$\overline{HALT}$/$\overline{RESET}$ pulse width	10	–	10	–	10	–	clk. per.
57	$\overline{BGACK}$ high to control bus driven	1.5	–	1.5	–	1.5	–	clk. per
58[6]	$\overline{BG}$ high to control bus driven	1.5	–	1.5	–	1.5	–	clk. per

NOTES:

1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the values given in these columns.
2. Actual value depends on clock period.
3. In the absence of $\overline{DTACK}$, $\overline{BERR}$ is an asynchronous input using the asynchronous input set-up time (#47).
4. For power up, the MPU must be held in $\overline{RESET}$ state for 100ms to allow stabilization of on-chip circuitry. After the system is powered up, #56 refers to the minimum pulse width required to reset the system.
5. If the asynchronous set-up time (#47) requirements are satisfied, the $\overline{DTACK}$-low to data set-up time (#31) and $\overline{DTACK}$–low to $\overline{BERR}$–low set-up time (#48) requirements can be ignored. The data must only satisfy the data-in to clock-low set-up time (#27) for the following clock cycle, $\overline{BERR}$ must only satisfy the late-$\overline{BERR}$-low to clock-low set-up time (#27A) for the following clock cycle.
6. The processor will negate $\overline{BG}$ and begin driving the bus again if external arbitration logic negates $\overline{BR}$ before asserting $\overline{BGACK}$.
7. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.
8. The falling edge of S6 triggers both the negation of the strobes ($\overline{AS}$ and $\overline{DS}$) and the falling edge of E. Either of these events can occur first, depending on the loading on each signal. Specification #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of the E clock.

2

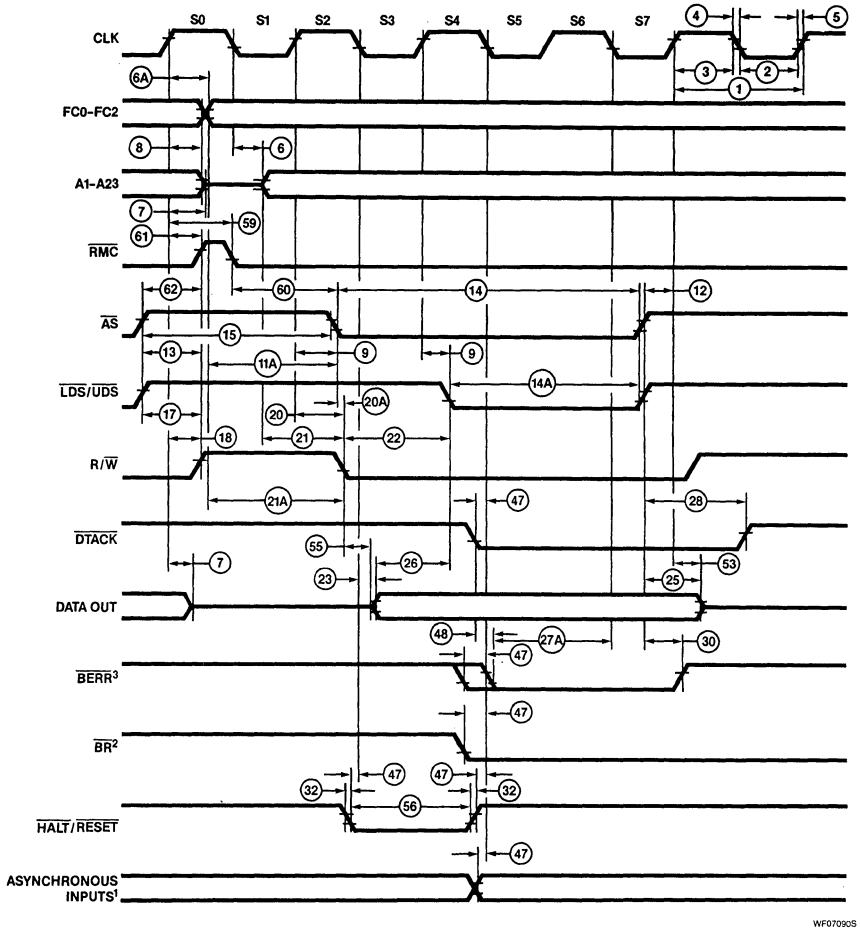
These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



NOTES:
1. Set-up time for the asynchronous inputs IPL0/2, IPL1, and VPA guarantees their recognition at the next falling edge of the clock.
2. BR need fall at this time only in order to insure being recognized at the end of this bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.
4. The timing for the first falling edge (47) of BERR are for BERR without DTACK, the timings for the second falling edge (27A and 48) are for BERR with DTACK.

Figure 53. Read Cycle Timing Diagram

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



NOTES:
1. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.
2. Because of loading variations, R/$\overline{W}$ may be valid after $\overline{AS}$ even though both are initiated by the rising edge of S2 (specification 20A).
3. The timing for the first falling edge (47) of $\overline{BERR}$ are for $\overline{BERR}$ without $\overline{DTACK}$; the timings for the second falling edge (27A and 48) are for $\overline{BERR}$ with $\overline{DTACK}$.

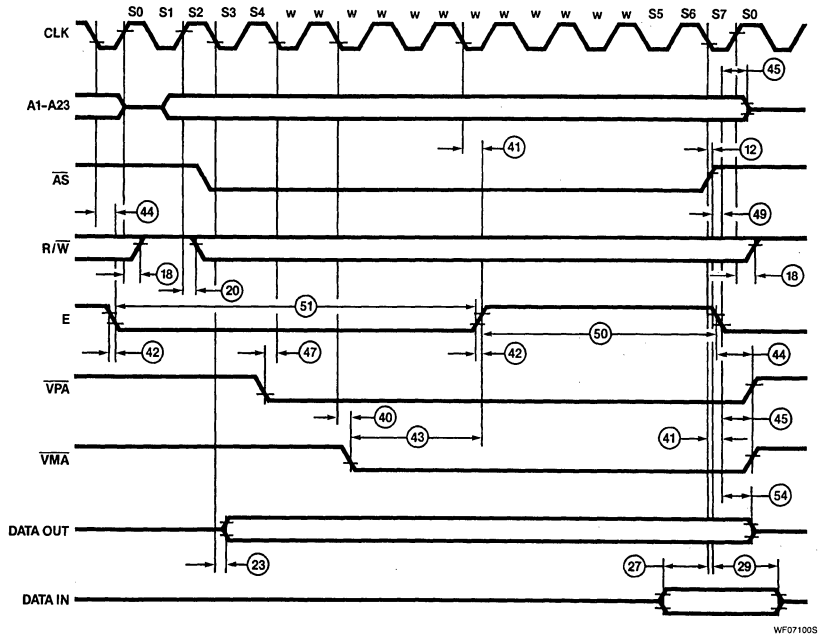
Figure 54. Write Cycle Timing Diagram

AC ELECTRICAL CHARACTERISTICS — SCN68010 to Synchronous Peripheral Cycles V_{CC} = 5.0Vdc ±5%, GND = 0Vdc, $T_A = T_L$ to T_H (refer to figures 55 and 56)

NO.	CHARACTERISTIC	8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	
12[1]	Clock low to $\overline{AS}$, $\overline{DS}$ high	–	70	–	55	–	50	ns
17[2]	$\overline{AS}$, $\overline{DS}$ high to R/$\overline{W}$ high (read)	40	–	20	–	10	–	ns
20[1]	Clock high to R/$\overline{W}$ low	–	70	–	60	–	60	ns
23	Clock low to data out valid (write)	–	70	–	55	–	55	ns
27	Data in to clock low (set-up time on read)	15	–	10	–	10	–	ns
40	Clock low to $\overline{VMA}$ low	–	70	–	70	–	70	ns
41	Clock low to E transition	–	70	–	55	–	45	ns
42	E output rise and fall time	–	25	–	25	–	25	ns
43	$\overline{VMA}$ low to E high	200	–	150	–	90	–	ns
44	$\overline{AS}$, $\overline{DS}$ high to $\overline{VPA}$ high	0	120	0	90	0	70	ns
45	E low to control address bus (address hold time)	30	–	10	–	10	–	ns
47	Asynchronous input set-up time	20	–	20	–	20	–	ns
49[3]	$\overline{AS}$, $\overline{DS}$ high to E low	−70	70	−55	55	−45	45	ns
50	E width high	450	–	350	–	280	–	ns
51	E width low	700	–	550	–	440	–	ns
54	E low to data out invalid	30	–	20	–	15	–	ns

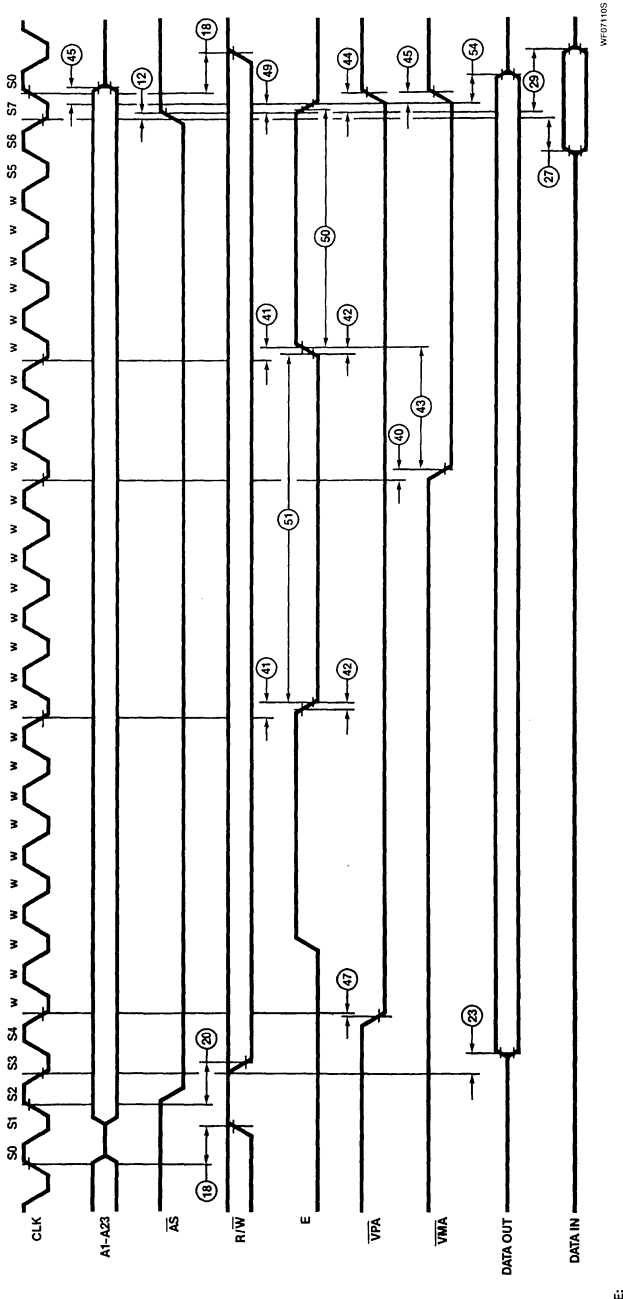
NOTES:
1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the values given in these columns.
2. Actual value depends on clock period.
3. The falling edge of S6 triggers both the negation of the strobes ($\overline{AS}$ and $\overline{DS}$) and the falling edge of E. Either of these events can occur first, depending on the loading on each signal. Specification #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of the E clock.

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



NOTE:
This timing diagram is included for those who wish to design their own timing circuit to generate $\overline{VMA}$. It shows the best case possibly attainable.

Figure 55. SCN68010 to Synchronous Peripheral Timing Diagram — Best Case



NOTE:
This timing diagram is included for those who wish to design their own circuit to generate VMA. It shows the worst case possibly attainable.

Figure 56. SCN68010 to Synchronous Peripheral Timing Diagram — Worst Case

AC ELECTRICAL CHARACTERISTICS — Bus Arbitration V_{CC}=5.0Vdc ±5%; GND = 0Vdc; $T_A = T_L$ to T_H (see figures 57, 58, 59)

NO.	CHARACTERISTIC	8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	
7	Clock high to address, data bus high impedance (maximum)	–	80	–	70	–	60	ns
16	Clock high to control bus high impedance	–	80	–	70	–	60	ns
33	Clock high to $\overline{BG}$ low	–	70	–	60	–	50	ns
34	Clock high to $\overline{BG}$ high	–	70	–	60	–	50	ns
35	$\overline{BR}$ low to $\overline{BG}$ low	1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	clk. per.
36[2]	$\overline{BR}$ high to $\overline{BG}$ high	1.5	90ns +3.5	1.5	80ns +3.5	1.5	70ns +3.5	clk. per.
37	$\overline{BGACK}$ low to $\overline{BG}$ high	1.5	90ns 3.5	1.5	80ns 3.5	1.5	70ns 3.5	clk. per.
37A[3]	$\overline{BGACK}$ low to $\overline{BR}$ high	20	1.5 Clocks	20	1.5 Clocks	20	1.5 Clocks	ns
38	$\overline{BG}$ low to control, address, data bus high impedance ($\overline{AS}$ high)	–	80	–	70	–	60	ns
39	$\overline{BG}$ width high	1.5	–	1.5	–	1.5	–	clk. per
46	$\overline{BGACK}$ width	1.5	–	1.5	–	1.5	–	clk. per.
47	Asynchronous input set-up time	20	–	20	–	20	–	ns
57[2]	$\overline{BGACK}$ high to control bus driven	1.5	–	1.5	–	1.5	–	clk. per.
58[1,2]	$\overline{BG}$ high to control bus driven	1.5	–	1.5	–	1.5	–	clk. per.

NOTES:
1. The nanosecond value shown in the specification is the asynchronous input set-up time (spec. #47).
2. The processor will negate $\overline{BG}$ and begin driving the bus again if external arbitration logic negates $\overline{BR}$ before asserting $\overline{BGACK}$.
3. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

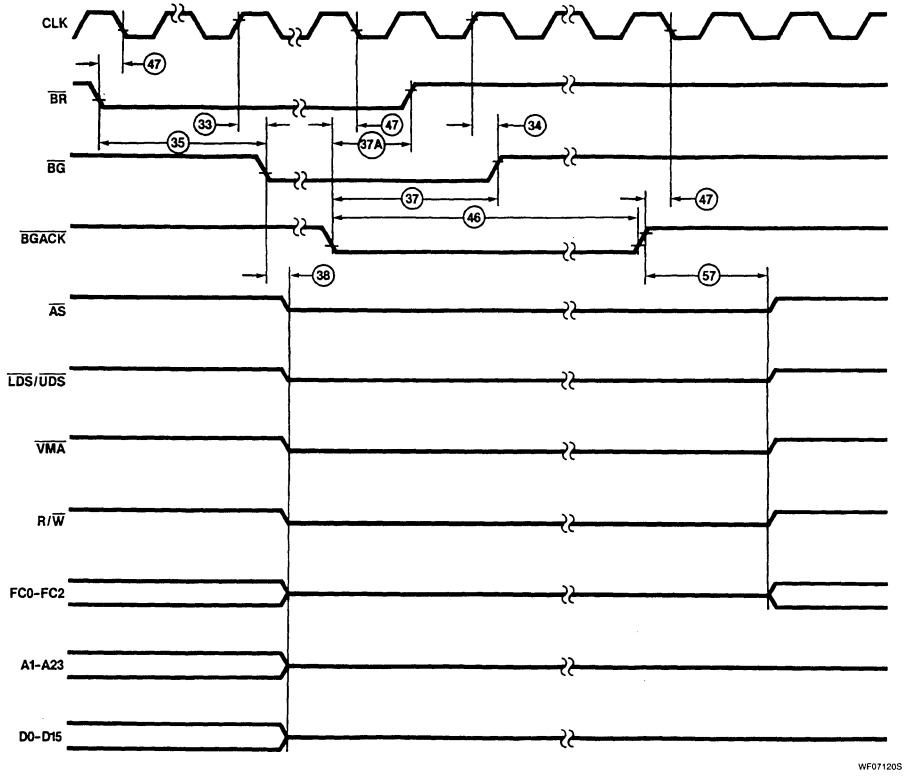


Figure 57. Bus Arbitration Timing — Idle Bus Case

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

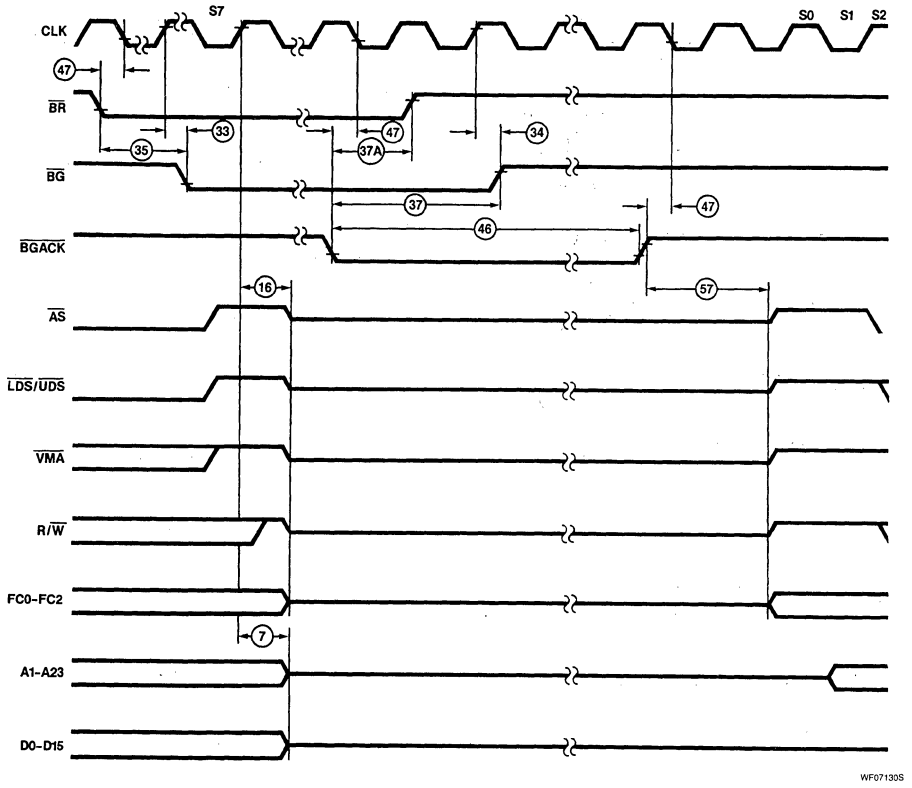


Figure 58. Bus Arbitration Timing — Active Bus Case

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

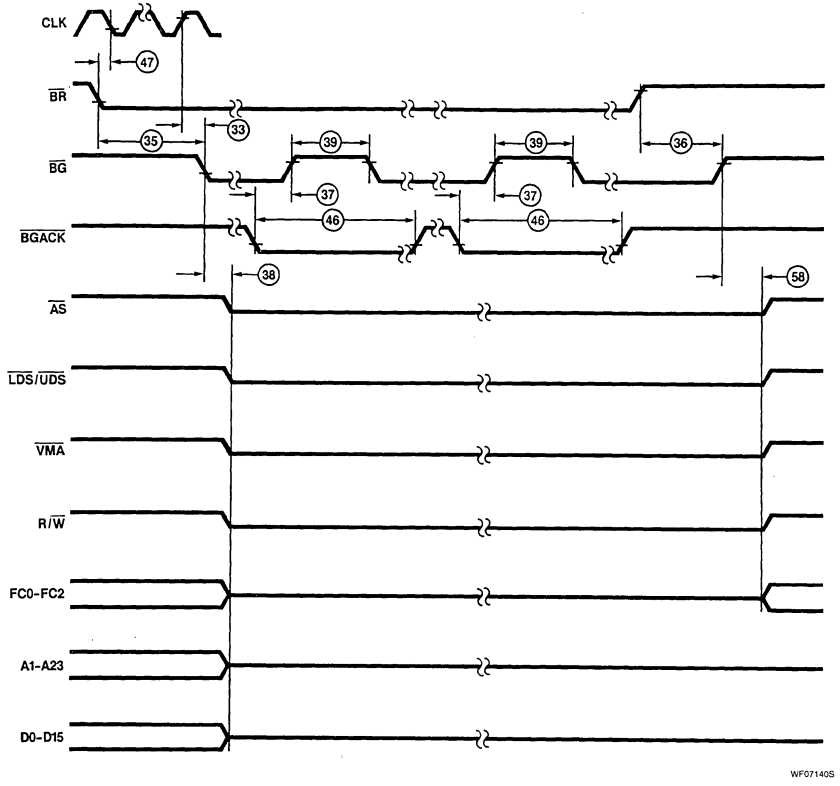


Figure 59. Bus Arbitration Timing — Multiple Bus Requests

Signetics

SCB68154
Interrupt Generator

Preliminary Specification

Microprocessor Products

DESCRIPTION

The Signetics SCB68154/8X825 Interrupt Generator provides an interface between an interrupting device and a system bus such as the VMEbus or VERSAbus™. Figure 1 shows a typical configuration of the SCB68154/8X825. The SCB68154/8X825 has three primary functions:

1. Generates bus interrupt requests.
2. Resides in the interrupt acknowledge daisy-chain.
3. Allows a status/ID byte (interrupt vector) to be supplied to the system if needed.

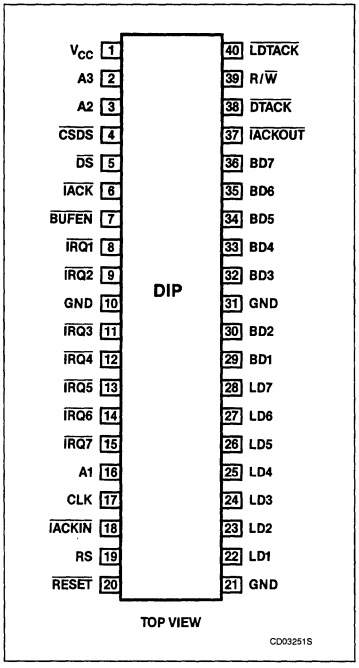
The SCB68154/8X825 has seven interrupt request levels, $\overline{IRQ7}$ – $\overline{IRQ1}$, which are selected by using the interrupt request register. The local master writes to the interrupt request register to generate an interrupt request on any interrupt request level. The interrupt request register may be read to determine if an interrupt has been acknowledged. If a level with an interrupt request pending is acknowledged, the SCB68154/8X825 will allow a status/ID byte to be supplied to the system. Seven bits of the status/ID byte come from the interrupt vector register with the user externally supplying the LSB. If the SCB68154/8X825 does not have an interrupt on the level acknowledged, the SCB68154/8X825 will pass the interrupt acknowledge on via the interrupt acknowledge daisy-chain output. The user can enable all interrupt request levels and clear all interrupt request levels by setting specific bits in the interrupt vector register.

The SCB68154/8X825 was designed primarily for interface to the VMEbus. For more information regarding the protocol definitions, proper use, and application of this device, refer to the VMEbus Specification Manual.

FEATURES

- **Interrupts generator for VMEbus and VERSAbus systems**
- **Generates 7 bus interrupt requests**
- **Two internal registers for system control**
- **Interrupt enable and interrupt clear bits**
- **Allows status/ID byte to be supplied during interrupt acknowledge**
- **High speed bipolar technology**
- **Single +5V supply**

PIN CONFIGURATION



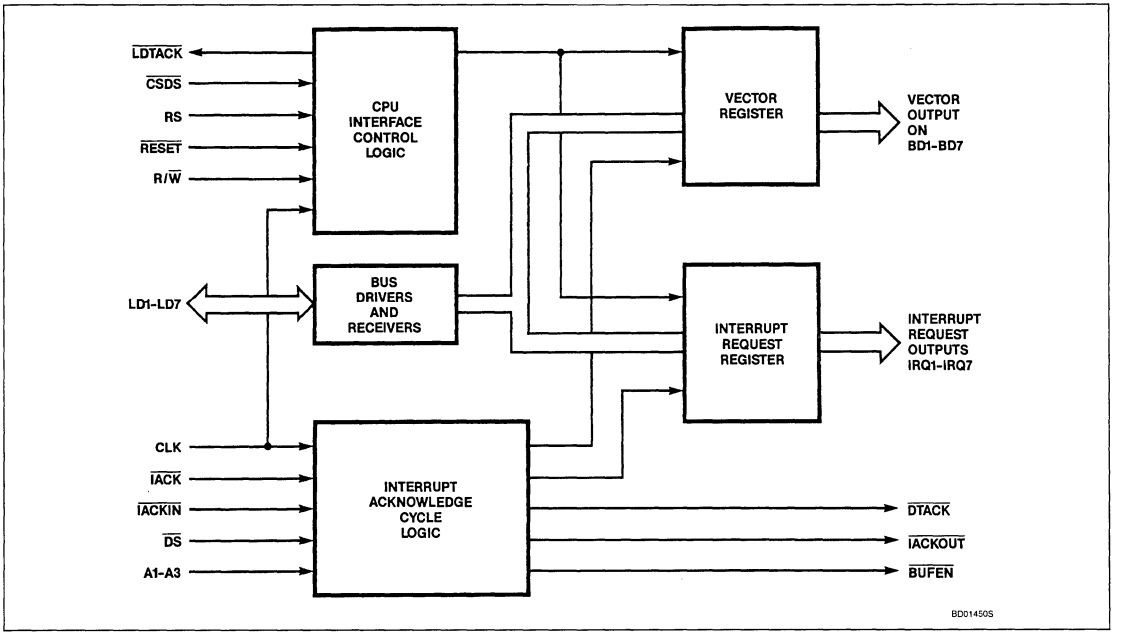
VERSAbus is a registered trademark of Motorola, Inc.

Interrupt Generator SCB68154

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C
Ceramic DIP	SCB68154C2I40
Plastic DIP	SCB68154C2N40

BLOCK DIAGRAM



Interrupt Generator | SCB68154

PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{CC}	1	I	**Supply Voltage:** + 5V power supply
A1 – A3	16, 3, 2	I	**Address Lines:** Address inputs from system bus. The internal level being acknowledged is encoded on these inputs. A1 is LSB.
$\overline{CSDS}$	4	I	**Chip Select:** Active low chip select input for register I/O. This input must be qualified by the local master's data strobe prior to input (see figure 1).
$\overline{DS}$	5	I	**Data Strobe:** Active low data strobe input from the system used to enable interrupt vector output.
$\overline{IACK}$	6	I	**Interrupt Acknowledge:** Active low interrupt acknowledge input from the system bus.
$\overline{BUFEN}$	7	O	**Buffer Enable:** Active low totem pole output to enable the data buffer required to drive the outputs of the bus data pins (BD1 – BD7).
$\overline{IRQ1}$ – $\overline{IRQ7}$	8, 9, 11 – 15	O	**Interrupt Request:** Active low totem pole system interrupt request output.
GND	10, 21, 31	I	**Ground**
CLK	17	I	**Clock:** Clock input (typically CPU clock).
$\overline{IACKIN}$	18	I	**Interrupt Acknowledge In:** Active low interrupt acknowledge daisy-chain input.
RS	19	I	**Register Select:** Register select input.
$\overline{RESET}$	20	I	**Reset:** Active low input resets all internal registers, $\overline{IACKOUT}$, and $\overline{IRQn}$.
LD1 – LD7	22 – 28	I/O	**Local Data:** Three state local data bus.
BD1 – BD7	29, 30, 32 – 36	O	**Bus Data:** Three state data pins used for vector output.
$\overline{IACKOUT}$	37	O	**Interrupt Acknowledge Out:** Active low totem pole interrupt acknowledge daisy-chain output.
$\overline{DTACK}$	38	O	**Data Transfer Acknowledge:** Active low, totem pole output. This signal indicates that valid data is available on the bus during interrupt acknowledge cycle.
R/$\overline{W}$	39	I	**Read/Write:** Register read/write input. This signal specifies the data transfer cycle in process is to be either read or write.
$\overline{LDTACK}$	40	O	**Local Data Transfer Acknowledge:** Active low, open collector, data transfer acknowledge output to the local bus.

FUNCTIONAL DESCRIPTION

Typical Configuration

The SCB68154/8X825 provides a vehicle for interprocessor communications on an intelligent peripheral controller board, or a CPU board as shown in Figure 1. The local data pins (LD1 – LD7) serve as a local data bus. This allows the local master to access the Interrupt Generator's two internal registers. During an interrupt acknowledge, the SCB68154/8X825 will allow for a status/ID byte to be supplied to the system. The SCB68154/8X825 supplies seven of the eight needed status/ID bits. The user is allowed to externally supply the least significant bit (LSB), typically the system address line A1 of the status/ID byte. The $\overline{IRQ1}$ – $\overline{IRQ7}$, $\overline{DTACK}$, and BD1 – BD7 outputs require external buffers to provide adequate drive to the system bus. $\overline{BUFEN}$ provides the output enable control for the data buffer that is required for BD1 – BD7.

Register Selection

The SCB68154/8X825 has two internal registers which can be programmed for system control. They are the interrupt vector register and the interrupt request register. Figure 2 shows the programming model. Both registers can be read from as well as written to. The interrupt vector register (register R0) is selected by the register select (RS) input equal to 0. Setting bit 1 of register R0 enables all interrupt levels for the SCB68154/8X825. Writing a 1 to bit 2 of register R0 resets all interrupt levels in the interrupt request register as well as the $\overline{IRQn}$ outputs. Subsequent interrupt requests will be honored. Bit 2 of R0 will always be read as 0. The high order bits, bits 7 – 3, of register R0 are the high order bits of the status/ID byte. The seven bit output of the status/ID byte are formed by concatenating the high order bits (bits 7 – 3) of register R0 with system bus address lines A3 and A2. Bus address lines A3 and A2 are output on BD2 and BD1 respectively.

The interrupt request register (R1) is selected by RS input equal to 1. Setting bit "n" in R1 will generate an interrupt on interrupt request level $\overline{IRQn}$. Any number (up to the maximum of seven) interrupt requests can be generated in a single access of R1.

The state of only those levels, which a 1 has been written to, is affected. Writing a 0 to any level does not change the current state of that level. For example, if $\overline{IRQ1}$ is currently asserted, writing a 0 to bit 1 of R1 does not de-assert $\overline{IRQ1}$, nor clear bit 1 in R1.

Note that interrupt requests on the same level are not stackable. To generate another interrupt request on a level currently asserted, the user must wait until that level has been acknowledged, before generating another interrupt request on that level.

Since interrupts are acknowledged independently of the local CPU, the interrupt vector register should not be modified while interrupts are pending. Any attempt to modify the interrupt vector register, while interrupts are

pending, could cause the status/ID byte to change while the interrupted master is reading it. This could cause the interrupted master to acquire an indeterminate vector. Therefore, the interrupt request register should be examined to make certain there are no interrupts pending before attempting to modify the interrupt vector register.

All data transfers between the local CPU and the SCB68154/8X825 are done using local data lines (LD1 – LD7), register select (RS), read/write (R/$\overline{W}$) and a chip select input ($\overline{CSDS}$). The SCB68154/8X825 supplies a local data transfer acknowledge ($\overline{LDTACK}$) to complete the transfer of data between the local CPU and itself.

Interrupt/Interrupt Acknowledge

The SCB68154/8X825 generates the maximum defined seven bus interrupts, on the $\overline{IRQ1}$ – $\overline{IRQ7}$ outputs, in a VMEbus or VERSAbus system. An interrupted master will acknowledge only a single level of the seven interrupt levels. To allow for multiple interrupters on the level acknowledged, VMEbus and VERSAbus systems use an interrupt acknowledge daisy-chain. The SCB68154/8X825 resides in this interrupt acknowledge daisy-chain.

When the system interrupt acknowledge ($\overline{IACK}$) is asserted, the interrupt acknowledge daisy-chain starts at the first slot in the system bus. The level being acknowledged is specified by the interrupted master on address lines A1 – A3. The system bus interface on the SCB68154/8X825 is initiated only if $\overline{IACK}$, interrupt acknowledge daisy-chain in ($\overline{IACKIN}$), and data strobe 0 (DS0) are all received asserted. If the system bus interface is initiated and the SCB68154/8X825 has an interrupt request on the level specified, it will not pass the daisy-chain signal on. It will, instead, clear the interrupt request level acknowledged, $\overline{IRQn}$, as well as the appropriate bit in the interrupt request register. It will also assert buffer enable ($\overline{BUFEN}$), place seven bits of the status/ID byte on the bus data outputs (BD1 – BD7) and assert data transfer acknowledge ($\overline{DTACK}$).

If the system bus interface is initiated, but the SCB68154/8X825 has no interrupt request on the level being acknowledged, it will pass the daisy-chain input ($\overline{IACKIN}$) on via the interrupt acknowledge out ($\overline{IACKOUT}$).

Arbitration

The system bus interface, as well as the local master interface, are independent processes. Either can be initiated at any time, without respect to the other. The SCB68154/8X825 will arbitrate between the processes, allowing proper system operation without any degradation in performance.

Reset

When $\overline{RESET}$ is asserted, the SCB68154/8X825 drives the outputs $\overline{IRQn}$ and $\overline{IACKOUT}$ high. It also resets the interrupt request and the interrupt vector registers.

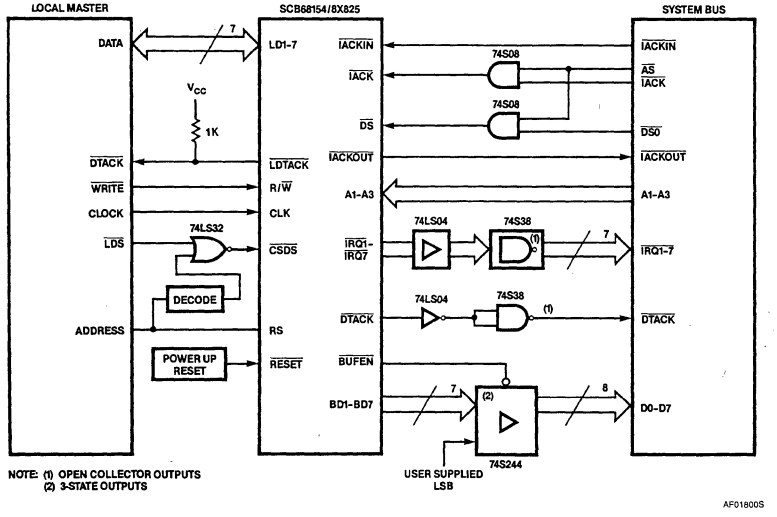


Figure 1. Typical Configuration

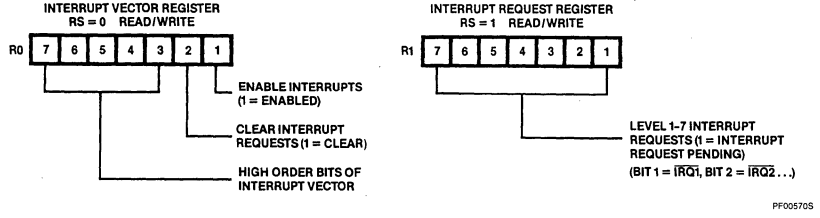


Figure 2. Programming Model

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
Supply voltage[3]	−0.5 to +7.0	V
Input voltage[3]	−0.5 to +5.5	V
Voltage applied to output in off-state[3]	−0.5 to +5.5	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{CC}	Supply voltage		4.75	5.25	V
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
I_{IL}	Input low current	$V_{CC} = 5.25V$, $V_{IL} = 0.4V$		−410	μA
I_{IH}	Input high current	$V_{CC} = 5.25V$, $V_{IH} = 2.7V$		20	μA
I_{OS}	Short circuit output current	$V_{CC} = 5.25V$, $V_{OUT} = 0V$ Other outputs not grounded	−15	−100	mA
I_{OZL}	High-Z low output current BD1 – BD7	$V_{CC} = 5.25V$, $V_{OL} = 0.5V$		−20	μA
I_{OZH}	High-Z high output current BD1 – BD7	$V_{CC} = 5.25V$, $V_{OH} = 2.5V$		20	μA
V_{OL}	Output low voltage			0.5	V
	$\overline{LDTACK}$	$V_{CC} = 4.75V$, $I_{OL} = 20mA$			
	All other outputs	$V_{CC} = 4.75V$, $I_{OL} = 8mA$			
V_{OH}	Output high voltage	$V_{CC} = 4.75V$, $I_{OH} = -400\mu A$	2.5		V
I_{CEX}	Open collector leakage current				
	$\overline{LDTACK}$	$V_{CC} = 4.75V$, $V_{OUT} = 4.75V$		100	μA
I_I	Input leakage current	$V_{CC} = 5.25V$, $V_{IN} = 5.25V$		100	μA
V_{IC}	Input clamp voltage	$V_{CC} = 4.75V$, $I_{IN} = -10mA$	−1.2		V
I_{CC}	Supply current	$V_{CC} = 5.25V$, $V_{IN} = 0V$		130	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on + 150°C maximum junction temperature and thermal resistance of 60°C/W junction to ambient for ceramic package (116°C/W for plastic package).
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all inputs swing between 0.4V and 2.4V with a transition time of 3ns maximum and output voltages are checked at 0.8V and 2.0V.
6. If the falling edge of $\overline{IACKIN}$ occurs last, then t_{ADIK} is valid. If the falling edge of $\overline{DS}$ occurs last, then t_{ADDS} is valid. If the falling edges of $\overline{IACKIN}$ and $\overline{DS}$ occur simultaneously then either t_{ADDS} or t_{ADIK} is valid.
7. If the falling edge of $\overline{IACKIN}$ occurs last, then t_{IKBF} is valid. If the falling edge of $\overline{DS}$ occurs last, then t_{DSBF} is valid. If the falling edges of $\overline{IACKIN}$ and $\overline{DS}$ occur simultaneously then either t_{IKBF} or t_{DSBF} is valid.
8. If the falling edge of $\overline{IACKIN}$ occurs last, then t_{IRA1} is valid. If $\overline{DS}$ occurs last, then t_{IRA2} is valid. If the falling edges of both $\overline{IACKIN}$ and $\overline{DS}$ occur simultaneously then either t_{IRA1} or t_{IRA2} is valid.
9. True only if no request pending on levels being acknowledged. If the falling edge of $\overline{IACKIN}$ occurs last, then t_{DYO1} is valid. If the falling edge of $\overline{DS}$ occurs last then t_{DYO2} is valid. If the falling edges of both $\overline{IACKIN}$ and $\overline{DS}$ occur simultaneously then either t_{DYO1} or t_{DYO2} is valid.
10. If the rising edge of $\overline{IACK}$ occurs first, then t_{IDT} is valid. If the rising edge of $\overline{DS}$ occurs first then t_{DSDT} is valid. If the rising edges of both $\overline{IACK}$ and $\overline{DS}$ occur simultaneously, then either t_{IDT} or t_{DSDT} is valid.
11. If the rising edge of $\overline{IACK}$ occurs first then t_{ITST} is valid. If the rising edge of $\overline{DS}$ occurs first then t_{DTST} is valid. If the rising edges of both $\overline{IACK}$ and $\overline{DS}$ occur simultaneously then either t_{ITST} or t_{DTST} is valid.
12. If the rising edge of $\overline{IACK}$ occurs first, then t_{IBF} is valid. If the rising edge of $\overline{DS}$ occurs first then t_{DBF} is valid. If the rising edges of both $\overline{IACK}$ and $\overline{DS}$ occur simultaneously, then either t_{IBF} or t_{DBF} is valid.
13. If the rising edge of $\overline{IACK}$ occurs first then t_{IDTK} is valid. If the rising edge of $\overline{DS}$ occurs first then t_{DDTK} is valid. If the rising edges of both $\overline{IACK}$ and $\overline{DS}$ occur simultaneously, then either t_{IDTK} or t_{DDTK} is valid.
14. True only if no request pending on level being acknowledged. If the rising edge of $\overline{IACK}$ occurs first then t_{IOUT} is valid. If the rising edge of $\overline{DS}$ occurs first then t_{DOUT} is valid. If the rising edge of both $\overline{IACK}$ and $\overline{DS}$ occur simultaneously then either t_{IOUT} or t_{DOUT} is valid.
15. t_{TST} is always greater than or equal to t_{DTH}.
16. These parameters are guaranteed at the values listed; these values were determined either by system bench testing or by Signetics' characterization procedures. All other tabular entries are taken directly from simulation results run at a range of operational frequencies; these values are not tested or guaranteed.

2

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5]

PARAMETER		LIMITS		UNIT
		Min	Max	
Register read (see figure 3)				
t_{RSS}	RS valid to $\overline{CSDS}$ low set-up time	0		ns
t_{RWS}	R/$\overline{W}$ to $\overline{CSDS}$ low set-up time	0		ns
t_{DTV}	$\overline{LDTACK}$ low to LD1 – 7 valid	5.4	19.4	ns
t_{DTCS}	$\overline{LDTACK}$ low to $\overline{CSDS}$ high	0		ns
t_{RSH}	$\overline{CSDS}$ high to $\overline{RS}$ valid hold time	0		ns
t_{RWH}	$\overline{CSDS}$ high to R/$\overline{W}$ high hold time	0		ns
t_{DTH}	$\overline{CSDS}$ high to LD1 – 7 valid hold time	11.5	33.5	ns
t_{TST}[15]	$\overline{CSDS}$ high to LD1 – 7 three state	11.5	34.5	ns
t_{CSDT}[16]	$\overline{CSDS}$ high to $\overline{LDTACK}$ high	9.2	27.2	ns
t_{CSH}	$\overline{CSDS}$ high time	20		ns
t_{ACCR}	$\overline{CSDS}$ low to $\overline{LDTACK}$ low read access time	$t_{CKPD} \div 2 + 12.5$	$3t_{CKPD} \div 2 + 40.3$	ns
Register write (see figure 4)				
t_{RSS}	RS valid to $\overline{CSDS}$ low set-up time	0		ns
t_{RWS2}	R/$\overline{W}$ low to $\overline{CSDS}$ low set-up time	0		ns
t_{DS}	LD1 – LD7 valid to $\overline{CSDS}$ low set-up time	0		ns
t_{RSH}	$\overline{CSDS}$ high to RS valid hold time	0		ns
t_{RWH2}	$\overline{CSDS}$ high to R/$\overline{W}$ low hold time	0		ns
t_{DH}	$\overline{CSDS}$ high to LD1 – LD7 valid	0		ns
t_{CSDT}	$\overline{CSDS}$ high to $\overline{LDTACK}$ high	9.2	27.2	ns
t_{DTCS}	$\overline{LDTACK}$ low to $\overline{CSDS}$ high	0		ns
t_{IRQ}	$\overline{LDTACK}$ low to $\overline{IRQn}$ low	1.2	8.2	ns
t_{ACCW}	$\overline{CSDS}$ low to $\overline{LDTACK}$ low write access time	$t_{CKPD} \div 2 + 13.3$	$3t_{CKPD} \div 2 + 40.3$	ns
t_{CSH}	$\overline{CSDS}$ high time	20		ns
Interrupt acknowledge (see figure 5)				
t_{IKDS}	$\overline{IACK}$ low to $\overline{DS}$ low	0		ns
t_{ADDS}[6]	A1 – A3 valid to $\overline{DS}$ low set-up	0		ns
t_{ADIK}[6]	A1 – A3 valid to $\overline{IACKN}$ low set-up	0		ns
t_{IKBF}[7]	$\overline{IACKN}$ low to $\overline{BUFEN}$ low	$t_{CKPD} + 18.3$	$3t_{CKPD} \div 2 + 29.2$	ns
t_{DSBF}[7]	$\overline{DS}$ low to $\overline{BUFEN}$ low	$t_{CKPD} + 18.3$	$3t_{CKPD} \div 2 + 29.2$	ns
t_{IRA1}[8]	$\overline{IACKN}$ low to $\overline{IRQn}$ high	$t_{CKPD} + 12.5$	$3t_{CKPD} \div 2 + 29.2$	ns
t_{IRA2}[8]	$\overline{DS}$ low to $\overline{IRQn}$ high	$t_{CKPD} + 29.2$	$3t_{CKPD} \div 2 + 29.2$	ns
t_{DYO1}[9]	$\overline{IACKN}$ low to $\overline{IACKOUT}$	$t_{CKPD} + 11.2$	$3t_{CKPD} \div 2 + 29.2$	ns
t_{DYO2}[9]	$\overline{DS}$ low to $\overline{IACKOUT}$ low	$t_{CKPD} + 11.2$	$3t_{CKPD} \div 2 + 29.2$	ns

AC ELECTRICAL CHARACTERISTICS (Continued)

PARAMETER		LIMITS		UNIT
		Min	Max	
t_{BFBD}	$\overline{BUFEN}$ low to BD1 – BD7	3.2	38.2	ns
t_{BFDT}	$\overline{BUFEN}$ low to $\overline{DTACK}$ low	38.2	50	ns
t_{ADRH}	$\overline{DTACK}$ low to A1 – A3 valid hold time	0		ns
t_{DTDS}	$\overline{DTACK}$ low to $\overline{DS}$ high	0		ns
t_{DTIK}	$\overline{DTACK}$ low to $\overline{IACK}$ high	0		ns
t_{IDT}[10]	$\overline{IACK}$ high to BD1 – BD7 valid hold time	3.0	8.4	ns
t_{DSDT}[10]	$\overline{DS}$ high to BD1 – BD7 valid hold time	3.0	8.4	ns
t_{ITST}[11]	$\overline{IACK}$ high to BD1 – BD7 three state	3.0	9.1	ns
t_{DTST}[11]	$\overline{DS}$ high to BD1 – BD7 three state	3.0	9.1	ns
t_{DBF}[12]	$\overline{DS}$ high to $\overline{BUFEN}$ high	12.2	32.2	ns
t_{IBF}[12]	$\overline{IACK}$ high to $\overline{BUFEN}$ high	12.2	32.2	ns
t_{DDTK}[13]	$\overline{DS}$ high to $\overline{DTACK}$ high	12.2	32.2	ns
t_{IDTK}[13]	$\overline{IACK}$ high to $\overline{DTACK}$ high	12.2	32.2	ns
t_{IOUT}[14]	$\overline{IACK}$ high to $\overline{IACKOUT}$ high	6.2	17.2	ns
t_{DOUT}[14]	$\overline{DS}$ high to $\overline{IACKOUT}$ high	6.2	17.2	ns
t_{IAKH}	$\overline{IACK}$ high time	20		ns
t_{IINH}	$\overline{IACK}$ high to $\overline{IACKIN}$ high	0		ns
t_{IKIN}	$\overline{IACK}$ low to $\overline{IACKIN}$ low	0		ns
Reset timing (see figure 6)				
t_{RST}	$\overline{RESET}$ low time	51		ns
Clock timing (see figure 7)				
t_{CKH}[16]	Clock high	45		ns
t_{CKPD}	Clock period	90		ns

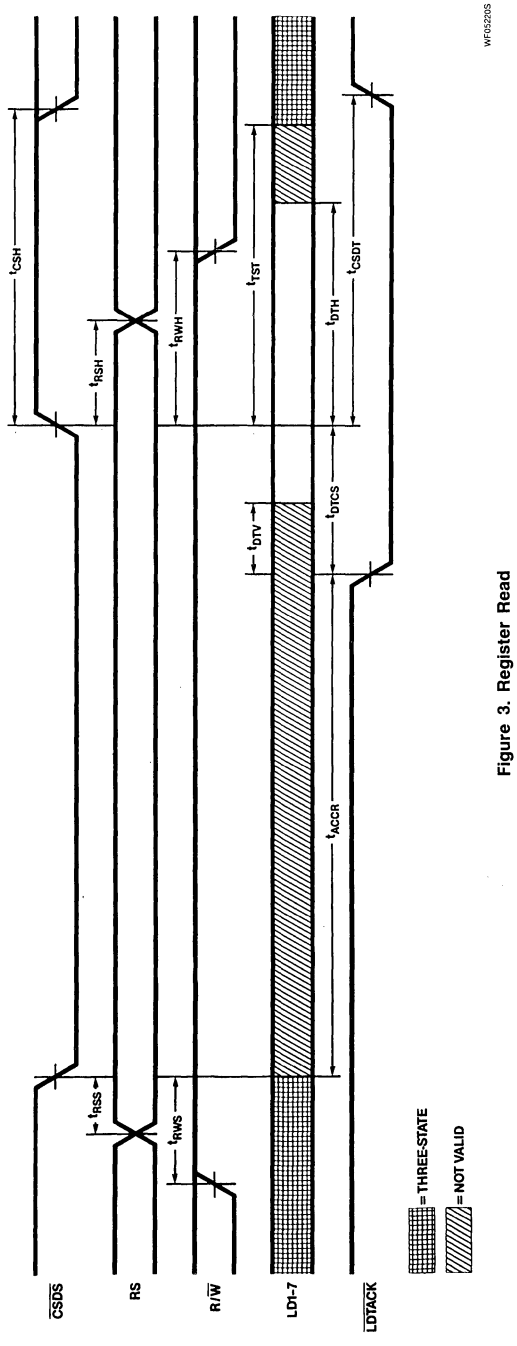


Figure 3. Register Read

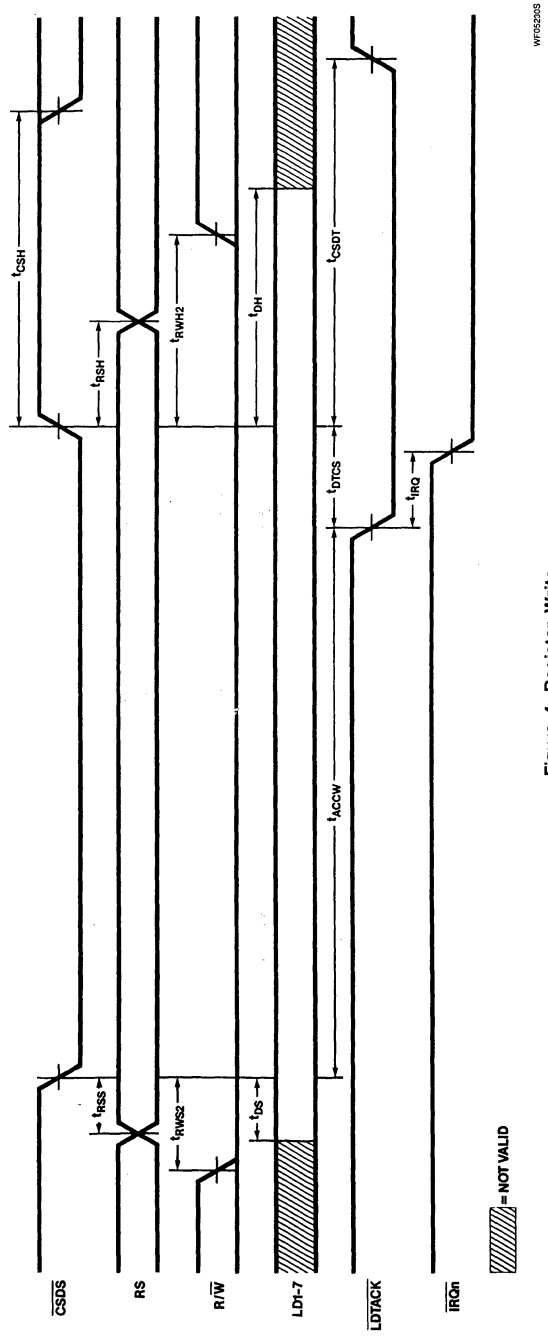


Figure 4. Register Write

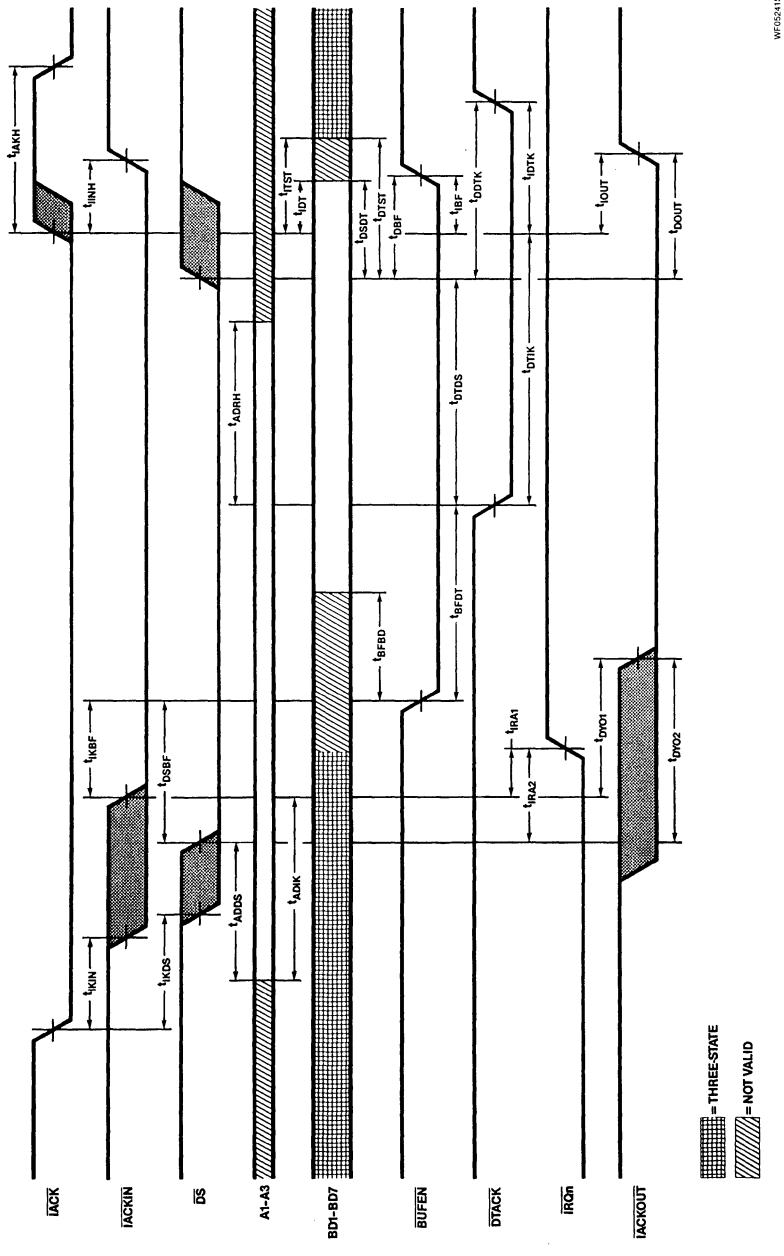


Figure 5. Interrupt Acknowledge

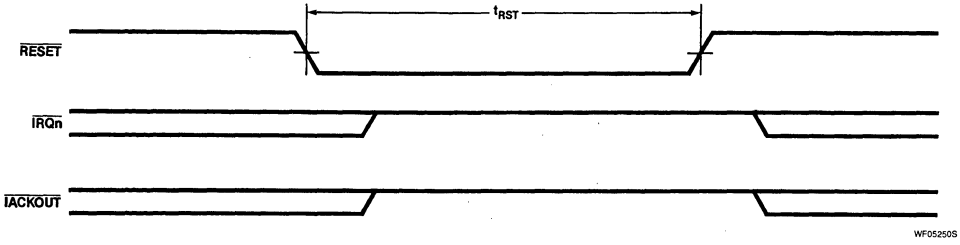


Figure 6. Reset Timing

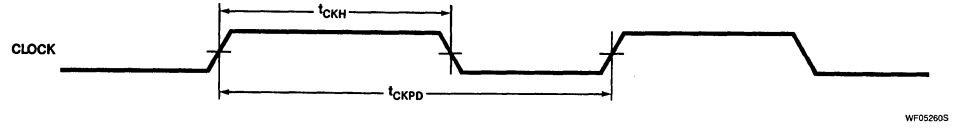


Figure 7. Clock Timing

SCB68155
Interrupt Handler

Preliminary Specification

Microprocessor Products

DESCRIPTION

The Signetics SCB68155/8X824 is an asynchronous interrupt handler for VMEbus and VERSAbus™ systems. Up to 14 interrupts are prioritized by the SCB68155/8X824 to one of seven levels and are output on the interrupt priority level lines ($\overline{IPL0}$ – $\overline{IPL2}$). The SCB68155/8X824 prioritizes the interrupts in the following manner: local bus requests over system bus requests with the non-maskable interrupt ($\overline{NMI}$) considered the highest priority local interrupt ($\overline{NMI}$ over $\overline{IRQ7}$, then $\overline{LRQ6}$ – $\overline{LRQ1}$ over $\overline{IRQ6}$ – $\overline{IRQ1}$).

The local interrupt requests can be programmed to be either active high or low, and either edge or level sensitive. The system bus interrupt requests are always active low and level sensitive. The non-maskable interrupt is always negative edge triggered.

During a local interrupt acknowledge sequence, two modes of response are available: vectored mode or device-supplies-the vector mode.

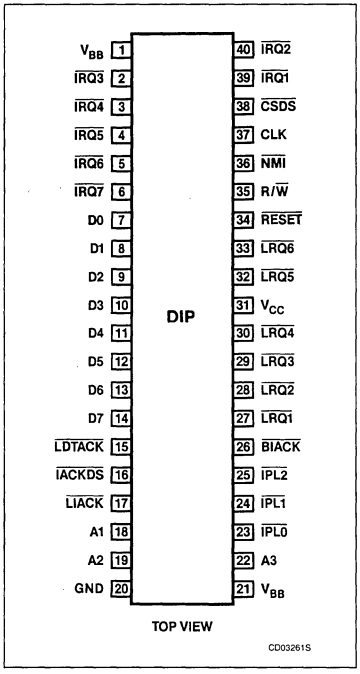
For system bus responses, the SCB68155/8X824 works with a bus requester (for example, the Signetics SCB68175/8X821 Bus Master or SCB68172 VMEbus Controller), to acquire a status/ID byte (interrupt vector) from the system.

The SCB68155/8X824 was designed primarily for interface to the VMEbus. For more information regarding the protocol definitions, proper use, and application of this device, refer to the VMEbus Specification Manual.

FEATURES

- **Asynchronous interrupt handler for VMEbus and VERSAbus systems**
- **Receives and prioritizes non-maskable, six local and seven system bus interrupts**
- **Interrupts may be polled in lieu of real time operation**
- **Programmable local interrupt response**
- **Works with the SCB68175/8X821 to acquire status/ID byte (vector) during bus interrupt acknowledge**
- **Complete device status, including last interrupt acknowledged**
- **High speed bipolar technology**

PIN CONFIGURATION



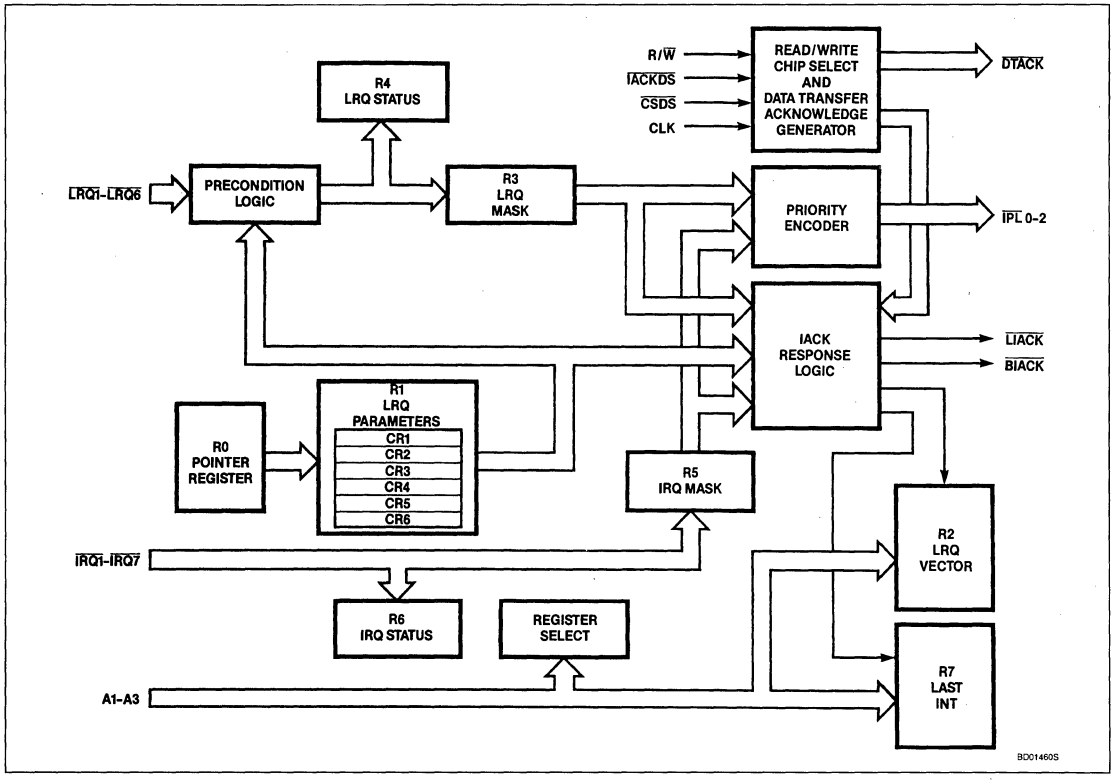
VERSAbus is a registered trademark of Motorola, Inc.

Interrupt Handler | SCB68155

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C
Ceramic DIP	SCB68155CAI40
Plastic DIP	SCB68155CAN40

BLOCK DIAGRAM



2

PIN DESCRIPTION

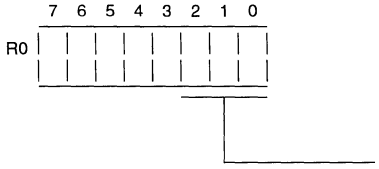
MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{BB}	1, 21	I	**Supply Voltage:** Supply voltage for internal gates.
$\overline{IRQ1}$ – $\overline{IRQ7}$	39, 40, 2 – 6	I	**Bus Interrupt Request:** Active low inputs for bus generated interrupts.
D0 – D7	7 – 14	I/O	**Bus Data:** Three state local data bus.
$\overline{LDTACK}$	15	O	**Local Data Transfer Acknowledge:** Active low, open collector output. Indicates that valid data is available on the local data bus during interrupt acknowledge cycle or data transfer cycle.
$\overline{IACKDS}$	16	I	**Interrupt Acknowledge:** Active low interrupt acknowledge input from the local master. This signal must be qualified by the local master's data strobe prior to input.
$\overline{LIACK}$	17	O	**Local Interrupt Acknowledge:** Active low interrupt acknowledge totem pole output to the local interrupting devices.
A1 – A3	18, 19, 22	I	**Address Lines:** Address inputs from local master.
GND	20	I	**Ground**
$\overline{IPL0}$ – $\overline{IPL2}$	23 – 25	O	**Interrupt Priority Level:** Active low totem pole outputs to the local master. The priority level of the interrupt request is encoded on these outputs.
$\overline{BIACK}$	26	O	**Bus Interrupt Acknowledge:** Active low interrupt acknowledge totem pole output to the system bus.
$\overline{LRQ1}$ – $\overline{LRQ6}$	27 – 30, 32, 33	I	**Local Interrupt Request:** User can define the active state of these inputs.
V_{CC}	31	I	**Supply Voltage:** +5V power supply.
$\overline{RESET}$	34	I	**Reset:** Active low input reset.
R/$\overline{W}$	35	I	**Read/Write:** This signal specifies the data transfer cycle to be either read or write.
$\overline{NMI}$	36	I	**Non – Maskable Interrupt:** Active low highest priority interrupt.
CLK	37	I	**Clock:** Clock input (typically CPU clock).
$\overline{CSDS}$	38	I	**Chip Select:** Active low chip select input for register I/O. This input must be qualified by the local master's data strobe prior to input.

REGISTERS

The SCB68155/8X824 contains eight internal registers (R0 – R7) accessible to the local master. There are also six sub-registers contained in register R1. Register R0 specifies which sub-register is to be accessed in R1. Register R2 stores the interrupt vector for vectored mode responses. Register R3 and R5 are the interrupt mask registers for the local and system bus interrupts respectively. Registers R4 and R6 are the status registers for local and bus interrupts respectively, allowing all interrupts to be polled. Register R7 can be read by the local master to determine the last interrupt acknowledged.

All data transfers between the SCB68155/8X824 and the local master are done using the local data bus (D0 – D7), address bus (A1 – A3), a chip select ($\overline{CSDS}$) and a read/write (R/$\overline{W}$) input.

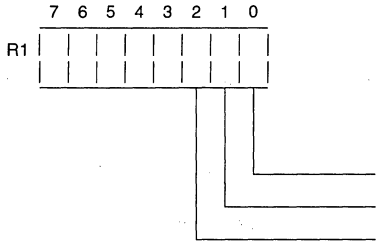
Register R0 – A3A2A1 = 000



Pointer register (write only).
Bit 0 – 2 of R0 specify which control sub-register CR1 – CR6 during an access of R1. During register I/O, bits 7 – 3 will read as 0.

B2 – B1 – B0	
000 – none	100 – CR4
001 – CR1	101 – CR5
010 – CR2	110 – CR6
011 – CR3	111 none

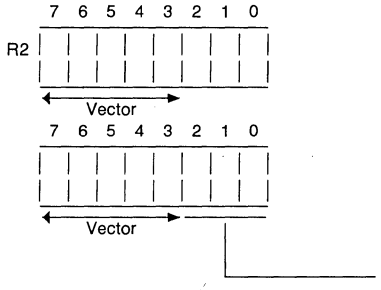
Register R1 – A3A2A1 = 001



Control registers CR1 – CR6 (read or write).
These six registers program the function of the local interrupt requests ($\overline{LRQ1}$ – $\overline{LRQ6}$). (CR1 programs $\overline{LRQ1}$, CR2 programs $\overline{LRQ2}$, etc). During register I/O, bits 7 – 3 will be read as 0.

$\overline{LRQn}$ active state (high/low)	(1 = active high)
$\overline{LRQn}$ edge/level sensitive	(1 = edge sensitive)
$\overline{LRQn}$ vector enable	(1 = enabled)

Register R2 – A3A2A1 = 010

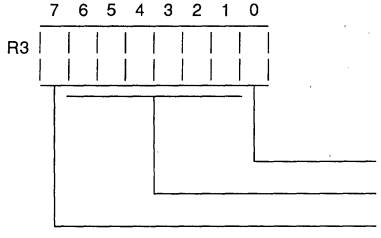


LRQ vector (read or write).
Bits 7 – 3 of this register are the top five bits of the local interrupt vector. During register I/O, bits 2 – 0 will be read as zeros.

LRQ vector output during local interrupt acknowledge (If vector enable = 1).

	100 – $\overline{LRQ4}$
001 – $\overline{LRQ1}$	101 – $\overline{LRQ5}$
010 – $\overline{LRQ2}$	110 – $\overline{LRQ6}$
011 – $\overline{LRQ3}$	111 – $\overline{NMI}$

Register R3 – A3A2A1 = 011

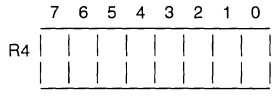


LRQ mask (read or write).
This register allows the user to mask local interrupts. It also enables vectored response for $\overline{NMI}$.

$\overline{NMI}$ Vector enable	(1 = active high)
$\overline{LRQ}$ 1 – 6 mask	(1 = interrupt enabled)
$\overline{NMI}$ mask	(1 = $\overline{NMI}$ enabled)

Bit 1 = $\overline{LRQ1}$	Bit 5 = $\overline{LRQ5}$
Bit 2 = $\overline{LRQ2}$	Bit 6 = $\overline{LRQ6}$
Bit 3 = $\overline{LRQ3}$	Bit 7 = $\overline{NMI}$

Register R4 – A3A2A1 = 100

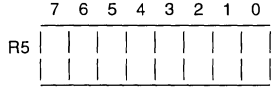


LRQ mask (read only).
Local interrupts can be polled through this register. During register I/O, bit 0 will be read as a 0.

Bit 1 = $\overline{LRQ1}$
Bit 2 = $\overline{LRQ2}$
Bit 3 = $\overline{LRQ3}$
Bit 4 = $\overline{LRQ4}$
Bit 5 = $\overline{LRQ5}$
Bit 6 = $\overline{LRQ6}$
Bit 7 = $\overline{NMI}$

$\overline{LRQ}$ status (1 = interrupt pending)
$\overline{NMI}$ status (1 = interrupt pending)

Register R5 – A3A2A1 = 101

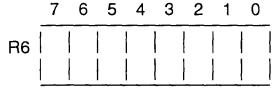


IRQ mask (read or write).
This register allows the user to mask system bus interrupts. During register I/O, bit 0 will be read as a 0.

Bit 1 = $\overline{IRQ1}$
Bit 2 = $\overline{IRQ2}$
Bit 3 = $\overline{IRQ3}$
Bit 4 = $\overline{IRQ4}$
Bit 5 = $\overline{IRQ5}$
Bit 6 = $\overline{IRQ6}$
Bit 7 = $\overline{IRQ7}$

(1 = interrupt enabled)

Register R6 – A3A2A1 = 110

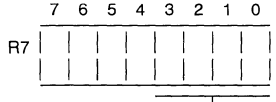


IRQ status (read only).
System bus interrupts can be polled through this register. During register I/O, bit 0 will be read as 0.

Bit 1 = $\overline{IRQ1}$
Bit 2 = $\overline{IRQ2}$
Bit 3 = $\overline{IRQ3}$
Bit 5 = $\overline{IRQ5}$
Bit 6 = $\overline{IRQ6}$
Bit 7 = $\overline{IRQ7}$

$\overline{IRQ}$ status (1 = interrupt pending)

Register R7 – A3A2A1 = 111



Last interrupt acknowledged (read only).
This register can be read by the local CPU to determine the last interrupt acknowledged. During register I/O, bits 7 – 4 will be read as 0.

0000 – none
0001 – $\overline{IRQ1}$
0010 – $\overline{IRQ2}$
0011 – $\overline{IRQ3}$
0100 – $\overline{IRQ4}$
0101 – $\overline{IRQ5}$
0110 – $\overline{IRQ6}$
0111 – $\overline{IRQ7}$
1000 – none
1001 – $\overline{LRQ1}$
1010 – $\overline{LRQ2}$
1011 – $\overline{LRQ3}$
1100 – $\overline{LRQ4}$
1101 – $\overline{LRQ5}$
1110 – $\overline{LRQ6}$
1111 – $\overline{NMI}$

FUNCTIONAL OPERATION

Typical Configuration

The SCB68155/8X824 can handle interrupts from 14 sources: seven bus interrupt requests generated on the $\overline{IRQ1}$ – $\overline{IRQ7}$ inputs, six local interrupt sources generated on the $\overline{LRQ1}$ – $\overline{LRQ2}$ inputs, and one non-maskable interrupt which may originate locally or from the system (such as the system's AC fail signal). All interrupts are encoded to one of seven levels and output on the $\overline{IPL0}$ – $\overline{IPL2}$ lines. Table 1 shows how the SCB68155/8X824 encodes the interrupts.

$\overline{BIACK}$ is the bus interrupt acknowledge signal which is asserted during a bus interrupt acknowledge sequence. $\overline{BIACK}$ can be used to get the associated bus requester (for example, the Signetics SCB68175/8X821 or SCB68172), to acquire an interrupt vector from the system bus. Figure 1 shows a typical SCB68155/8X824-SCB68175/8X821 system configuration.

$\overline{LIACK}$ is the local interrupt acknowledge signal which is asserted during a local interrupt acknowledge sequence. Figure 2 shows a typical configuration for the SCB68155/8X824.

Non-Maskable Interrupt (NMI)

The highest priority interrupt request is the non-maskable interrupt ($\overline{NMI}$). It is an active low, negative edge-triggered interrupt. $\overline{NMI}$ is considered by the SCB68155/8X824 to be the highest priority local interrupt, however, the user is not restricted to having it represent a local device. When the local master responds to an $\overline{NMI}$, bit 7 in the LRQ status register R4 is cleared to 0.

Both vectored and device-supplies-the-vector modes are available with $\overline{NMI}$. However, it is recommended that the SCB68155/8X824's response to an $\overline{NMI}$ be a vectored mode interrupt acknowledge.

Local Interrupts

The SCB68155/8X824 can handle interrupts generated by local devices through its six local interrupt request lines ($\overline{LRQ1}$ – $\overline{LRQ6}$). The local interrupt requests are prioritized with $\overline{LRQ6}$ being the highest priority, and $\overline{LRQ1}$ the lowest priority.

The response of the SCB68155/8X824 to an acknowledge of a local interrupt can be selected by means of the SCB68155/8X824's R1 register. Pointer register R0 points to one of the six control sub-registers when accessing register R1. The six control registers (CR1 – CR6) in register R1 define the functions of the six local interrupt requests ($\overline{LRQ1}$ – $\overline{LRQ6}$).

Control Register 'n' Bit 0
Selects local interrupt requests 'n' ($\overline{LRQn}$), to be either low or high. Bit 0 = 1 defines active state to be high.

Control Register 'n' Bit 1
Selects local interrupt request 'n', to be either edge or level sensitive. Bit 1 = 1 defines LRQ1 to be edge sensitive. Two modes of operation for a local interrupt response are possible; vectored mode and device-supplies-the-vector mode. In vectored mode, the SCB68155/8X824 supplies the interrupt vector to the local CPU and asserts $\overline{LDTACK}$ to complete the transfer. In the device-supplies-the-vector mode, the local interrupting device supplies its own interrupt vector and asserts $\overline{LDTACK}$ to complete the transfer.

Control Register 'n' Bit 2
Selects either vectored mode or device-supplies-the-vector mode response. Bit 2 = 1 enables vectored mode operation for $\overline{LRQn}$. The vector register R2 allows the user to program the five most significant bits (bits 7 – 3) of the interrupt vector supplied in vectored mode. During a vectored local interrupt acknowledge cycle, the upper five bits of the vector register are concatenated with a 3-bit interrupt level (address lines A3 = B2 of the vector, A2 = B1 and A1 = B0). This forms the unique vector for the local interrupt request level being acknowledged.

The local interrupt request mask register R3 allows the user to selectively enable local interrupt requests by setting appropriate bits in the register.

The current state of the local interrupt requests can be determined by the local master by reading the local interrupt status register R4.

Local Interrupt Acknowledge

An interrupt acknowledge, by the local CPU, is signified by the assertion of the interrupt acknowledge input ($\overline{IACKDS}$). The SCB68155/8X824 responds by reading the three address lines (A1 – A3) to determine what level is being acknowledged. If a local interrupt is the highest priority interrupt pending on the level acknowledged, the SCB68155/8X824 will respond as though it is programmed for that level.

If vectored mode is programmed, the SCB68155/8X824 will assert the local interrupt acknowledge ($\overline{LIACK}$) and place the interrupt vector on the local data bus. To complete the transfer of the vector to the local CPU, the SCB68155/8X824 asserts the local data transfer acknowledge signal ($\overline{LDTACK}$).

If device-supplies-the-vector mode is programmed, the SCB68155/8X824 asserts the local interrupt acknowledge signal ($\overline{LIACK}$). The interrupting device is then allowed to place its own vector on the local data bus and assert $\overline{LDTACK}$.

When a local interrupt is acknowledged by the local master, the appropriate bit in the LRQ status register R4 is cleared to 0.

Bus Interrupts

The VMEbus specification defines a maximum of seven interrupt levels. The SCB68155/8X824 can handle seven system bus interrupts through its $\overline{IRQ1}$ – $\overline{IRQ7}$ lines. Bus interrupt request are active low level sensitive, and prioritized with $\overline{IRQ7}$ being the highest priority and $\overline{IRQ1}$ the lowest priority. The bus mask control register R5 allows the user to selectively enable bus interrupt requests by setting appropriate bits in the register. The local CPU can read the bus interrupt status register R6 to determine the current state of the bus interrupt requests.

Bus Interrupt Acknowledge

The local CPU asserts the interrupt acknowledge signal ($\overline{IACKDS}$) to signify an interrupt acknowledge. The SCB68155/8X824 responds by reading the interrupt level on A1 – A3 to determine what level is being acknowledged. If a local interrupt is not pending on the level acknowledged, and that bus level is not masked, the SCB68155/8X824 will assert bus interrupt acknowledge ($\overline{BIACK}$). If that bus level is masked, the SCB68155/8X824 will not respond to the interrupt acknowledge by the local master.

Part of the interrupt acknowledge sequence for a bus interrupt consists of acquiring a vector (status/ID byte) from the system bus. The bus signals required to acquire this vector are available with a bus controller. The

Table 1. SCB68155/8X824 Interrupt Level Encoding

INTERRUPT REQUEST LEVEL	INTERRUPT PRIORITY LEVEL OUTPUTS		
	$\overline{IPL2}$	$\overline{IPL1}$	$\overline{IPL0}$
$\overline{NMI}$, $\overline{IRQ7}$	0	0	0
$\overline{LRQ6}$, $\overline{IRQ6}$	0	0	1
$\overline{LRQ5}$, $\overline{IRQ5}$	0	1	0
$\overline{LRQ4}$, $\overline{IRQ4}$	0	1	1
$\overline{LRQ3}$, $\overline{IRQ3}$	1	0	0
$\overline{LRQ2}$, $\overline{IRQ2}$	1	0	1
$\overline{LRQ1}$, $\overline{IRQ1}$	1	1	0
None	1	1	1

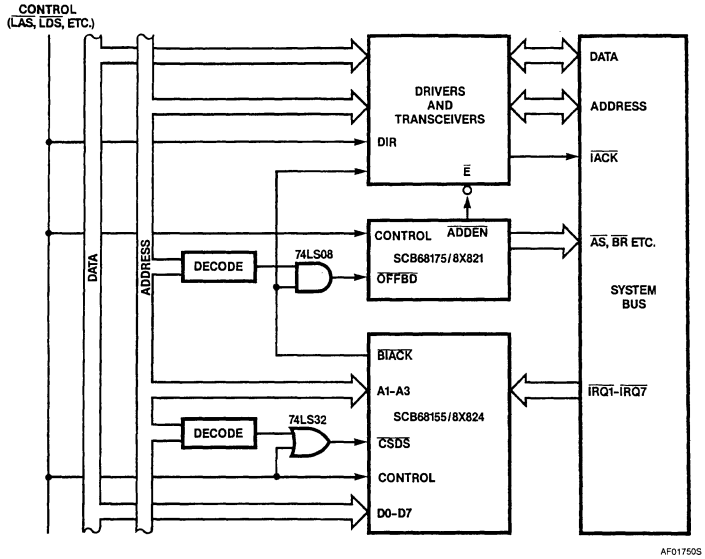


Figure 1. SCB68155/8X824-SCB68175/8X821 Typical System Configuration

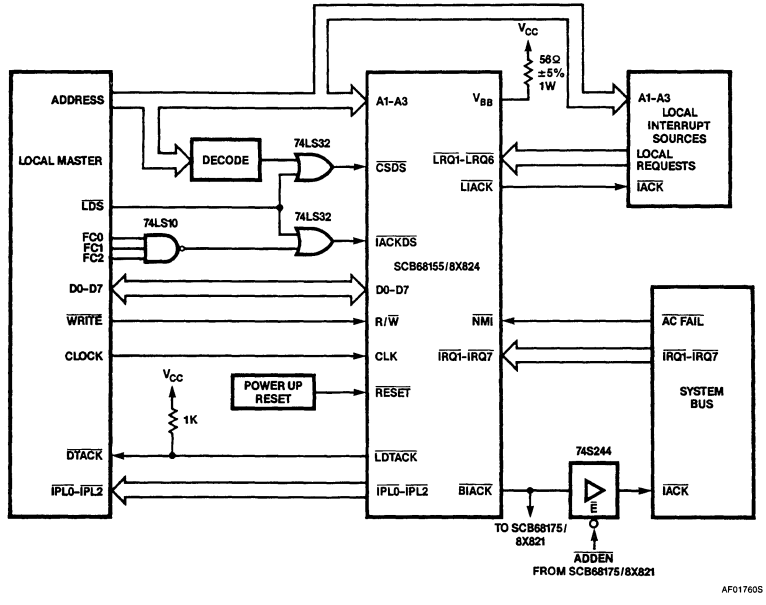


Figure 2. Typical SCB68155/8X824 Local Master Configuration

Signetics SCB68175/8X821 bus controller can be used by the SCB68155/8X824 to acquire the vector (status/ID byte), thereby eliminating the need for the SCB68155/8X824 to duplicate this bus control function. Because most interrupts are serviced by boards that already have the SCB68175/8X821, a one-chip addition of the SCB68155/8X824 gives that board complete interrupt handling capability.

Since the SCB68155/8X824 is an asynchronous device, it is possible for a local interrupt request to be asserted during acknowledgement of a bus interrupt on the same level. The SCB68155/8X824 passes all local interrupt requests through transparent latches which close during each interrupt acknowledge cycle. All possibility of contention is therefore eliminated.

Reset

When $\overline{\text{RESET}}$ is asserted, the SCB68155/8X824 drives $\overline{\text{LDTACK}}$, $\overline{\text{LIACK}}$, $\overline{\text{BIACK}}$ and $\overline{\text{IPL0}}$ – $\overline{\text{IPL2}}$ all high. The D0 – D7 I/O pins go to three-state and all internal registers are cleared.

2

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
Supply voltage[3]	−0.5 to +7.0	V
Input voltage[3]	−0.5 to +5.5	V
Voltage applied to output in off-state[3]	−0.5 to +5.5	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{CC}	Supply voltage		4.75	5.25	V
V_{BB}	Supply voltage		1.35	1.65	V
I_{CC}	V_{CC} supply current	$V_{CC} = 5.25V$		65	mA
I_{BB}	V_{BB} supply current	$V_{BB} = 1.65V$		190	mA
I_{IL}	Input low current	$V_{CC} = 5.25V$, $V_{BB} = 1.65V$, $V_{IL} = 0.4V$		−20	μA
I_{IH}	Input high current	$V_{CC} = 5.25V$, $V_{BB} = 1.65V$, $V_{IH} = 2.7V$		20	μA
I_{OS}	Short circuit output current except $\overline{LDTACK}$	$V_{CC} = 5.25V$, $V_{OUT} = 0V$[6]	−15	−100	mA
V_{OL}	Output low voltage	$V_{CC} = 4.75V$, $V_{BB} = 1.35V$, $I_{OL} = 8mA$		0.6	V
V_{OH}	Output high voltage except $\overline{LDTACK}$	$V_{CC} = 4.75V$, $V_{BB} = 1.35V$, $I_{OH} = -3mA$	2.5		V
I_I	Input leakage current	$V_{CC} = 5.25V$, $V_{IN} = 5.25V$		100	μA
I_{CEX}	Open collector leakage current $\overline{LDTACK}$	$V_{CC} = 4.75V$, $V_{OUT} = 4.25V$		100	μA
V_{IC}	Input clamp voltage	$V_{CC} = 4.75V$, $I_{IN} = -10mA$	−1.5		V
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on + 150°C maximum junction temperature and thermal resistance of 60°C/W junction to ambient for ceramic package (116°C/W for plastic package).
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all inputs swing between 0.4V and 2.4V with a transition time of 3ns maximum and output voltages are checked at 0.8V and 2.0V.
6. At any time, no more than one output should be connected to ground.
7. t_{TST} is always greater than or equal to t_{DTH}.
8. These parameters are guaranteed at the values listed; these values were determined by characterization procedures. All other tabular entries are taken directly from simulation results run at a range of operation frequencies; these values are not tested or guaranteed.

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5]

PARAMETER		LIMITS		UNIT
		Min	Max	
Register read (see figure 3)				
t_{RWS}	R/$\overline{W}$ to $\overline{CSDS}$ low set-up time	10		ns
t_{IAKS}	$\overline{IACKDS}$ high to $\overline{CSDS}$ low set-up time	10		ns
t_{ADRS}	A1 – A3 valid to $\overline{CSDS}$ low set-up time	30		ns
t_{DTV}	$\overline{CSDS}$ low to D0 – D7 set-up time		89	ns
t_{ACCR}	$\overline{CSDS}$ low to $\overline{LDTACK}$ low read access time		$2t_{CKPD} + 116$[8]	ns
t_{RWH}	$\overline{CSDS}$ high to R/$\overline{W}$ high hold time	0		ns
t_{ADRH}	$\overline{CSDS}$ high to A1 – A3 valid hold time	0		ns
t_{DTH}	$\overline{CSDS}$ high to D0 – D7 valid hold time	0	79	ns
t_{TST}[7]	$\overline{CSDS}$ high to D0 – D7 three state	0	80	ns
t_{ACK}	$\overline{CSDS}$ high to $\overline{LDTACK}$ high time	0	66	ns
t_{CSH}	$\overline{CSDS}$ high time	10		ns
t_{DTCS}	$\overline{LDTACK}$ low to $\overline{CSDS}$ high	0		ns
Register write (see figure 4)				
t_{RWS2}	R/$\overline{W}$ low to $\overline{CSDS}$ low set-up time	10		ns
t_{IAKS}	$\overline{IACKDS}$ high to $\overline{CSDS}$ low set-up time	10		ns
t_{ADRS}	A1 – A3 valid to $\overline{CSDS}$ low set-up time	30		ns
t_{DS}	D0 – D7 valid to $\overline{CSDS}$ low set-up time	0		ns
t_{ACCW}	$\overline{CSDS}$ low to $\overline{LDTACK}$ low write access time		$2t_{CKPD} + 116$[8]	ns
t_{RWH2}	$\overline{CSDS}$ high to R/$\overline{W}$ low hold time	0		ns
t_{ADRH}	$\overline{CSDS}$ high to A1 – A3 valid hold time	0		ns
t_{DH}	$\overline{CSDS}$ high to D0 – D7 valid hold time	0		ns
t_{ACK}	$\overline{CSDS}$ high to $\overline{LDTACK}$ high time	0	66	ns
t_{CSH}	$\overline{CSDS}$ high time	100		ns
t_{DTCS}	$\overline{LDTACK}$ low to $\overline{CSDS}$ high time	0		ns
Vector mode (see figure 5)				
t_{CSS}	$\overline{CSDS}$ high to $\overline{IACKDS}$ low set-up time	10		ns
t_{PDL}	$\overline{IACKDS}$ low to $\overline{LIACK}$ low propagation time	t_{CKPD}	$2t_{CKPD} + 68$	ns
t_{DAV}	$\overline{IACKDS}$ low to D0 – D7 vector valid		103	ns
t_{ACCV}	$\overline{IACKDS}$ low to $\overline{LDTACK}$ low (vector access time)	t_{AKPD}	$t_{ACKPD} + 116$[8]	ns
t_{IKH}	$\overline{IACKDS}$ high time	100		ns
t_{DAH}	$\overline{IACKDS}$ high to D0 – D7 valid hold time	0	111	ns
t_{TRST}[7]	$\overline{IACKDS}$ high to D0 – D7 three state	0	115	ns
t_{IKDT}	$\overline{IACKDS}$ high to $\overline{LDTACK}$ high	0	81	ns
t_{PHD}	$\overline{IACKDS}$ high to $\overline{LIACK}$ high propagation delay	0	42	ns
t_{DTIK}	$\overline{LDTACK}$ low to $\overline{IACKDS}$ high time	0		ns
t_{ADRS}	A1 – A3 valid to $\overline{IACKDS}$ low set-up time	0		ns
t_{ADRH}	$\overline{IACKDS}$ high to A1 – A3 valid hold time	0		ns

AC ELECTRICAL CHARACTERISTICS (Continued)

PARAMETER		LIMITS		UNIT
		Min	Max	
Device supplies the vector mode (see figure 6)				
t_{CSS}	$\overline{CSDS}$ high to $\overline{IACKDS}$ low set-up time	10		ns
t_{PDL}	$\overline{IACKDS}$ low to $\overline{LIACK}$ low propagation time delay	t_{CKPD}	$2t_{CKPD} + 68$	ns
t_{IKH}	$\overline{IACKDS}$ high time	100		ns
t_{PDH}	$\overline{IACKDS}$ high to $\overline{LIACK}$ high propagation delay	0	42	ns
t_{ADS}	A1 – A3 valid to $\overline{IACKDS}$ low set-up time	0		ns
t_{ADH}	$\overline{IACKDS}$ high to A1 – A3 valid hold time	0		ns
Bus interrupt acknowledge (see figure 7)				
t_{CSS}	$\overline{CSDS}$ high $\overline{IACKDS}$ set-up time	10		ns
t_{PDL2}	$\overline{IACKDS}$ low to $\overline{BIACK}$ low propagation delay	t_{CKPD}	$2t_{CKPD} + 68$	ns
t_{IKH}	$\overline{IACKDS}$ high time	100		ns
t_{PDH2}	$\overline{IACKDS}$ high to $\overline{BIACK}$ high propagation delay	0	50	ns
t_{ADS}	A1 – A3 valid to $\overline{IACKDS}$ low set-up time	0		ns
t_{ADH}	$\overline{IACKDS}$ high to A1 – A3 hold time	0		ns
Reset timing (see figure 8)				
t_{RST}	$\overline{RESET}$ low time	120		ns
Clock timing (see figure 9)				
t_{CKPD}	Clock period	100		ns
t_{CKH}	Clock high	50		ns

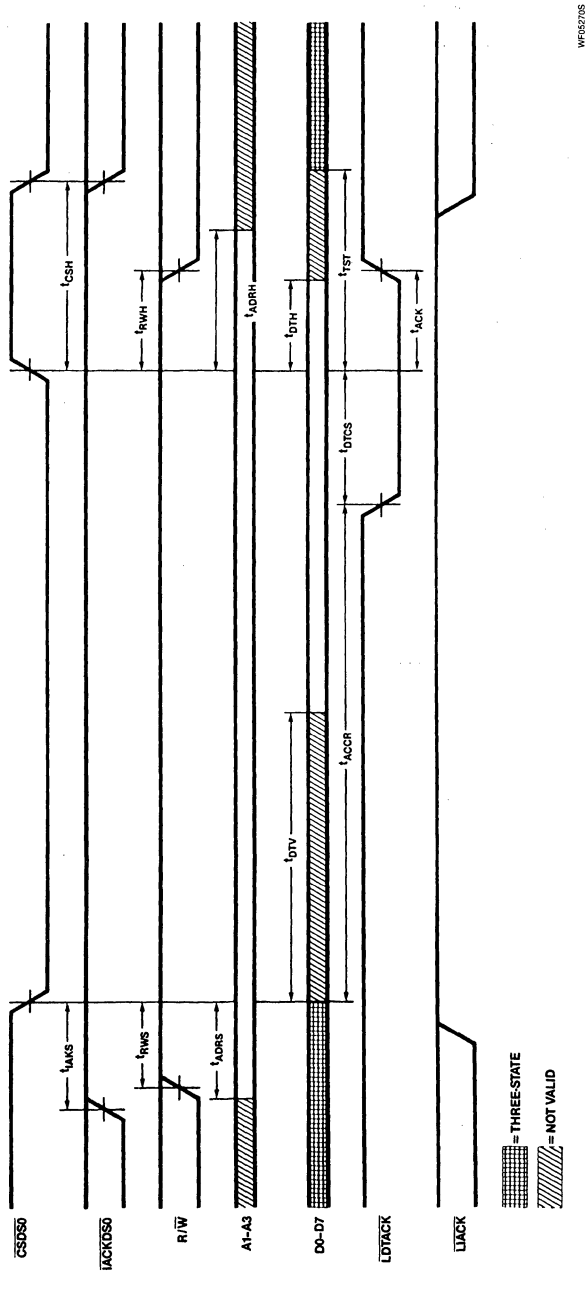


Figure 3. Register Read

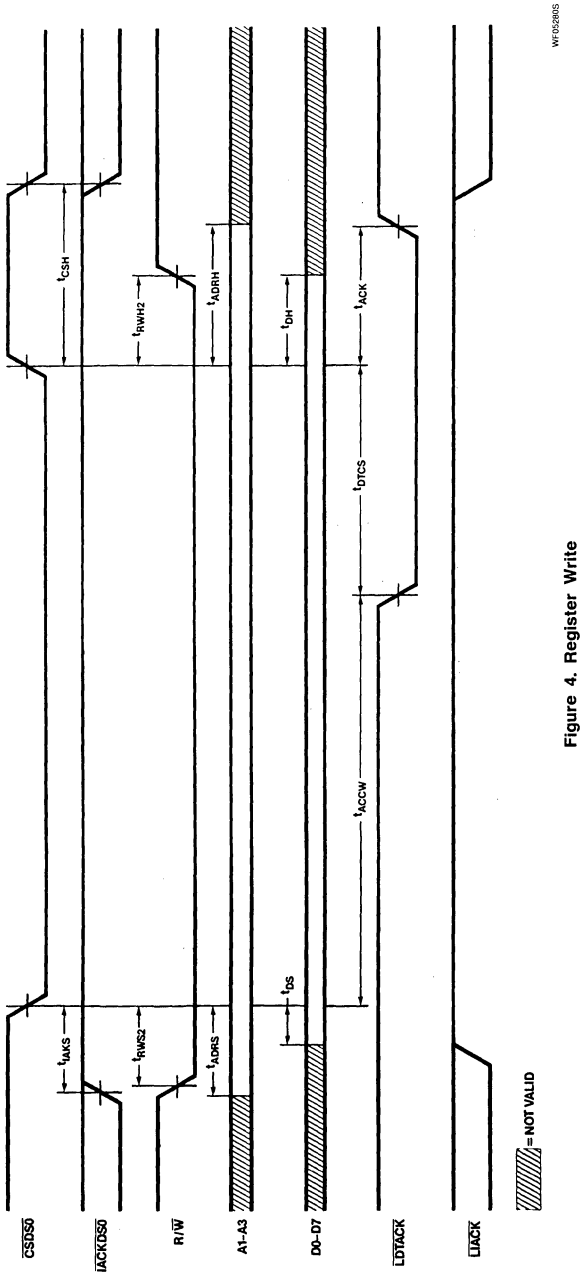


Figure 4. Register Write

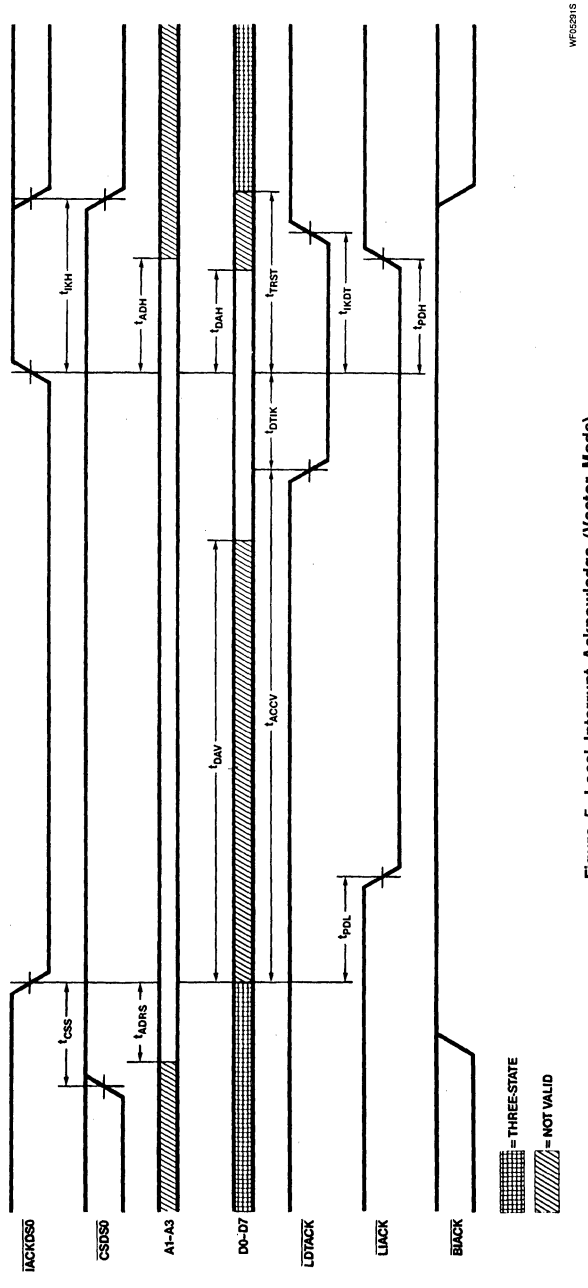


Figure 5. Local Interrupt Acknowledge (Vector Mode)

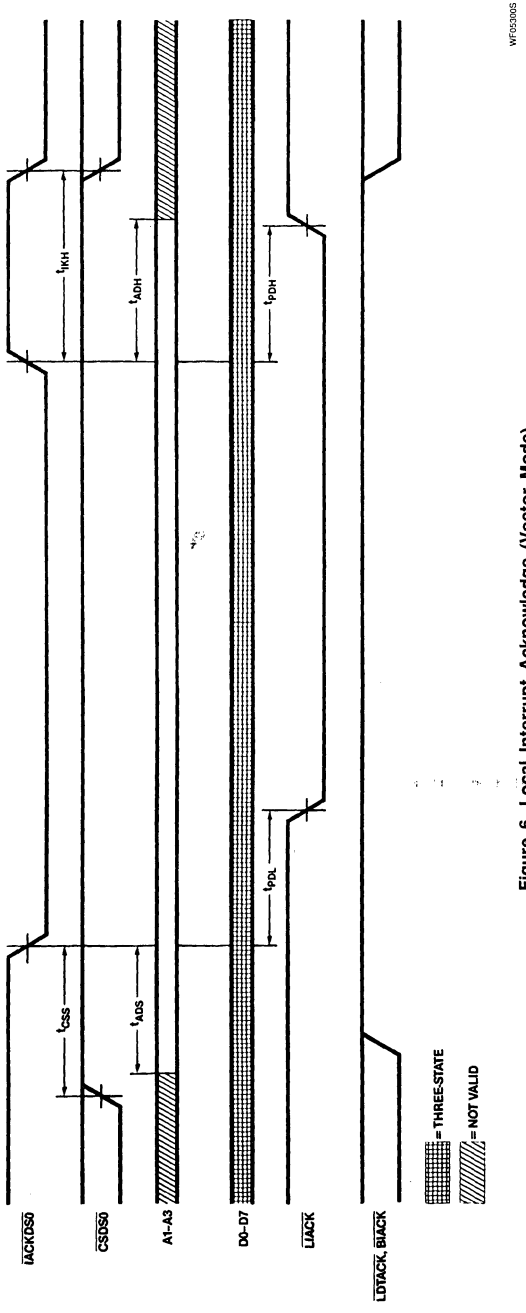


Figure 6. Local Interrupt Acknowledge (Vector Mode)

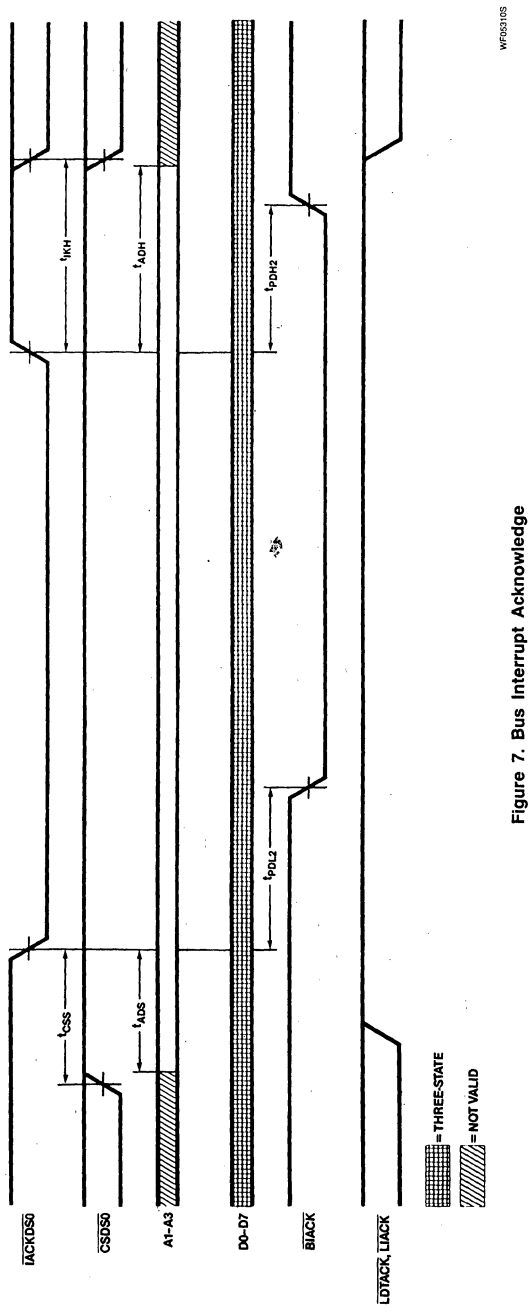


Figure 7. Bus Interrupt Acknowledge

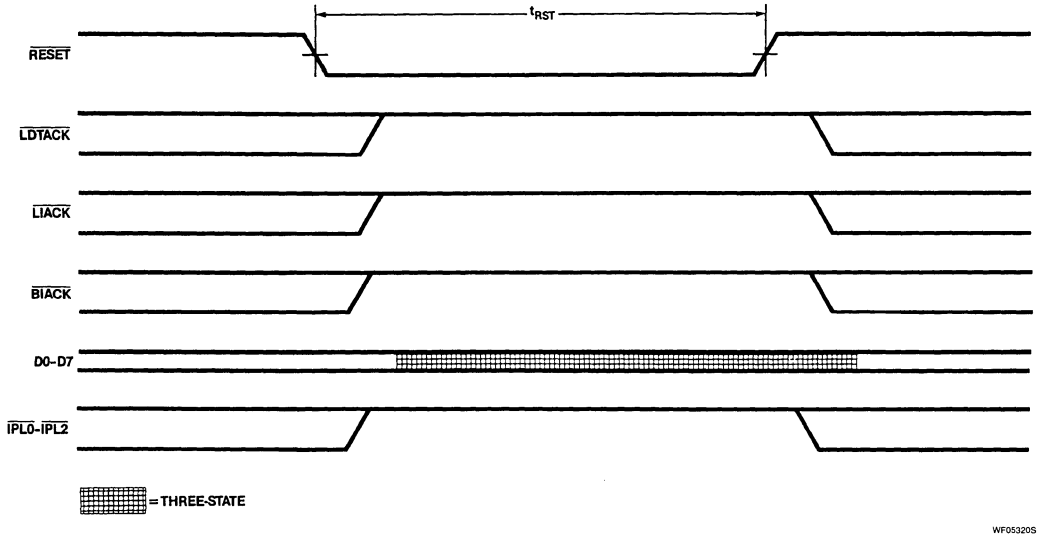


Figure 8. Reset

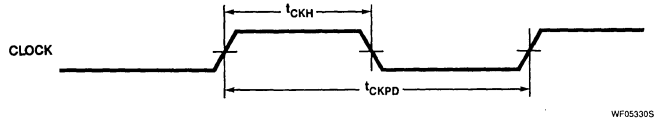


Figure 9. Clock

Signetics

SCB68171

Very Little Serial Interface Chip (VLSIC)

Objective Specification

Microprocessor Products

DESCRIPTION

The Signetics Very Little Serial Interface Chip (VLSIC) is a bipolar interface device which connects one or more VMSbus controllers to the VMSbus itself. It provides bus driving and receiving in addition to latching data in both the transmit and receive directions.

SERCLK on the VMSbus has a waveform as shown in figure 1, with four edges per cycle which are designated C1, S1, C2, S2. SYSCLK is used to discriminate (differentiate) the phases of SERCLK. The SYSCLK input should have a nominal 50% duty cycle and a cycle time which is 18% (2/11) that of SERCLK, but SYSCLK and SERCLK need not be synchronous or have any fixed phase relationship. The 16MHz SYSCLK on the VMEbus meets these requirements for a 2.9MHz SERCLK.

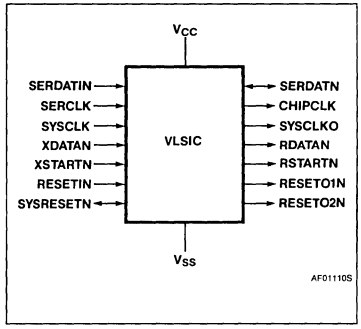
The VLSIC samples both of the XDATAN and XSTARTN inputs on the SERCLK edge designated C1. XDATAN is clocked directly to SERDATN. XSTARTN is clocked into an internal latch. The VLSIC then clocks the low-active OR (positive logic AND) of SERDATN and SERDATIN to RDATAN on the S1 edge. If XDATAN was high and XSTARTN was low at C1, and RSTARTN is high from the previous bit cell, the VLSIC then makes SERDATN low on the C2 edge, thus making a VMSbus start bit. If XDATAN was high at S1, then on the S2 edge it clocks the low active OR of SERDATN and SERDATIN to RSTARTN. If SERDATN was low at S1, it keeps (or makes) RSTARTN high at S2.

CHIPCLK is driven high from the C1 edge of SERCLK, and driven low from the C2 edge. Thus RDATAN and RSTARTN set-up to CHIPCLK edges by approximately the low time of SERCLK. The VMSbus controller(s) must meet the specified set-up and hold times to C1 for data on XDATAN and XSTARTN.

FEATURES

- **70mA open collector drive for SERDATN**
- **Low capacitive loading**
- **Discriminates SERCLK into single-phase chip clock output**
- **Separates data and start bits for both receive and transmit**
- **Provides single bus load for multiple VMSbus controllers**
- **Simplifies controller design and allows use of slower technology**
- **VMEbus receiver for SYSCLK**
- **VMEbus driver/receiver for SYSRESETN**
- **16-pin DIP**

FUNCTIONAL DIAGRAM



JAM FEATURE

A VMSbus controller is required to "jam" the bus by sending a string of "ones" (low) on SERDATN when a start bit is sensed while the controller is sending or tracking a frame. The first one bit should directly follow the misplaced start bit. Since the minimum S2-to-C1 time of SERCLK (25ns) is less than the sum of the maximum S2-to-RSTART-low time plus the minimum set-up of XDATAN to C1, a controller cannot do this using XDATAN in the normal fashion.

The following feature is provided to solve this problem. The condition XDATAN low, XSTARTN low, and CHIPCLK high directly sets the flipflop controlling SERDATN, and makes SERDATN low. (Note that a VMSbus controller would never assert both XDATAN and XSTARTN low in normal operation.) The assertion of both XDATAN and XSTARTN must occur soon enough to satisfy the SERDATN to S1 set-up requirements of all the modules on the VMSbus.

SYSRESET DRIVING/RECEIVING

The VLSIC is primarily intended for use in the P1 region of a VMEbus card. Space and functionality is at a premium in this area. Accordingly, the VLSIC includes an autonomous function of driving and receiving SYSRESETN on the VMEbus. A low on the RESETIN input makes the VLSIC drive the SYSRESETN pin low. SYSRESETN is also received and driven onto two open-collector outputs RESETO1N and RESETO2N. RESETO1N has a high drive capability and is suitable for connection to the RESETN pin of a 680x0 processor, while RESETO2N has lower drive and capacitance and can be connected to the processor's HALTN pin. This function has no connection to the rest of the VLSIC, and could thus be used for some other purpose.

VMSBUS CONTROLLER DESIGN

Controllers using VLSIC should signal as follows on XDATAN and XSTARTN.

1. Controllers should present the next bit on XDATAN and XSTARTN in response to the lowgoing edge of CHIPCLK. For a 2.9MHz SERCLK, they have at least 90nsec to do so, and approximately 120nsec from RDATAN valid.
2. Controllers may release XDATAN, XSTARTN to high in response to the highgoing edge of CHIPCLK. Since these are typically open-collector outputs of the controllers, there may be a timing advantage to do so.
3. A controller should present XDATAN and XSTARTN low in a "jam" situation, in a combinatorial fashion from RSTARTN. Thereafter the controller can release XSTARTN from the lowgoing edge of CHIPCLK, and may either signal 511 or 512 one bits in the usual fashion (1 and 2 above), or may just keep XDATAN low.
4. If two VMSbus controllers connected to the same VLSIC become "locally desynchronized", it is possible that one will present XDATAN low and the other XSTARTN low for the same bit cell. If this occurs, XDATAN predominates and SERDATN is driven low for a "one" bit. Thus the subsequent (transient) combination of XDATAN and XSTARTN low and CHIPCLK high actually has no effect. The controller presenting XSTARTN thereafter receives RDATAN low, and continues to try to send the start bit.

2

PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
SERCLK		I	**Direct connect to VMSbus clock:** Clocks SERDATN to RDATAN, RSTARTN. Clocks XDATAN, XSTARTN to SERDATN. Used to generate single-phase CHIPCLK.
SYSCLK		I	**Direct connect to VMEbus clock or other signal meeting the specified frequency relationship to SERCLK:** Used to discriminate the phases of SERCLK.
SYSCLKO		O	**Repeater of SYSCLK for onboard use.**
SERDATN		I/O	**Direct connect to VMSbus data in bipolar open-collector links:** Provides data output function for other types of links (e.g. optical). Requires pull-up resistor in either case.
SERDATIN		I	**Tied to high logic level in bipolar open-collector links:** Provides data input function for other types of links (e.g. optical).
CHIPCLK		O	**Single phase clock for VMSbus controllers.**
RDATAN		O	**Conveys one/zero bits to VMSbus controller(s).**
RSTARTN		O	**Conveys start bits to VMSbus controller(s).**
XDATAN		I	**Input from VMSbus controller(s):** In normal operation, a low on this line indicates a "one" bit should be sent. Simultaneous assertion of XDATAN and XSTARTN low, while CHIPCLK is high, drives SERDATN low directly in a "jam" condition.
XSTARTN		I	**Input from VMSbus controller(s):** In normal operation, a low on this line indicates a "start" bit should be sent. Simultaneous assertion of XDATAN and XSTARTN low, while CHIPCLK is high, drives SERDATN low directly in a "jam" condition.
RESETIN		I	**Input from onboard logic:** Low state forces SYSRESETN low.
SYSRESETN		I/O	**Direct connect to VMEbus system reset:** Open-collector output from RESETIN, received to drive RESETO1N and RESETO2N. Does not affect other VLSIC logic.
RESETO1N		O	**High-drive open-collector output from SYSRESETN.**
RESETO2N		O	**Low-drive open-collector output from SYSRESETN.**
V_{CC}			**+5 Volts**
V_{SS}			**Ground**

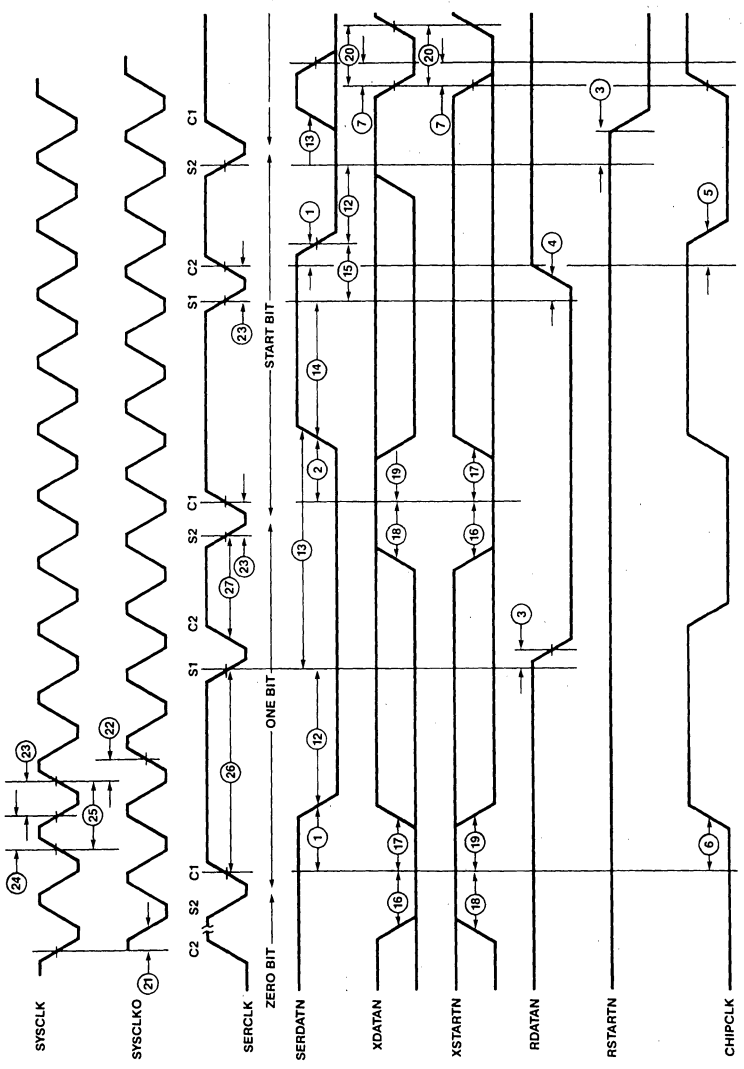


Figure 1. Waveform

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltage	−0.5 to +7.0	V
Input voltage	−0.5 to +5.5	V
Operating temperature range[2]	0 to +70	°C
Storage temperature	−65 to +150	°C

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V ±5%, T_A = 0°C to +70°C, R_L = 90, C_L = 15pF[3,4]

PARAMETER		TEST CONDITIONS	LIMITS Min	LIMITS Max	UNIT
I_{IL}	Low level input current	V_{IN} = 0.4V		−0.4	mA
I_{IH}	High level input current	V_{IN} = 2.7V		20	µA
V_{IL}	Low level input voltage			0.8	V
V_{IH}	High level input voltage				
	SERDATN, SERDATIN, SYSRESETN, RESETIN		1.65		V
	SYSCLK, SERCLK, XDATAN, XSTARTN		2.0		V
$V_{TH+} - V_{TH-}$	Hysteresis				V
	SERDATN, SERDATIN, SYSRESETN, RESETIN		0.15		
V_{OL}	Low level output voltage				
	CHIPCLK, RDATAN, RSTARTN, RESETO2N	I_{OL} = 8mA		0.5	V
	SYSCLKO	I_{OL} = 24mA		0.5	V
	SERDATN, RESETO1N, SYSRESETN	I_{OL} = 70mA		0.5	V
V_{OH}	High level output voltage				
	CHIPCLK, RDATAN, RSTARTN	I_{OH} = −0.4mA	2.7		V
	SYSCLKO	I_{OH} = −1mA	2.7		V
	SYSCLKO	I_{OH} = −2.6mA	2.4		V
I_{OH}	Output leakage current				
	SERDATN, RESETO1N, SYSRESETN	V_{OH} = 2.7V		60	µA
	SERDATN, RESETO1N, SYSRESETN	V_{OH} = 5.5V		250	µA
	RESETO2N	V_{OH} = 5.5V		100	µA
C_I	Input capacitance				
	SERCLK, SYSCLK			8	pF
C_{IO}	I/O capacitance				
	SERDATN			15	pF

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other conditions other than those indicated in the Electrical Characteristics section of this data sheet is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. Parameters are valid over specified temperature range.
4. All voltage measurements are referenced to ground (V_{SS}). All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
5. Output loading 15pF for miminum timing, 300pF for maximum timing.

AC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V ±5%, T_A = 0°C to +70°C, R_L = 90, C_L = 15pF[3,4] (except as noted)

NO.	CHARACTERISTIC	TENTATIVE LIMITS		UNIT
		Min	Max	
1	Prop, C1 or C2 to SERDATN low[5]	9	25	ns
2	Prop, C1 to SERDATN high[5]	10	46	ns
3	Prop, S1 (S2) to RDATAN (RSTARTN) low	7	18	ns
4	Prop, S1 (S2) to RDATAN (RSTARTN) high	6	15	ns
5	Prop, C2 to CHIPCLK low	4	12	ns
6	Prop, C1 to CHIPCLK high	3	9	ns
7	Prop, RDATAN low and RSTARTN low (with CHIPCLK high) to SERDATN low[5]	12	31	ns
8	RDATAN low to CHIPCLK low	t_{CL-6}	t_{CL}	ns
9	RDATAN high to CHIPCLK low	t_{CL-4}	t_{CL}	ns
10	RSTARTN low to CHIPCLK high	t_{CL-9}	t_{CL}	ns
11	RSTARTN high to CHIPCLK high	t_{CL-6}	t_{CL}	ns
12	Set-up, SERDATN and/or SERDATIN low to S1 or S2	8		ns
13	Hold, SERDATN and/or SERDATIN low after S1 or S2	6		ns
14	Set-up, SERDATN and SERDATIN high to S1 or S2	2		ns
15	Hold, SERDATN and SERDATIN high after S1 or S2	8		ns
16	Set-up, XDATAN, XSTARTN low to C1	10		ns
17	Hold, XDATAN, XSTARTN low after C1	0		ns
18	Set-up, XDATAN, XSTARTN high to C1	12		ns
19	Hold, XDATAN, XSTARTN high after C1	0		ns
20	Pulse Width, RDATAN and RSTARTN low with CHIPCLK high	12		ns
21	Prop, SYSCLK low to SYSCLKO low	5	12	ns
22	Prop, SYSCLK high to SYSCLKO high	3	7	ns
23	Pulse width, SYSCLK, SERCLK low (t_{CL})	25		ns
24	Pulse width, SYSCLK high	25		ns
25	Cycle time, SYSCLK (t_{SYCY})	62		ns
26	Pulse width, SERCLK high, C1 to S1			
	(t_{SYCY} = 62.5)	163.5		ns
	(general case)	$3t_{SYCY}-24$		ns
27	Pulse width, SERCLK high, C2 to S2			
	(t_{SYCY} = 62.5)	70	101	ns
	(general case)	70	$2t_{SYCY}-24$	ns

SCB68172
VMEbus Controller (BUSCON)

Objective Specification

Microprocessor Products

DESCRIPTION

The Signetics SCB68172 VMEbus Controller (BUSCON) is an interface device for the VMEbus. It can be used in three different configurations: master-only, slave-only, and master/slave. The SCB68172 can be used with a processor-type interface or with a DMA controller-type interface. In all configurations, it handles the VMEbus signaling protocol in compliance with revisions B and C of the VMEbus Specification.

CONFIGURATION/VERSION

Applications of the BUSCON are identified as follows (see figures 1 through 4):

VERSION	APPLICATION
PMS	Processor-type master/slave
DMAC	DMA controller-type master/slave
MS	Either PMS or (DMAC)
M	Master-only
S	Slave-only

All of these applications are handled by the SCB68172, with unused pins tied to stated logic levels in some of the applications.

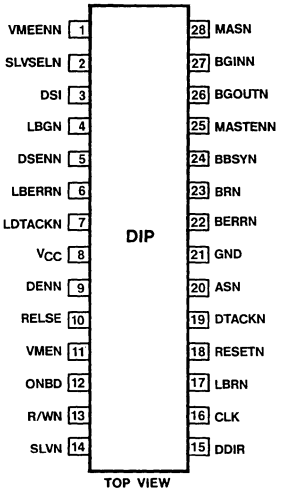
Figure 5 shows a functional model of the SCB68172 logic. The ASN, MASN, LBRN, BGINN, and RELSE inputs are internally synchronized to CLK before being presented to the state machine which determines the major functions of the device. The SLVN, ONBD, and VMEN signals are used directly in the state machine, although they are highly qualified to prevent metastable conditions on the state machine outputs. The BRN, BBSYN and LBGN signals are direct state machine outputs, while ASN, MASTENN, VMEENN, SLVSELN, and BGOUTN are derived from the state machine outputs plus some combinatorial qualification.

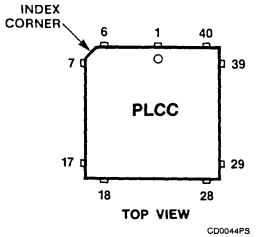
The DSI, R/WN, DTACKN, BERRN, LDTACKN, and LBERRN inputs function largely as direct combinatorial inputs. The DDIR, DTACKN, BERRN, LDTACKN, LBERRN, and (when applicable) MASN outputs are largely derived directly from these direct inputs, with some qualification from the state machine outputs. The DENN and DSENN outputs have complex multi-case logic which uses both the direct inputs and the state machine outputs.

FEATURES

- **Master, slave, or master/slave (dual ported) applications**
- **Helps assure VMEbus compatibility**
- **Allows for address decoding time**
- **Processor or DMA controller interface for master/requester**
- **Master/requester logic allows release on request (ROR) or release when done (RWD) operation, early or intercycle release**
- **Supports and exploits address lookahead**

PIN CONFIGURATION





Pin	Function	Pin	Function
1	NC	23	NC
2	VMEENN	24	DDIR
3	SLVSELN	25	CLK
4	DSI	26	LBRN
5	LBGN	27	RESETN
6	NC	28	NC
7	NC	29	NC
8	DSENN	30	DTACKN
9	LBERRN	31	ASN
10	NC	32	NC
11	LDTACKN	33	GND
12	NC	34	NC
13	V_{CC}	35	BERRN
14	NC	36	NC
15	DENN	37	BRN
16	RELSE	38	BBSYN
17	NC	39	NC
18	NC	40	NC
19	VMEN	41	MASTENN
20	ONBD	42	BGOUTN
21	R/WN	43	BGINN
22	SLVN	44	MASN

VMEbus Controller (BUSCON) SCB68172

ORDERING CODE

PACKAGES	V_{CC} = 5V± 5%, T_A = 0°C to 70°C
Ceramic DIP	SCB68172AC25I28
Plastic DIP	SCB68172AC25N28
Plastic LCC	SCB68172AC25A44

PIN DESCRIPTION

MNEMONIC	PIN NO.		TYPE	CONFIG	NAME AND FUNCTION
	DIP	PLCC			
CLK	16	25	I	All	**Clock:** User-supplied clock signal.
SLVN	14	22	I	S,MS	**Slave:** Active-low decode of the VMEbus address and address modifier lines indicating that the current cycle is for this board. SLVN should not be qualified with ASN nor VMEENN. It is first sampled on the rising clock edge after the rising edge on which ASN is first detected. It must remain valid until after the next low-going edge on DTACKN or BERRN. In a master-only application, SLVN should be pulled up to V_{CC}.
ASN	20	31	I/O I	M,MS S	**Address Strobe:** Direct connect to VMEbus ASN.
VMEN	11	19	I	M,MS	**VME Decode:** Active-low decode of the master's address lines, indicating that the master's current cycle is for a slave on the VMEbus. VMEN should not be qualified with MASN nor MASTENN. It is first sampled on the rising clock edge after the one on which MASN is first detected. Thereafter, it must remain valid until MASN goes false (high). In a slave-only configuration, VMEN should be pulled up to V_{CC}.
LBRN	17	26	I	M,MS	**Local Bus Request:** Connected to the low-active bus request output of a DMA controller. Typically tied to a high logic level in processor-type interfaces.
ONBD	12	20	I	MS	**Onboard:** Active-high decode of the master's address lines, indicating that the master's current cycle is for an onboard slave that is dual-ported with the VMEbus. ONBD should not be qualified with MASN or MASTENN. It is first sampled on the rising clock edge after the one on which MASN is first detected. Thereafter, it must remain valid until after MASN goes false (high). In a master-only or slave-only application, ONBD should be grounded. If a master/slave configuration does not contain "local slaves" as shown in figure 3, VMEN and ONBD should both be connected to an active-low "VME decode". A cycle between the onboard master and a local slave (VMEN high, ONBD low) is ignored by BUSCON, and can proceed concurrently with a cycle between another VMEbus master and an onboard dual-ported slave.
MASN	28	44	I I/O	M,PMS DMAC	**Master's Address Strobe:** RMW and Sequential VMEbus master cycles are accomplished by holding MASN low across several data strobes. If LBGN is high at the end of the RESETN low time, the state of ASN is driven onto MASN whenever BUSCON does not have control of the VMEbus. In a slave-only application, MASN should be pulled up to V_{CC}.
MASTENN	25	41	O	MS	**Master Enable:** In a master/slave application, the low state of this signal enables the master onto the shared bus and enables shared-bus responses back to the master. MASTENN also provides the direction control for the VMEbus address transceivers.
VMEENN	1	2	O	M,MS	**VME Enable:** Active-low enable for the VMEbus address drivers (master-only) or transceivers (master/slave).
SLVSELN	2	3	O	S,MS	**Slave Select:** Active-low select for the onboard slave resources (the shared/dual ported slaves in a master/slave application). Derived from MASN and ONBD, or from ASN and SLVN. If necessary, MASTENN and VMEENN are cycled to provide address set-up time before SLVSELN is asserted.
BRN	23	37	O	M,MS	**Bus Request:** Active-low, open collector VMEbus request. Direct connect to the selected level among VMEbus BR0* – BR3*.
BGINN	27	43	I	M,MS	**Bus Grant In:** Direct connect to the selected level among VMEbus BG0IN* – BG3IN*.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	CONFIG	NAME AND FUNCTION
	DIP	PLCC			
BGOUTN	26	42	O	M,MS	**Bus Grant Out:** Direct connect to the selected level among VMEbus BG0OUT* – BG3OUT*.
BBSYN	24	38	O	M,MS	**Bus Busy:** Active-low, open collector direct connect to VMEbus BBSY*.
LBGN	4	5	I/O	M,MS	**Local Bus Grant:** Active-low, open collector. Can be connected to the bus grant input of a DMA controller. Asserted when LBRN is low and the BUSCON has control of the VMEbus. Grounded, or driven low during RESET, to prevent the ASN state being driven onto MASN when the BUSCON is not in control of the VMEbus.
RELSE	10	16	I	M,MS	**Release:** Active-high signal indicating that the onboard logic wants to release control of the VMEbus. In DMA controller applications, the BGACKN output of the DMAC should be connected to (or positive-logic ANDed into) this signal.
DTACKN	19	30	I/O O	M,MS S	**Data Transfer Acknowledge:** Active-low, open collector. Direct connect to VMEbus DTACK*.
BERRN	22	35	I/O O	M,MS S	**Bus Error:** Active-low , open collector. Direct connect to VMEbus BERR*.
LDTACKN	7	11	O, I/O I	M, MS S	**Local DTACK:** Act ive-low, open collector. Output to onboard master and/or input from onboard slave.
LBERRN	6	9	O, I/O I	M, MS S	**Local Bus Error:** Onboard active-low, open collector. Output to onboard master and/ or input from onboard slave.
DSI	3	4	I	All	**Data Strobe:** The high-active or of the onboard data strobes, which may be from the onboard master or VMEbus master.
DSENN	5	8	O	M,MS	**Data Strobe Enable:** Low-active, used to enable the onboard data strobes onto the VMEbus.
R/WN	13	21	I	All	**Read/Write:** Onboard R/W signal from the onboard master or VMEbus master.
DDIR	15	24	O	All	**Data Direction Control:** Direction control for VMEbus data transceivers. A high level indicates the "onboard-to-VMEbus" direction.
DENN	9	15	O	All	**Data Enable:** Low-active enable for VMEbus data transceivers.
RESETN	18	27	I	All	**RESET:** Low-active reset. Clears BUSCON logic.
V_{CC}	8	13	I	All	**Power Supply:** +5 volts.
GND	21	33	I	All	**Ground:** 0V reference.

ADDRESS DECODING

Both the VMEbus and current high-speed processors provide short address-to-strobe set-up times, such that with all but the most simple decode schemes, designers must provide for delaying the strobe until decoder outputs have become valid. However, BUSCON operates as a finite-state machine and must synchronize address strobes and other inputs before it can act on them. The BUSCON design allows this synchronization time to be overlapped with address decoding.

In general, most BUSCON inputs do not have critical timing parameters. Exceptions are the three address decode signals. Figure 6 shows a somewhat simplified model of the VMEbus slave selection logic in the SCB68172. The ASN signal is qualified and sampled by flip-flops A and B on each rising edge of CLK. Flip-flop C is set when ASN is high between cycles, and cleared by a falling edge on DTACKN or BERRN.

On the rising edge of CLK after flip-flop B samples ASN low, if C is still set and SLVN is low, flip-flop D is set, indicating slave selection. (In reality, there are more terms in the logic to set D.) Once D is set, it remains set until flip-flop A samples ASN high and a similar circuit (not shown) samples DSI low.

Since SLVN is a direct input to flip-flop D, it must meet a set-up time to the clock after ASN is sampled low. Viewed asynchronously, SLVN should be valid slightly less than one clock period after ASN goes low, through shortly after DTACKN goes low.

The onboard logic driven by MASN, VMEN, and ONBD is similar but not as complex. Neither flip-flop C nor a data-strobe-related signal are used, and there are separate flip-flops corresponding to D for each of the VMEN and ONBD signals. VMEN and ONBD should be valid slightly less than one clock period after MASN goes low, through shortly after MASN goes high.

Because ASN and MASN are used directly to clear the corresponding "flip-flop B", their minimum high times are relatively short. However, for consecutive cycles, the inactive time of "flip-flop D" (and signals derived from it) will be at least two clock periods because of the feedback path from "D" to "B".

VMEbus ARBITRATION

BUSCON begins VMEbus arbitration by driving BRN low if MASN, VMEN and ONBD indicate a VMEbus cycle (or if LBRN goes low) and the BUSCON does not have control of the bus.

After driving BRN, BUSCON waits for the BGINN input which is connected to the selected one among the four VMEbus arbitration levels. (During this time it can of course respond to cycles from other VMEbus masters.) When it receives BGINN low while holding BRN low, it drives BBSYN low and thereafter releases BRN. (If it receives

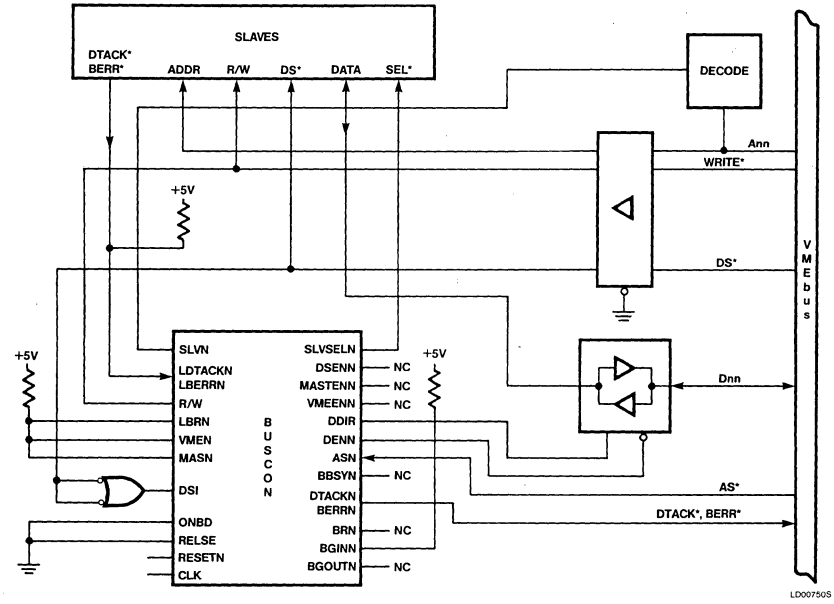


Figure 1. Slave-Only Application

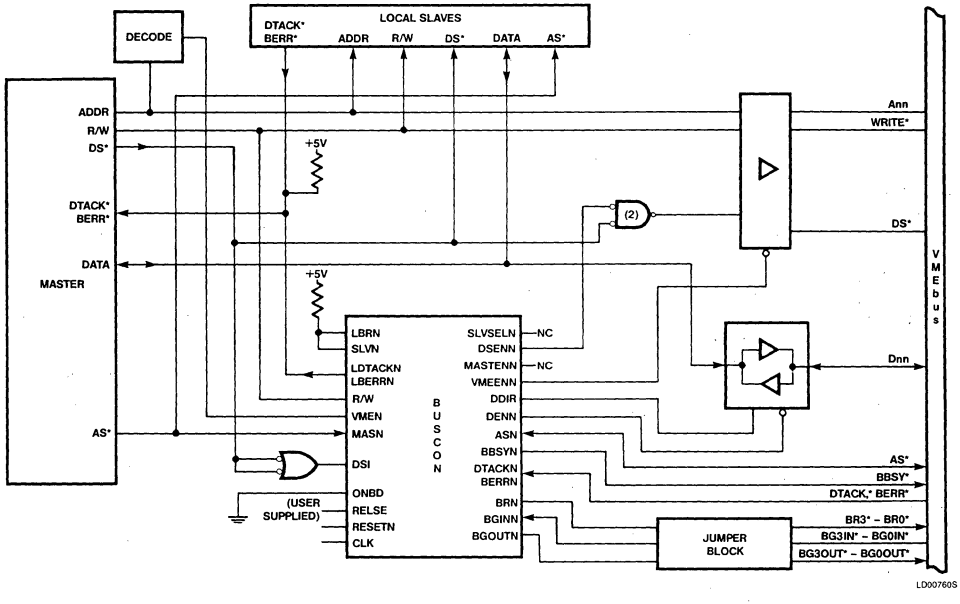


Figure 2. Master-Only Application

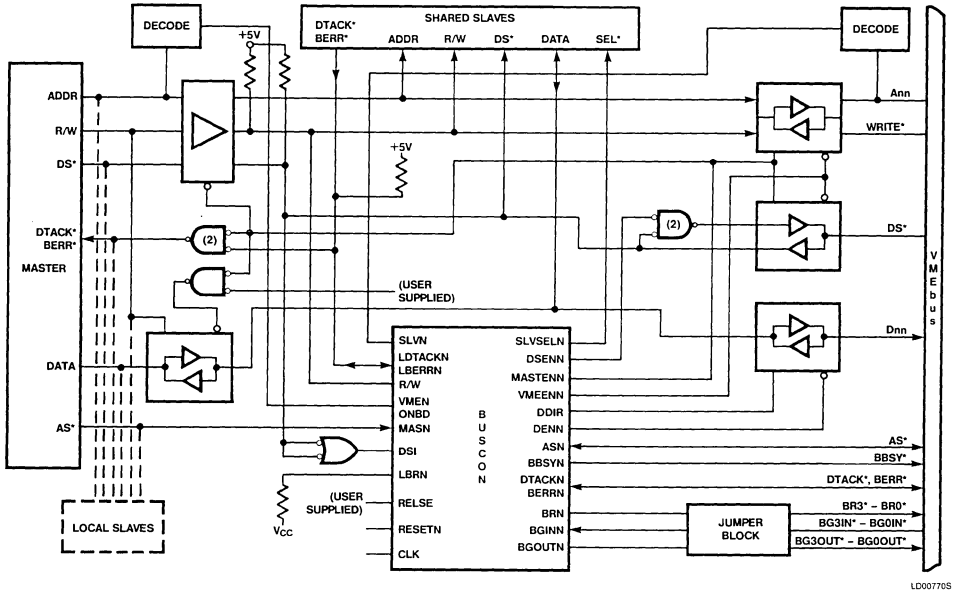


Figure 3. Processor-Type Master/Slave Application

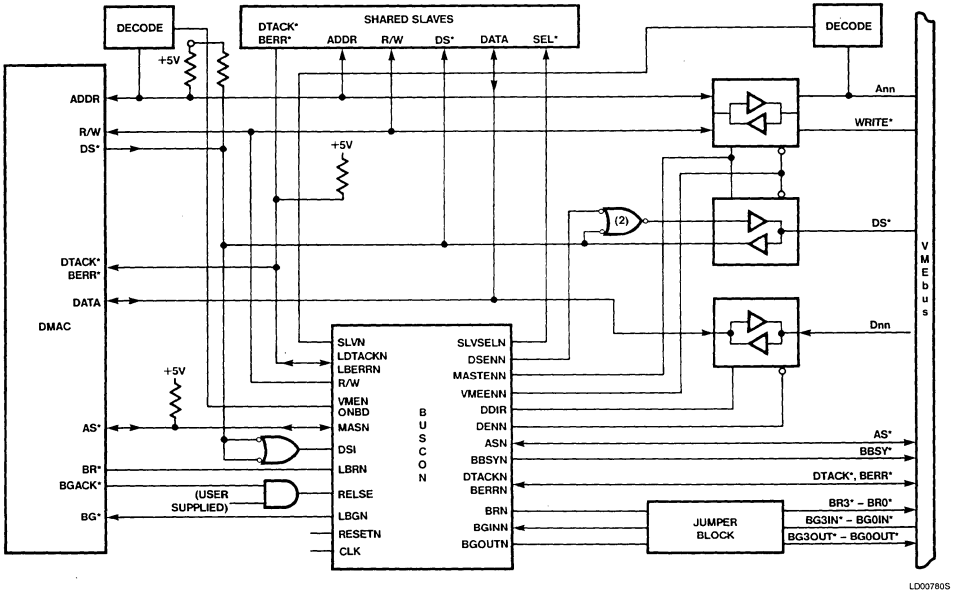


Figure 4. DMAC-Type Master/Slave Application

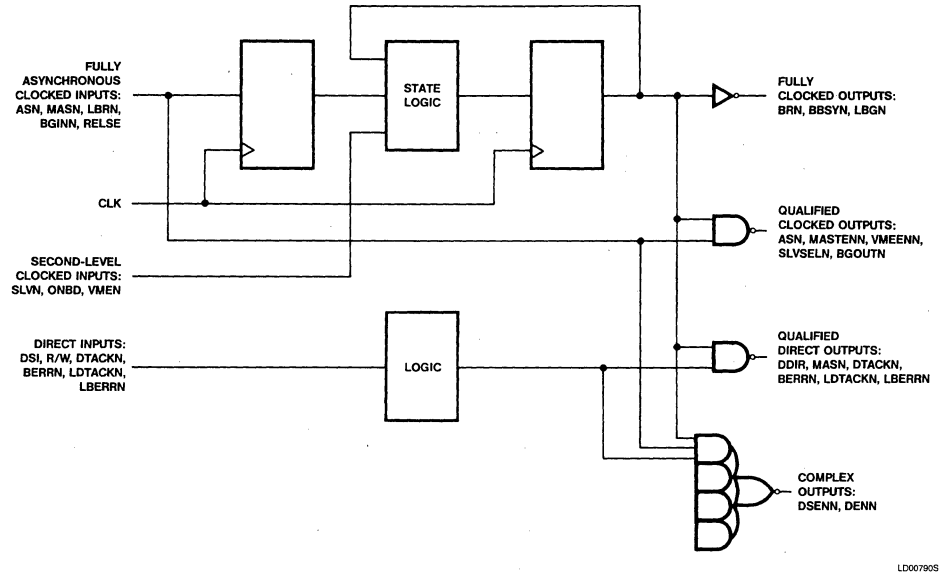


Figure 5. Functional Model

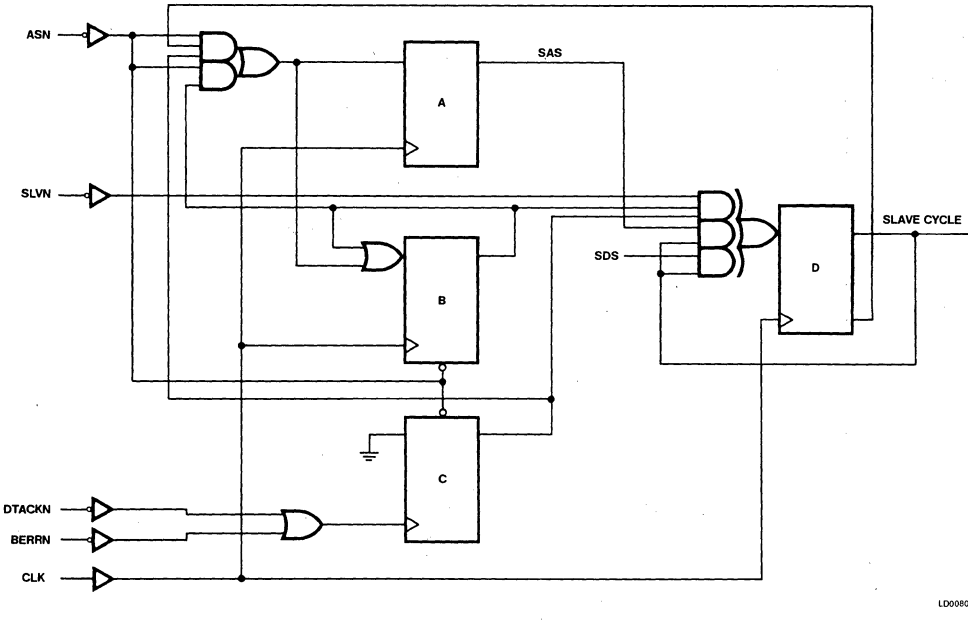


Figure 6. Simplified Model of the SCB68172 Slave Select Logic

BGINN low at any other time, it drives BGOUTN low and continues to do so until BGINN goes high.)

Once BUSCON has driven BBSYN low, it waits for any current VMEbus cycle to complete as evidenced by ASN high. Then it begins to drive ASN (initially high) and drives VMEENN low to enable the address out onto the VMEbus.

If the BRN was initiated by MASN, BUSCON then waits two clock periods for address set-up time before driving ASN low. If the bus acquisition was initiated by LBRN, it waits for MASN.

BUSCON will release the BBSYN signal on a rising clock edge at which all of the following conditions are met:

1. It is at least three clock periods after the edge on which BBSYN was asserted, and
2. Any prior master's cycle has completed and VMEENN has been driven low, and
3. The BGINN input was high on the last rising clock edge, and
4. The RELSE input was high on the last rising clock edge, and
5. It is not the clock edge at which BUSCON asserts ASN, and
6. It is not the clock edge at which BUSCON withdraws ASN, and
7. LBRN was high on the last rising clock edge.

If BBSYN is released while BUSCON is not driving ASN low, then VMEENN goes high when BBSYN is released, to release the VMEbus. Otherwise, VMEENN goes high shortly before ASN goes high.

RELSE is provided to allow user determination of the method of VMEbus release. The BGACKN output of a DMA controller can be connected to (or included in) this signal to allow the device to control how long it keeps the bus. The OR of the VMEbus requests can be connected to (or included in) this signal for release on request (ROR) operation. If RELSE is connected to a constant logic high, BUSCON will release the bus as soon as possible, i.e. during the first bus cycle.

VMEbus MASTER OPERATION

When the BUSCON has VMEbus control, VMEbus cycles indicated on MASN and VMEN produce ASN low on the VMEbus. DDIR and DENN control the VMEbus data transceivers. DDIR reflects the R/WN line.

In a write operation, DENN is driven low to drive data onto the VMEbus whenever the BUSCON has control of the VMEbus, R/WN is low, and the previous VMEbus slave has released DTACKN and BERRN to high. (The DTACKN/BERRN requirement does not apply to subsequent cycles among consecutive writes, if R/WN is maintained continuously low.) DSENN is then driven low when DENN has been low for more than a clock period, and DTACKN and BERRN are high, but not before ASN is driven low. DSENN goes high after DSI goes low or MASN goes high, whichever occurs first. DENN goes high after R/WN goes high, or with VMEENN going high, whichever occurs first.

In a read operation (R/WN is high), DENN goes low to drive data in from the VMEbus after ONBD and VMEN have been sampled, DSI is high, and MASTENN is low. DSENN goes low after DSI, DTACKN, and BERRN are all high, but not before ASN goes low. DSENN and DENN go high after DSI goes low or MASN goes high, whichever occurs first.

DTACKN and BERRN are inputs and drive LDTACKN and LBERRN as outputs. LDTACKN and LBERRN are released when the onboard master makes DSI low. If the VMEbus slave continues to hold DTACKN or BERRN low thereafter, DSENN, LDTACKN and LBERRN are inhibited for the next cycle until the response is released.

MASTENN and VMEENN are kept low while the BUSCON has VMEbus control. The MAS-to-AS delay thus provides automatic address-set-up time for subsequent VMEbus cycles.

The need to transceive the data strobes in a master/slave application, plus qualify the onboard master's strobes with DSENN for output, can be accomplished in several ways as shown in figure 7.

MASTER/SLAVE SWITCHING

BUSCON includes arbitration and switching logic between VMEbus slave cycles and onboard master cycles (to a shared onboard slave or the VMEbus). The logic remains in its previous direction until forced to switch by another cycle. This provides minimum overhead for slave-only or master-only operation, and for consecutive cycles from the same master.

If a master cycle to a shared slave occurs, or BGINN arrives when requesting the VMEbus, after a slave cycle with another VMEbus master, VMEENN goes high to disable the address from the VMEbus. On the next clock edge, MASTENN goes low to enable the master's address back out onto the onboard bus.

For a VMEbus master cycle, if the current VMEbus cycle is also over, VMEENN then goes low to enable the address out onto the VMEbus.

For a master cycle to a shared slave, SLVSELN goes low one clock period after MASTENN goes low, or if the master direction is continuing, after ONBD is sampled high. SLVSELN goes high shortly after MASN goes high. DTACKN and BERRN are isolated from LDTACKN and LBERRN. DSENN is kept high. DENN is kept high except in a write cycle when BUSCON has VMEbus control.

If an onboard master cycle and VMEbus slave cycle both arrive for the shared slaves within the same clock period, the previous direction of the master/slave switch is retained.

VMEbus SLAVE OPERATION

If a VMEbus slave cycle occurs after a master cycle, or while BUSCON is requesting the VMEbus, MASTENN goes high, and on the subsequent clock VMEENN goes low to enable the VMEbus address and control signals onto the board.

SLVSELN goes low one clock period after VMEENN goes low, to signal the shared slave(s) that a cycle is occurring. If the slave mode is continuing, SLVSELN goes low after SLVN is sampled low. SLVSELN goes high shortly after ASN goes high.

DDIR reflects R/WN (in the opposite sense from master operation). LDTACKN and LBERRN are inputs and drive DTACKN and BERRN as outputs.

In write operations, DENN is driven low (to enable data in) whenever R/WN is low and LDTACKN and LBERRN are high. When switching between master and slave operation with R/WN low, DENN sequences like VMEENN.

In read operations, DENN is driven low (to enable data out) after SLVN has been sampled low, and while R/WN and DSI are both high.

SLAVE-ONLY USE

This is the simplest application of the BUSCON. However, handling of board-selection logic from a simple VMEbus address decode, plus driving and sequencing of DTACKN and BERRN, can save VMEbus designers cost and board space even in this application.

SLAVE DESIGN

In the MS and S configurations, slaves operate off the data strobes and SLVSELN rather than address and data strobes. It should be noted that SLVSELN will typically go low after the data strobes go low. Data should not be written nor placed on the data lines until SLVSELN goes low.

68000 DUAL-PORTED OPERATION

BUSCON is ideal for use on a VMEbus board containing a 68000 processor. The obvious approach to dual-porting memory and other slaves, on a board with a 68000, is to use the BRN input of the 68000 to suspend processor operation while another master accesses the onboard slave. This works fine except when the processor has already started a cycle for the VMEbus. This latter coincidence threatens a "deadlock" situation and requires that the processor be "rolled back" off the board's shared bus so that the other master's cycle can occur first. The 68000 has a feature which can be used for this; assertion of both its BERRN and HALTN inputs cause it to suspend operation and retry the cycle when the inputs are released.

The BUSCON does not use these features because there is a flaw in the retry logic. The retry logic does not function during an indivisible RMW sequence (TAS instruction), even in the read cycle. Instead, the assertion of BERRN and HALTN causes an actual bus error exception. It is believed that there is no reliable and general programming solution to the problem of finding the start of the TAS instruction for restart. With 6801x processors, the situation is better because the TAS can be restarted. In any case BUSCON elects to isolate the processor with a few more packages rather than adding complexity to the error-handling software because of dual-ported design.

DMA USE

The BUSCON can be used for VMEbus boards which contain a DMA controller but no processor. Such DMA applications are always master/slave due to the need to program the DMA controller from the VMEbus. There are two operational features of the SCB68172 that are intended for use with DMA controllers. First, the LBRN input can be used to request control of the VMEbus directly, rather than waiting for MASN low and VMEN low as in a processor application. Second, the SCB68172 samples the state of the LBGN pin when RESETN is low. If LBGN is low at the end of RESETN, MASN is used as an input only. If LBGN is high (at the end of the RESETN pulse), the BUSCON thereafter drives the state of VMEbus ASN onto MASN whenever it does not have control of the VMEbus.

For 68000 family DMA controllers, MASN is connected directly to the controller's address strobe pin. The VMEbus ASN-to-MASN feature satisfies the requirement of some DMA controllers that ASN be low on cycles which program them, and also serves to delay the activity of a controller which gets an LBGN response during the last VMEbus cycle by another master.

When LBRN is sampled low, if the BUSCON has retained VMEbus control from previous DMA activity, it continues to retain control for the duration of LBRN being low, and drives LBGN low on the next clock.

Otherwise, it drives VMEbus BRN low on the next clock. When BGINN is sampled low, BBSYN is driven low on the next clock. LBGN is driven low on the same clock as BBSYN if VMEbus ASN is high. If ASN is low, LBGN is

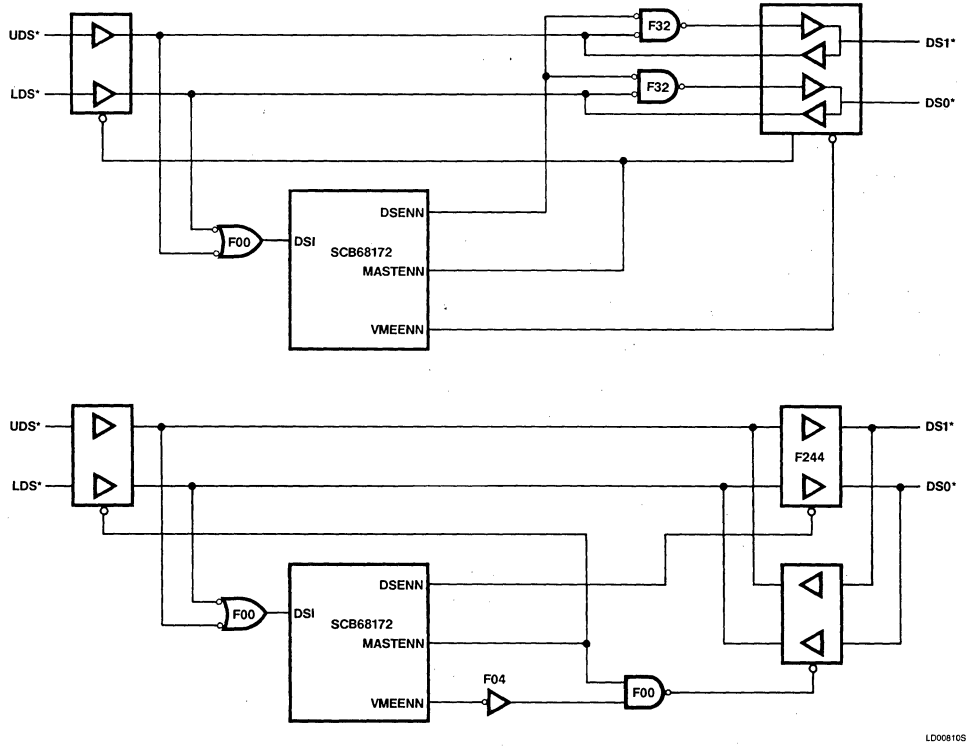


Figure 7. Data Strobe Handling Alternatives in Master /Slave Applications

driven low one clock after BBSYN, except when ASN low and BGINN are both sampled low for the first time in the same clock period and the VMEbus cycle addresses this board – in this last case, LBGN is driven low two clocks after BBSYN. In either of the last two cases, VMEbus ASN low makes MASN low before LBGN goes low, which keeps the DMA controller from starting until the current VMEbus cycle is over.

Note that the local bus request/grant logic and ASN-to-MASN drive feature are separate capabilities, either or both of which can be used in applications not involving a DMA controller. However, note also that when the state of the ASN is driven onto MASN, this is done without conditioning by the state of the master-slave switch. This means MASN can go low before MASTENN and VMEENN have been cycled to bring the VMEbus address onto the board. (The low state of SLVSELN indicates that the VMEbus address is valid on the board.)

DMA applications are always considered master/slave due to the need to program the DMA controller. The BUSCON assumes that it must always have VMEbus control before answering an LBRN with LBGN. If this is not desired, i.e., if the DMA controller will sometimes be programmed to do onboard transfers solely and the designer wishes to optimize for this case, then a processor-type interface should be used, and isolation devices and additional onboard logic are required.

INTERRUPT HANDLING

When a processor handles interrupts from onboard sources and from the VMEbus, the design must include logic to decide whether an interrupt acknowledge cycle is an onboard or offboard cycle. This logic is quite different from the address decoding logic used to make this decision on other cycles.

Performance can be maximized if the interrupt logic can provide ONBD and VMEN within the specified time after MASN goes low, or if the signals can be made to meet their specified set-up and hold times for CLK. In this case ONBD and VMEN need be selected between the IACK and non-IACK sources. Otherwise (i.e., if the interrupt logic presents these signals slowly and asynchronously), MASN must also be selected between the IACK and non-IACK sources.

MASTER BLOCK TRANSFER

The block transfer feature of the VMEbus allows considerable performance improvement for transferring a block of consecutive memory locations. The BUSCON can be used for block transfer operations in the master role.

Master block transfers are applicable to cache subsystems or block transfer on processor boards, and to DMAC-type designs. The only requirements for master block transfers operation with the BUSCON are that external logic must place a block transfer address modifier (AM) code on the VMEbus, and then hold MASN low across a number of data strobes. (Note that a long block transfer can compromise the operation of other VMEbus masters. One strategy to avoid such problems could be to do a minimum of 4 or 8 transfers without interruption, and then switch to release on request (ROR) operation.)

A sample circuit for master block transfers is shown in figure 8. Here, a block transfer is triggered whenever the DMAC accesses a certain range of addresses. The SEQ signal could of course be generated in other ways.

SLAVE BLOCK TRANSFERS

VMEbus slaves that are capable of block transfers latch the bus address into a set of counters on the leading edge of ASN, and then increment the address for each data transfer. The SCB68172 can be used on such slave boards in accordance with revision C of the VMEbus specification.

The revision C specification introduces a limitation on block transfers, namely that a master is not allowed to continue a block transfer across a 256-byte boundary. This limitation has a number of advantages, including reducing the number of counter devices needed on slave boards, allowing straightforward use of page or static column modes on dynamic memories, providing periodic windows in a long block transfer in which higher-priority masters can gain bus control, and (effectively) preventing a block transfer from crossing from one slave board to another.

It is this last advantage that is of particular importance for the SCB68172. It means that VMEbus slaves can make a positive selection-decision after ASN goes low, and this decision will remain valid for the duration of the cycle even if it is a block transfer cycle.

In a block transfer which selects an SCB68172-based slave board, SLVSELN remains low through the block, until ASN goes high. The onboard slave logic then uses the data strobes to define each data transfer.

The data strobe and acknowledge signals are handled in a high- speed combinatorial fashion by the SCB68172 in both the master and slave roles. Block transfers are inherently faster on the VMEbus because the address need be passed and decoded only once, and because the slave can look ahead (pipeline) subsequent transfers in a block read cycle. With the SCB68172, this inherent speed advantage is augmented by the advantage of combinatorial over sequential (arbitrated) logic.

TRI-STATE ENABLE SWITCHING (MASTENN, VMEENN, DENN)

As a result of speed optimization of master/slave switching, some parts used in PMS applications may exhibit short high-going transients on MASTENN, VMEENN, and/or DENN, if requests for access to the shared slave(s) arrive closely in time from both the onboard master and the VMEbus master. These transients should pose no problem as long as the signals are used as intended (i.e., as tristate enables). The following points apply:

1. A transient will occur only when SLVSELN has been high for at least one clock period, and at least one clock period before a subsequent low on SLVSELN.
2. A transient will occur only if the current master/slave direction is maintained.
3. Low-going transients (which could cause tristate conflicts) do not occur.
4. Commonly such transients will be eliminated by external capacitance, and/or rejected by receivers on other parts. In any case the logic levels on signals controlled by these enable signals should not be affected.
5. Edge-sensitive use of these signals is inadvisable in a PMS application.

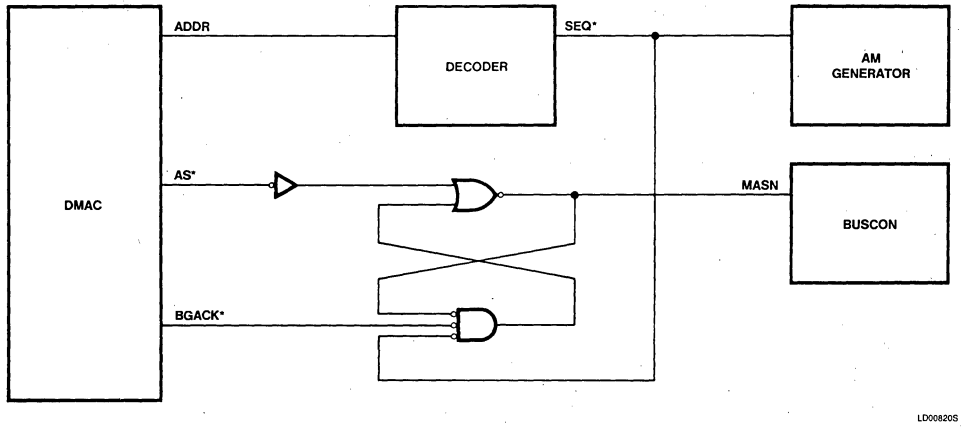
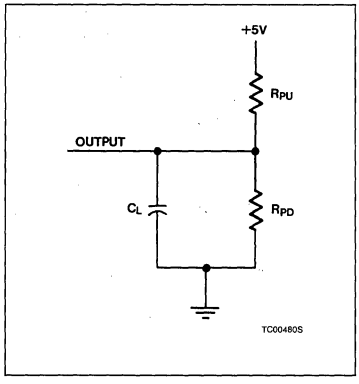


Figure 8. Master Sequential Mode with DMAC

TEST CONDITIONS

Unless otherwise noted, the following timing parameters are based on loading as follows:



SIGNALS	R_{PU}	R_{PD}	C_L
ASN, BRN, BBSYN, DTACKN, BERRN LDTACKN, LBERRN, LBGN, BGOUTN, VMEENN, SLVSELN, DSENN, DENN,	150	235	300
DDIR	2K	N/A	15
MASTENN	280	N/A	45
MASN	180	1K	45

2

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltage	−0.5 to +7.0	V
Input voltage	−0.5 to +5.5	V
Operating temperature range[2]	0 to +70	°C
Storage temperature	−65 to +150	°C

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V± 5%, T_A = 0°C to +70°C[3,4]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
DSI, ONBD, SLVN, VMEN, LBRN, RELSE, RESETN					
I_{IL}	Input low current	V_{IN} = 0.4V		−400	µA
I_{IH}	Input high current	V_{IN} = 2.7V		20	µA
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
ASN, MASN, BGINN, DTACKN, BERRN, LDTACKN, LBERRN					
I_{IL}	Input low current	V_{IN} = 0.4V		−400	µA
I_{IH}	Input high current	V_{IN} = 2.7V		20	µA
V_{TH+}	High-going threshold voltage		1	1.65	V
V_{TH-}	Low-going threshold voltage		0.8	1.15	V
$V_{TH+} - V_{TH-}$	Hysteresis		0.2		V
R/WN, CLK					
I_{IL}	Input low current	V_{IN} = 0.4V		−800	µA
I_{IH}	Input high current	V_{IN} = 2.7V		40	µA
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
BGOUTN, VMEENN, SLVSELN, DSENN, DENN, DDIR (low current totem pole)					
V_{OL}	Output low voltage	I_{OL} = 8mA, V_{CC} = 4.75V		0.5	V
V_{OH}	Output high voltage	I_{OH} = −0.4mA, V_{CC} = 4.75V	2.7		V
I_{OS}	Short-circuit output current	V_{OUT} = 0V	−15	−100	mA
MASTENN (high current totem pole)					
V_{OL}	Output low voltage	I_{OL} = 24mA, V_{CC} = 4.75V		0.5	V
V_{OH}	Output high voltage	I_{OH} = −2.6mA, V_{CC} = 4.75V	2.4		V
V_{OH}	Output high voltage	I_{OH} = −1mA, V_{CC} = 4.75V	2.7		V
I_{OS}	Short-circuit output current	V_{OUT} = 0V	−40	−100	mA
MASN (low current tristate)					
V_{OL}	Output low voltage	I_{OL} = 8mA, V_{CC} = 4.75V		0.5	V
V_{OH}	Output high voltage	I_{OH} = −0.4mA, V_{CC} = 4.75V	2.7		V
I_{OS}	Short-circuit output current	V_{OUT} = 0V	−15	−100	mA
I_{OZL}	Three-state-off leakage current, low level	V = 0.4V		−21	µA
I_{OZH}	Three-state-off leakage current, high level	V = 2.7V		20	µA
ASN (high current tristate)					
V_{OL}	Output low voltage	I_{OL} = 64mA, V_{CC} = 4.75V		0.5	V
V_{OH}	Output high voltage	I_{OH} = −7.8mA, V_{CC} = 4.75V	2.4		V
V_{OH}	Output high voltage	I_{OH} = −3mA, V_{CC} = 4.75V	2.7		V
I_{OS}	Short-circuit output current	V_{OUT} = 0V	−120	−300	mA
I_{OZL}	Three-state-off leakage current, low level	V = 0.4V		−60	µA
I_{OZH}	Three-state-off leakage current, high level	V = 2.7V		60	µA

DC ELECTRICAL CHARACTERISTICS (Continued)

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
LDTACKN, LBERRN, LBGN (low current open collector)					
V_{OL}	Output low voltage	I_{OL} = 8mA, V_{CC} = 4.75V		0.5	V
I_{OH}	Output leakage current	V = 5.5V		100	µA
DTACKN, BERRN, BRN, BBSYN (high current open collector)					
V_{OL}	Output low voltage	I_{OL} = 40mA, V_{CC} = min		0.4	V
V_{OL}	Output low voltage	I_{OL} = 70mA, V_{CC} = min		0.5	V
I_{OH}	Output leakage current	V = 2.7V		60	µA
I_{OH}	Output leakage current	V = 5.5V		250	µA
I_{CC}	V_{CC} Supply current	V_{CC} = max		160	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other conditions other than those indicated in the Electrical Characteristics section of this data sheet is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. Parameters are valid over specified temperature range.
4. All voltage measurements are referenced to ground (V_{SS}). For testing, all signals swing between 0.4V and 2.4V with a transition time of 10ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.

AC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V± 5%, T_A = 0°C to +70°C[3,4]

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS		UNIT	NOTES
			Min	Max		
Clock and general parameters						
1	9 – 18, 20	CLK cycle time (clk)	55		ns	
2	9 – 18, 20	CLK low time	25		ns	
3	9 – 18, 20	CLK high time	20		ns	
Asynchronous input set-up time to CLK high						
4	9, 10, 11, 12, 13, 18, 20	ASN, MASN low	15		ns	5
5	9, 10, 11, 13	ASN, MASN high	14		ns	5
6	9, 10, 11, 12, 13, 20	SLVN, VMEN low	22		ns	6
7	11, 17, 18	SLVN, VMEN high	20		ns	6
8	9, 10, 13	ONBD low	22		ns	6
9	18	ONBD high	25		ns	6
10	13, 16, 20	LBRN, RELSE, BGINN low	9		ns	5
11	14, 15	LBRN, RELSE, BGINN high	11		ns	5
12	11, 12, 13	DSI low (end of slave cycle)	16		ns	5
Asynchronous input hold time from CLK high						
13		ASN, MASN, DSI	0		ns	7
14		ONBD, VMEN	0		ns	8
15		LBRN, RELSE, BGINN	2		ns	7
Propagation, CLK high to:						
16	16	BGOUTN low	12	27	ns	
17	20	LBGN low	12	30	ns	
18		LBGN high	16	41	ns	
19	13, 16, 20	BBSYN, BRN low	17	37	ns	
20	13, 20	BBSYN, BRN high (C_L = 50)	20	47	ns	
		(C_L = 300)	40	68	ns	
21	9, 10, 13	ASN low	15	37	ns	
22	9, 10	ASN high	13	31	ns	
23	11, 12, 13, 18, 20	SLVSELN, VMEENN low	14	35	ns	
24	13, 18, 20	MASTENN low	15	38	ns	
Miscellaneous						
25		RESETN width low	6clk		ns	9
26	16	BGINN low to BGOUTN low	clk+10	2clk+35	ns	
27	16	BGINN high to BGOUTN high	3	10	ns	
28	9, 11	R/WN high to DSI high (start of read cycle)	10		ns	
29	9, 12	DSI low to RWN low (end of read cycle)	10		ns	

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS		UNIT	NOTES
			Min	Max		
Address decoding						
30	11, 12, 13, 20	ASN low to SLVN valid		clk–5	ns	
31	9, 10, 13, 18	MASN low to VMEN valid		clk–5	ns	
32	9, 10, 13, 18	MASN low to ONBD valid		clk–7	ns	
33	17	SLVN high after DTACKN low	14		ns	
34	9, 10	VMEN, ONBD valid after MASN high	6		ns	
35	9, 10	MASN high	15		ns	10
36	9, 10, 11	DSI low	20		ns	
37	11	ASN high	20		ns	10
VMEbus acquisition						
38	13	MASN low to BRN low	clk+15	2clk+45	ns	11
38A	13, 20	ASN low to BGINN low (early release by other master)	10		ns	
39	13, 20	BGINN low to BBSYN low	clk+15	2clk+45	ns	
40	13, 20	BBSYN low to BRN high	clk	clk+50	ns	
41	13	BGINN low to VMEENN low, DENN low (write)	clk+12	2clk+40	ns	12
42	13	ASN high to VMEENN low, DENN low (write)	11	31	ns	12,13
43	13	VMEENN low to ASN low	2clk–10	2clk+15	ns	14
43A	13, 16, 20	BBSYN or BGOUTN low to BGINN high	10		ns	
VMEbus master cycles						
44	9, 10	ASN high (successive VMEbus master cycles)	2clk–15		ns	14
45	9, 10	MASN low to ASN low (subsequent cycle retaining VMEbus control)	clk+13	2clk+45	ns	14
46	9, 10	ASN low to DSEN low	–4	5	ns	17, 18
47	10, 13	R/WN low to DDIR high	6	15	ns	
48	10, 13	DDIR high to DENN low (write)	2	7	ns	15
49	10	DTACKN and BERRN high to DENN low (write, 1st bus cycle or preceded by read)	7	21	ns	15
50	10	DENN low to DSENN low (write)	clk+10	2clk+40	ns	17
51	9, 11, 13	R/WN high to DDIR low	6	16	ns	
52	9	DDIR low to DENN low (read)	5		ns	16
53	9	DSI high to DENN low (read)	8	18	ns	16
54	9	DSI high to DSENN low (read)	8	18	ns	18
55	9, 10	DTACKN and BERRN high to DSENN low	6	19	ns	17, 18
56	9, 10	DTACKN or BERRN low to LDTACKN or LBERRN low	6	17	ns	
57	9, 10	LDTACKN or LBERRN low to DSI low or MASN high	0		ns	
58	9, 10	DSI low to DSENN high	7	16	ns	19
59	9, 10	MASN high to DSENN high	13	28	ns	
60	10	R/WN high to DSENN high (after a write)	5	14	ns	20
61	9, 10	MASN high to ASN high (unless early release)	clk+11	2clk+38	ns	
62	9	MASN high to DENN high (read)	13	28	ns	21
63	9	DSI low to DENN high (read)	7	16	ns	21
64	10	R/WN high to DENN high (write)	5	14	ns	22
65	9, 10	DSENN high to LDTACKN and LBERRN high	7	24	ns	23
66	9, 10	DTACKN and BERRN high to LDTACKN and LBERRN high	7	23	ns	23
VMEbus release						
67	14	BBSYN low	2clk		ns	
68	14, 15	BGINN high to BBSYN high	clk+18	2clk+70	ns	24
69	14, 15	RELSE high to BBSYN high	clk+20	2clk+72	ns	24
70	14	ASN low to BBSYN high (early release)	clk–25		ns	
71	14	MASN high to VMEENN high (early release)	8	20	ns	
72	14	MASN high to DENN high (early release, write)	8	20	ns	
73	14	DENN (write) and VMEENN high to ASN high (early release)	5	15	ns	
74	14	ASN high to ASN released (early release)	5	20	ns	
75	15	ASN high to BBSYN high (intercycle release)	clk–10		ns	
76	15	DENN (write) and VMEENN high to BBSYN high (intercycle release)	5	30	ns	
77	14, 15	BBSYN high to RELSE low	0		ns	

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS		UNIT	NOTES
			Min	Max		
Master to slave switching						
78	11, 13, 20	(External) ASN low to MASTENN high	10	28	ns	25
79	11, 13, 20	SLVN Low to MASTENN high	7	17	ns	25
80	11	SLVSELN high to MASTENN high	7		ns	25
81	11, 20	MASTENN high to VMEENN low	14	clk+36	ns	
81A	11, 13	VMEENN low to DDIR change	−5	+5	ns	
82	11, 13, 20	VMEENN low to SLVSELN low	clk−11	clk−5	ns	26
VMEbus slave cycles						
83	12	SLVSELN high (successive slave cycles)	2clk−5		ns	26
84	12	ASN low to SLVSELN low (already in slave state)	clk+12	2clk+40	ns	26
85	12, 13	R/WN low to DDIR low	6	14	ns	
86	12, 13	DDIR low to DENN low (write)	5	12	ns	27
87	12	LDTACKN and LBERRN high to DENN low (write)	7	20	ns	27
88	12, 13	R/WN high to DDIR high	5	14	ns	
89	11	DDIR high to DENN low (read)	2	7	ns	28
90	11	ASN low to DENN low (read)	clk+20	2clk+47	ns	28
91	11	DSI high to DENN low (read)	6	16	ns	28
92	11, 12, 13, 18	SLVSELN low and DSI high to LDTACKN or LBERRN low	0		ns	29
93	11, 12, 13, 18	LDTACKN or LBERRN low to DTACKN or BERRN low	10	20	ns	
94	12, 13	LDTACKN or LBERRN low to DENN high (write)	9	22	ns	
94A	11, 12, 13, 18	DTACKN or BERRN low to DSI low or ASN high	0		ns	
95	11	DSI low to DENN high (read)	7	16	ns	
96	11, 12, 18, 20	ASN high to SLVSELN high	13	28	ns	
97	11, 12, 13	DSI low to DTACKN and BERRN high				
		(C_L = 50)	17	37	ns	
		(C_L = 300)	37	60	ns	
98	11, 12, 13, 20	DSI low to LDTACKN and LBERRN high	0	35	ns	30, 32
99	11, 12	LDTACKN and LBERRN high to (next) DSI high	0		ns	30
Slave to master switching						
100	18	MASN low to VMEENN, DENN (VMEbus slave write) high	18	clk+47	ns	31
101	18	ONBD high to VMEENN, DENN (VMEbus slave write) high	8	24	ns	31
102		VMEN low to VMEENN, DENN (VMEbus slave write) high	18	21	ns	31
103	13, 18, 20	SLVSELN high to VMEENN, DENN (VMEbus slave write) high	0		ns	31
104	13, 20	DSI low (selected) to VMEENN, DENN (VMEbus slave write) high	14	clk+40	ns	31
105	13, 18, 20	VMEENN high to MASTENN low	24	clk+20	ns	
107	13, 20	MASTENN low to VMEENN low, DENN low (write, if next cycle on VMEbus)	3		ns	12,15
108	18	MASTENN low to SLVSELN low (if next cycle on board)	clk−13	clk	ns	33
Onboard cycles						
109	18	SLVSELN high (successive onboard cycles)	3clk+4		ns	33
110	18	MASN low to SLVSELN low (MASTENN already low)	clk+17	2clk+42	ns	33
111	18	MASN high to SLVSELN high	12	26	ns	

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS		UNIT	NOTES
			Min	Max		
DMAC-Type operation						
112	19	BBSYN high to MASN active (C_L = 50)	0	5	ns	34
		(C_L = 300)	−25	−15	ns	34
113	19	ASN high to MASN active	13	40	ns	34
114	19	ASN low to MASN low	10	25	ns	
115	19	ASN high to MASN high	6	16	ns	
116	19	ASN high to MASN released	12	32	ns	35
117	19	LBGN low to MASN released	5	12	ns	35
118	20	LBRN low to BRN low (if BBSYN released)	clk+15	2clk+45	ns	
119		LBRN low to LBGN low	clk+10	2clk+37	ns	36
120	20	BGINN low to LBGN low (ASN high)	clk+15	2clk+45	ns	
		(ASN low)	2clk+15	3clk+45	ns	
121	20	MASN low (output) to LBGN low	2clk+12		ns	
122	20	LBGN low to LBRN high	0		ns	
123	20	LBRN high to LBGN high	clk+14	2clk+48	ns	37
124	14, 15	LBRN high to BBSYN high (C_L = 50)	clk+18	2clk+55	ns	24
		(C_L = 300)	clk+35	2clk+75	ns	24
125	20	ASN high to LBGN high (selected)	2clk+18	3clk+50	ns	37
126	20	DSI low to LBGN high (selected)	2clk+20	3clk+52	ns	37

NOTES:

5. These set-up times guarantee recognition at a rising edge of CLK, but the device will operate correctly if they are not met. If the asynchronous input is changed between the set-up and hold times, the new state of the input may be recognized at this clock or the following clock.
6. These set-up times are required on the rising edge of CLK following the one on which ASN or MASN is first recognized low. If parameters 30, 31, and 32 are met, these parameters are automatically guaranteed.
7. These hold times guarantee (continued) recognition of the signal state at a rising edge of CLK, but the device will operate correctly if they are not met.
8. These hold times are required on the rising edge of CLK preceding the one on which MASN is first recognized high. Parameter 34 provides a more straightforward requirement which guarantees these times.
9. This parameter applies after V_{CC} and the clock signal are both within the specified limits.
10. These minimum times are to guarantee recognition. Operation will be limited by 44, 83, and 109 if the strobe is high for less than 2clk.
11. BRN is driven low, and this acquisition sequence applies, only if BBSYN is high.
12. VMEENN is driven low only when 41, 42, and 107 have been met.
13. Applies to ASN of VMEbus cycle which does not select this board as a slave.
14. ASN goes low when 43, 44, and 45 are met. 44 is not applicable on the first cycle after acquiring the VMEbus.
15. In a write operation, DENN goes low when 41, 42, 48, 49, and 107 are met. 49 does not apply for subsequent cycles in a series of writes if R/WN is held low throughout.
16. In a read operation, DENN goes low when 52 and 53 are met.
17. In a write operation, DSENN goes low when 46, 50, and 55 are met. 55 is significant only for subsequent cycles in a series of writes with R/WN held low throughout.
18. In a read operation, DSENN goes low when 46, 54, and 55 are met.
19. DSENN goes high when either 58 or 59 is met.
20. Applies only if R/WN remains low after a write cycle, so that DSENN goes low again.
21. In a read operation, DENN goes high when either 63 or 64 is met.
22. In a write operation, DENN goes high when either 64 or 71 is met.
23. LDTACKN and LBERRN go high when either 65 or 66 is met.
24. BBSYN is always released in response to BGINN, RELSE, and LBRN all high. However, if this condition is detected during a clock period in which the decision to change ASN is made, the release of BBSYN is delayed one clock period so that 70 or 75 is met.
25. MASTENN goes high when 78, 79, and 80 are all met.
26. SLVSELN goes low when 82, 83, and 84 (as applicable) are met.
27. In a write operation, DENN goes low when 86 and 87 are met.
28. In a read operation, DENN goes low when 89, 90, and 91 are met.
29. The onboard slave(s) should wait for both SLVSELN and DSI before driving a response.
30. Since BUSCON itself terminates DTACKN and BERRN when DSI goes low, the onboard slaves must meet this requirement to assure that a "lingering" LDTACKN or LBERRN is not presented as DTACKN or BERRN when the next DSI occurs. 99 is the real requirement – 98 max is derived from it, plus the 40nsec minimum high time of VMEbus data strobes and an allowance for receiver skew.
31. VMEENN goes high only when 100, (101 or 102), 103, and 104 are met. 104 applies only if a VMEbus slave cycle (with this board) is ending.
32. The onboard slave(s) must meet this requirement so that the local response is not inadvertantly presented to the onboard master.
33. SLVSELN goes low when 108, 109, and 110 (as applicable) are met.
34. MASN is driven out of Hi-Z state only when 112 and 113 are met.
35. MASN is released to Hi-Z state only when 116 and 117 are met.
36. The max figure applies only if BUSCON has kept VMEbus control (i.e., if BBSYN is low).
37. LBGN goes high only when 123, 125, and 126 are met, but 125 and 126 apply only if a VMEbus slave cycle (with this board) is in progress.

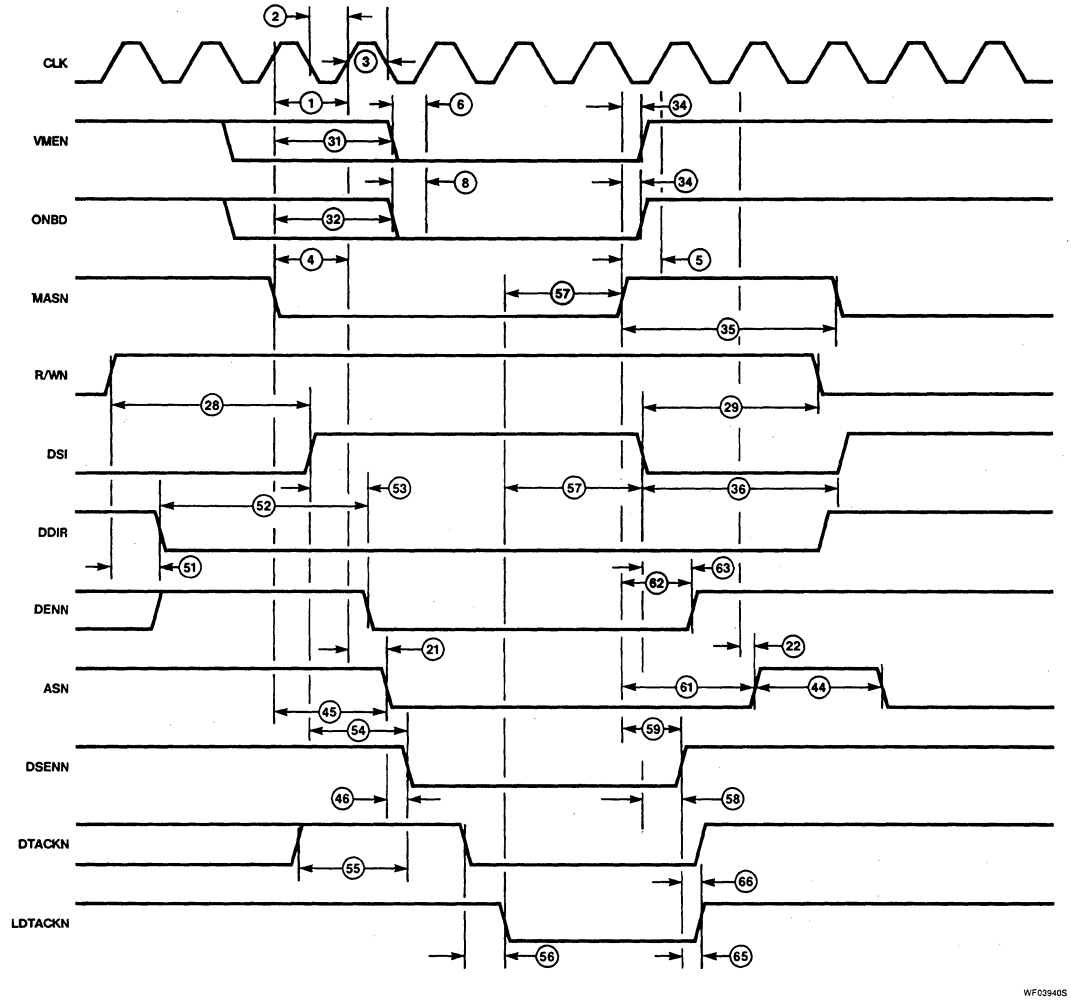


Figure 9. Read Cycle From Onboard Master to VMEbus Slave (Already in Control of VMEbus)

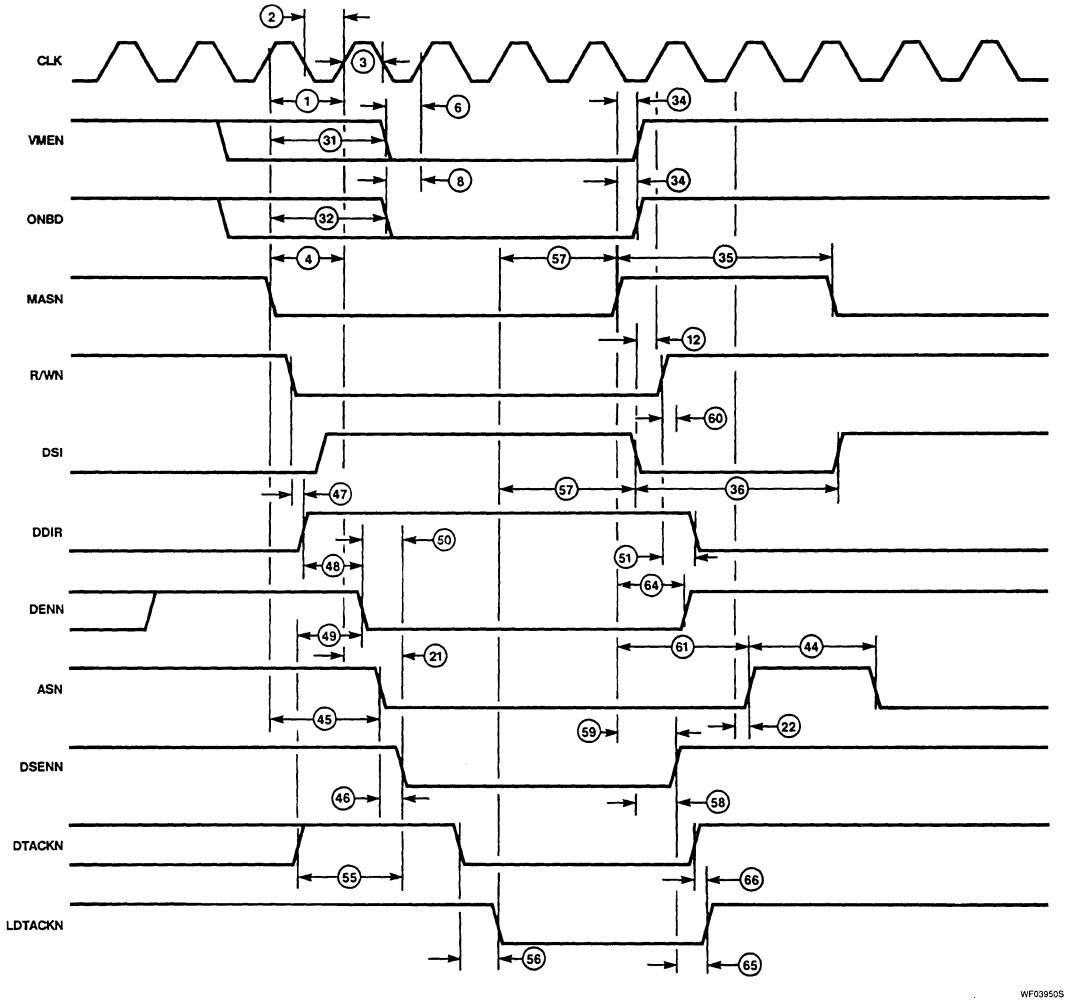


Figure 10. Write Cycle From Onboard Master to VMEbus Slave (Already in Control of VMEbus)

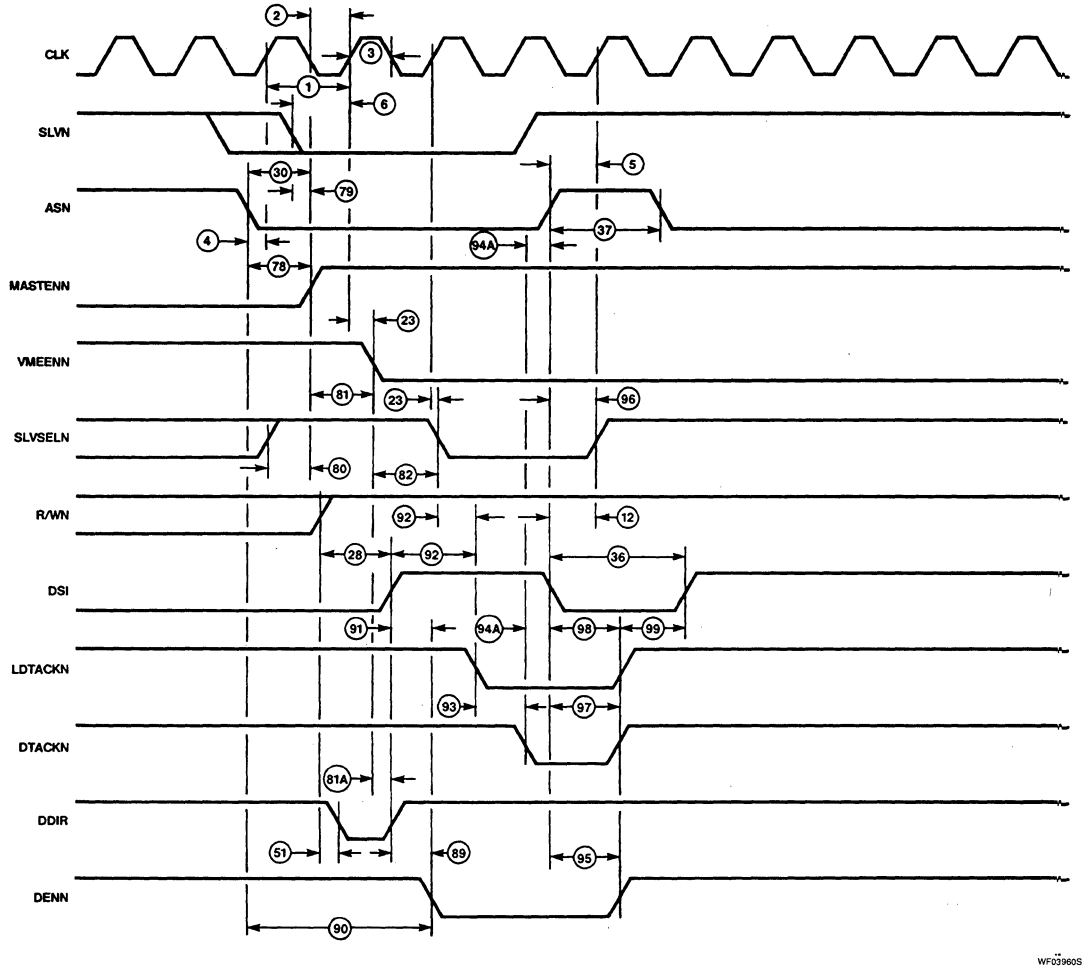


Figure 11. Slave Read Cycle (Starting in Master Mode)

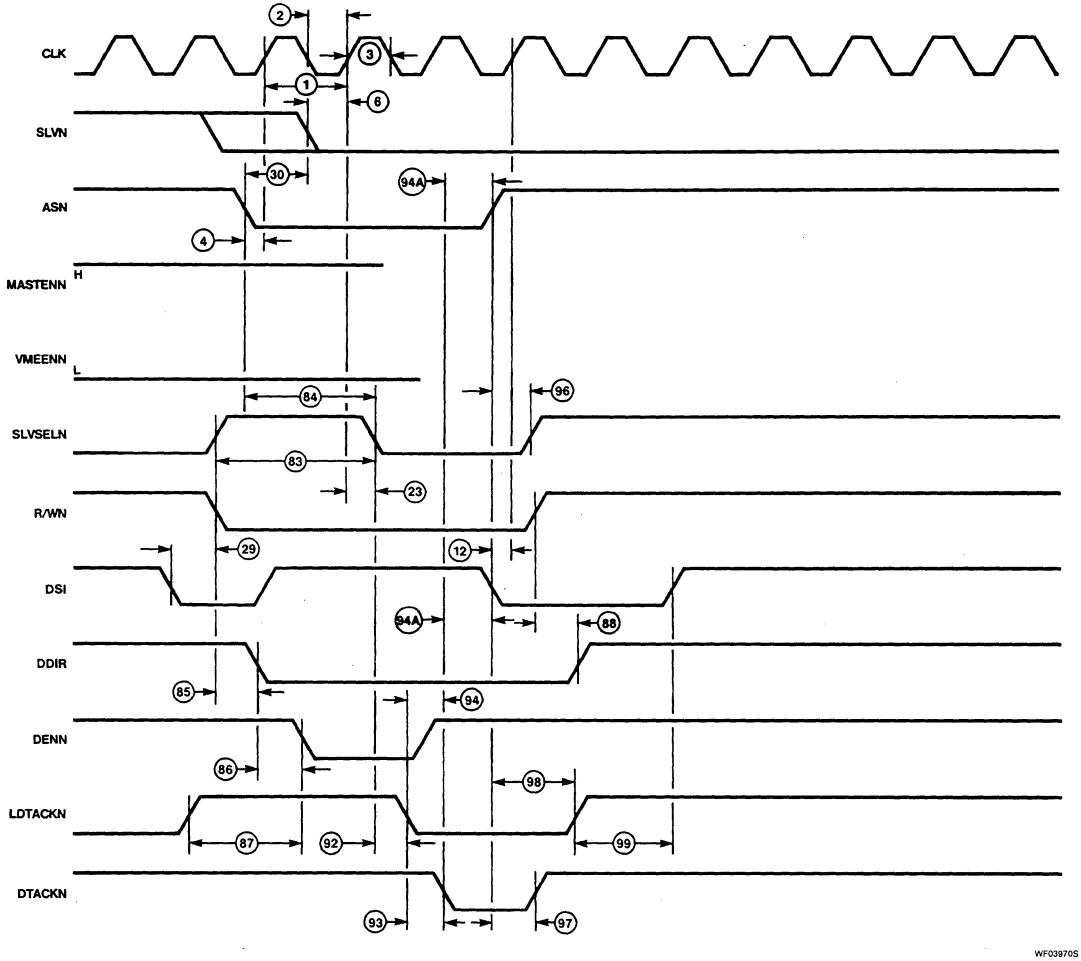


Figure 12. Slave Write Cycle (Starting in Slave Mode)

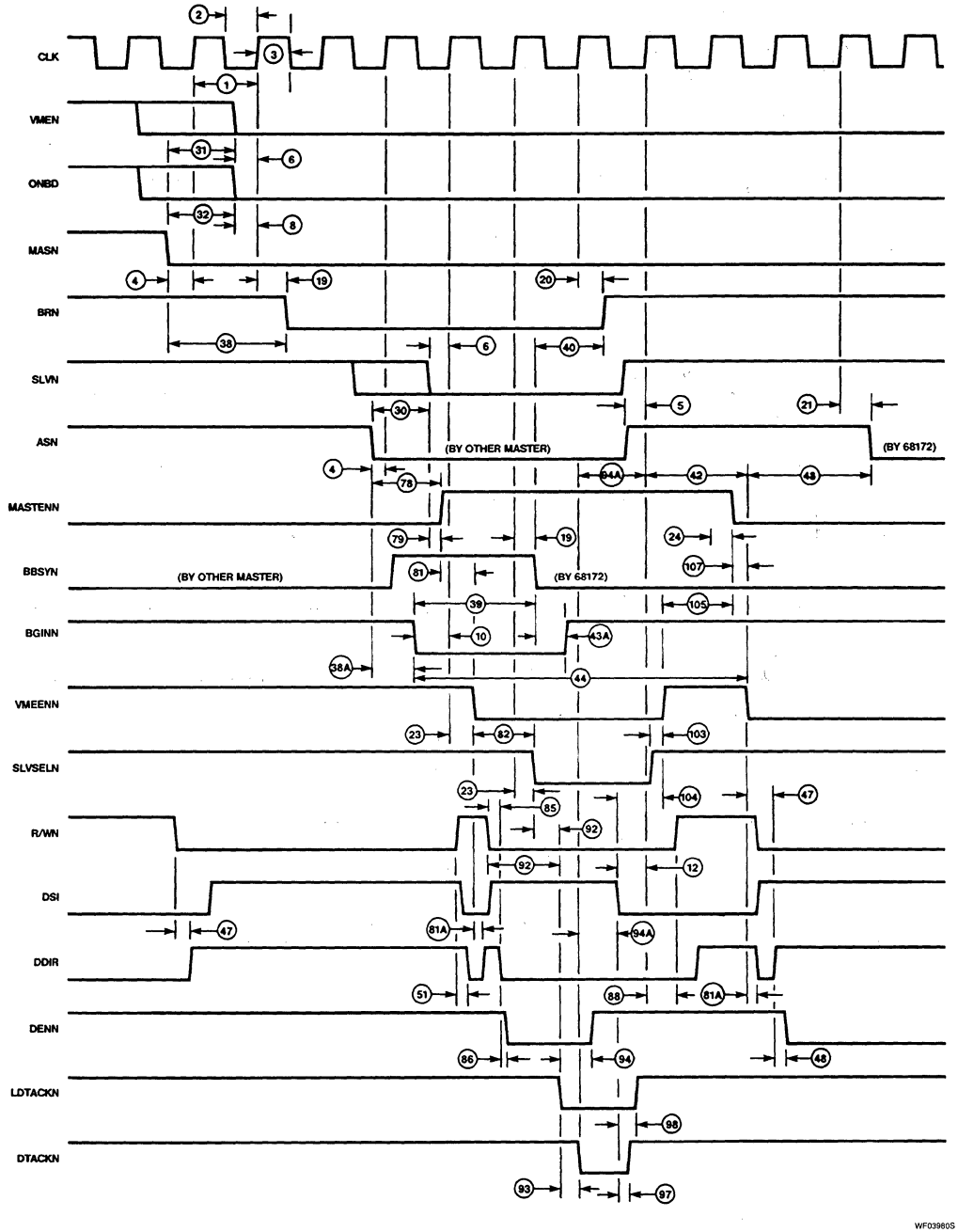


Figure 13. VMEbus Acquisition, Including Slave Write Cycle

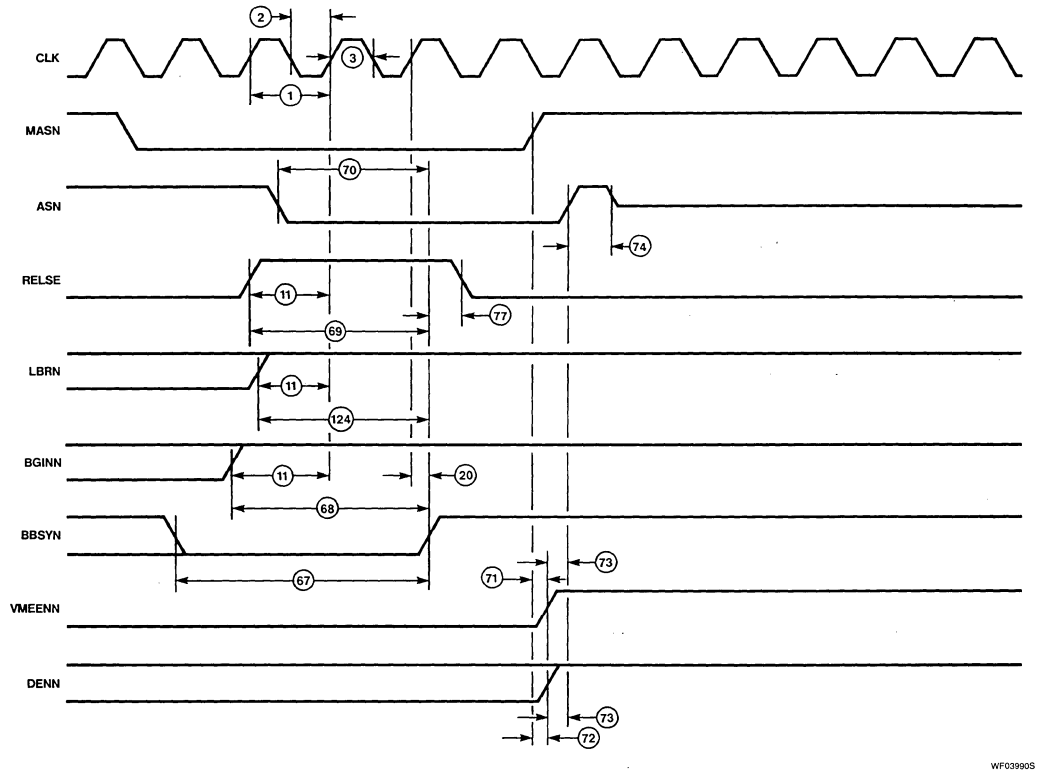


Figure 14. "Early" VMEbus Release

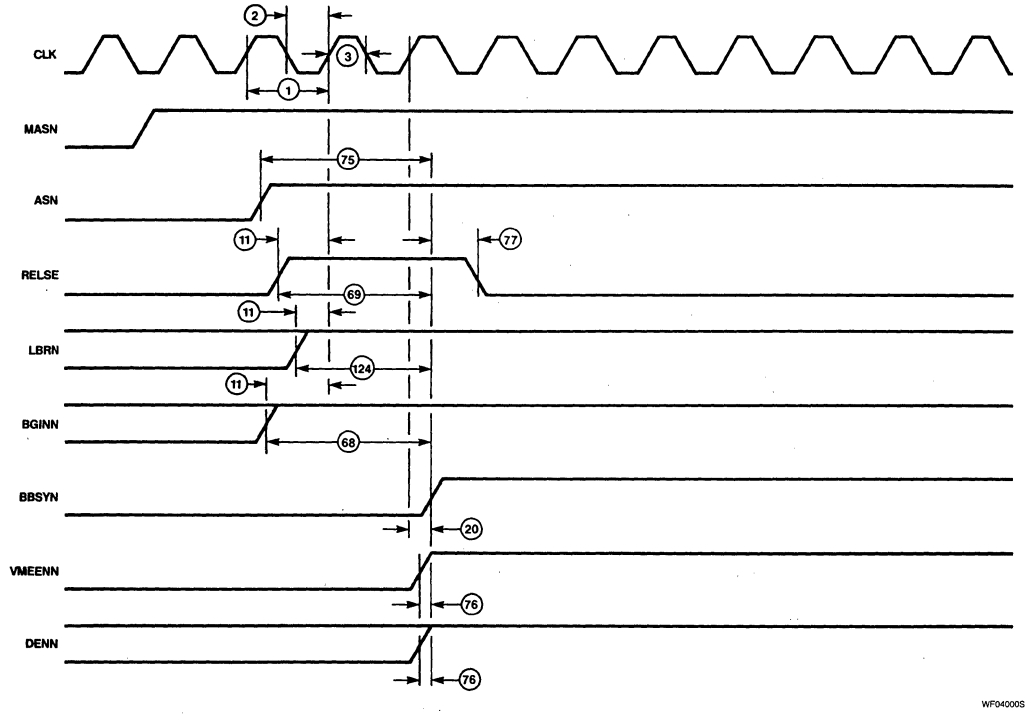


Figure 15. Intercycle VMEbus Release

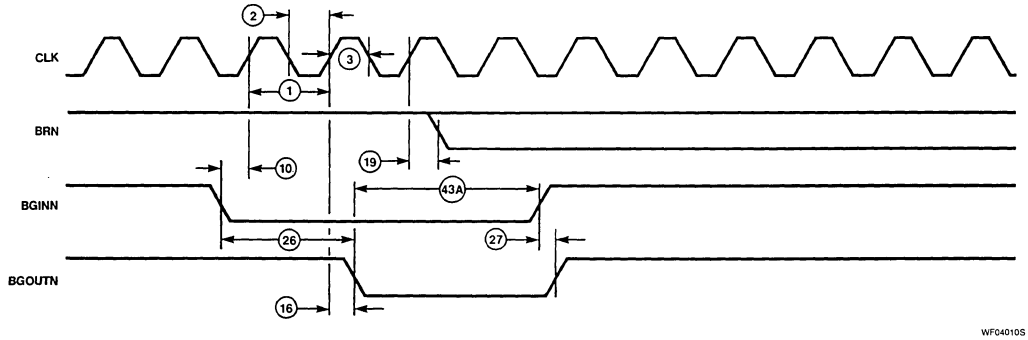


Figure 16. Passing a Bus Grant, Including Earliest Possible Bus Request Thereafter

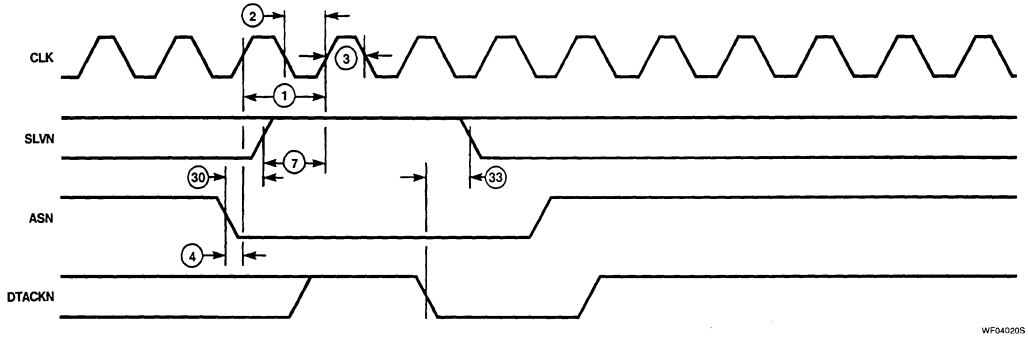


Figure 17. Non-selecting Cycle on VMEbus

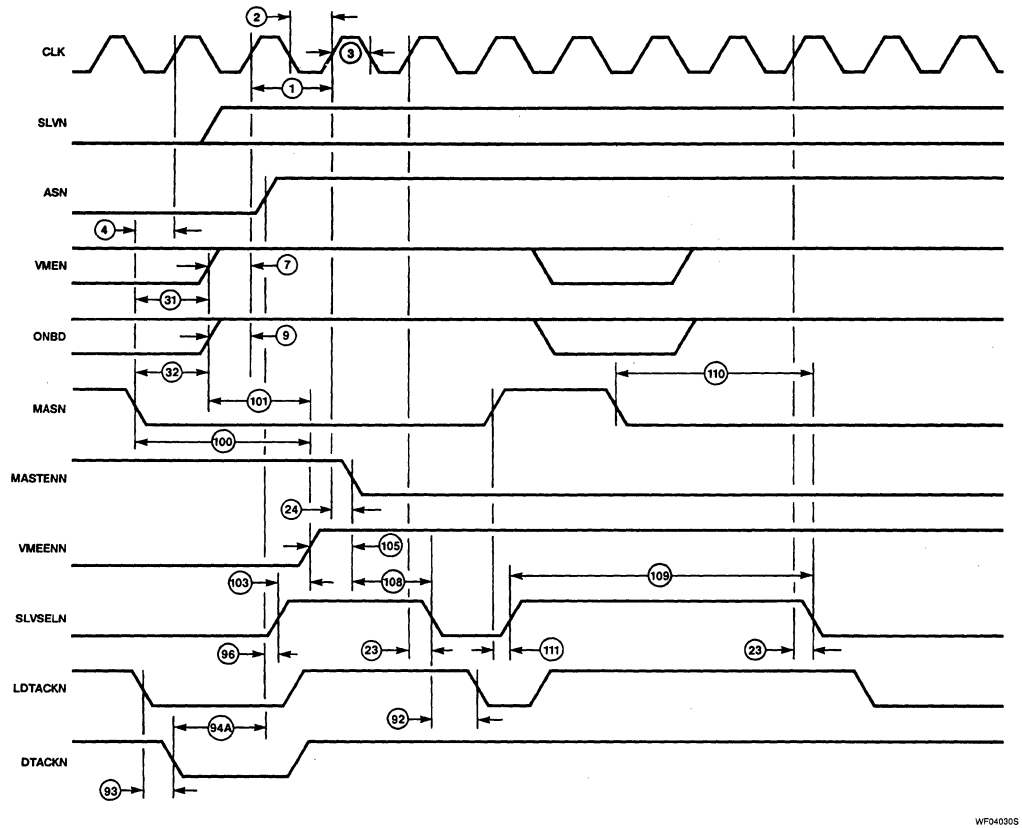


Figure 18. Two Onboard Cycles Following a Slave Cycle

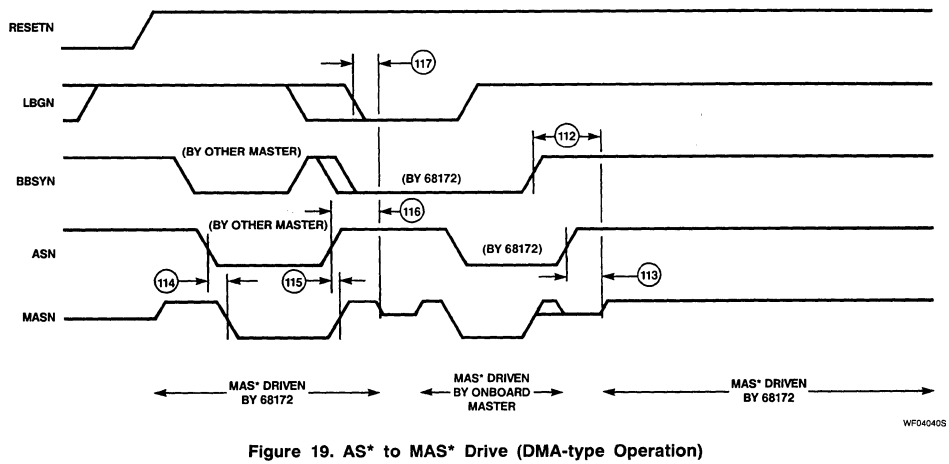


Figure 19. AS* to MAS* Drive (DMA-type Operation)

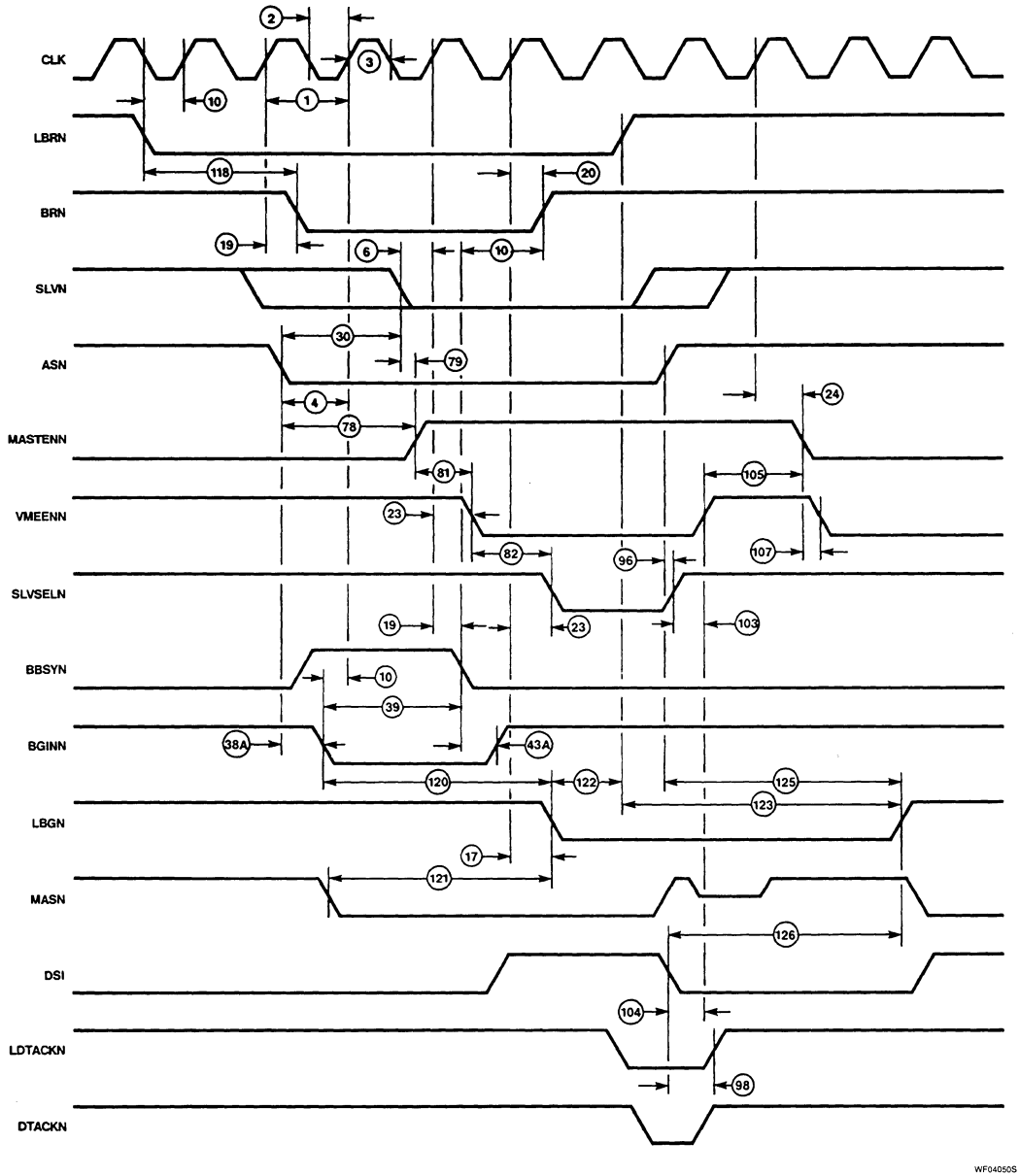


Figure 20. DMA-type VMEbus Acquisition, With Slave Cycle From Another VMEbus Master

Signetics

SCB68173
VMSbus Controller (VMSCON)

Objective Specification

Microprocessor Products

THE VMSbus

THE VMSbus provides a secondary control and data path in the backplane of VMEbus systems, and can also be used in other backplanes and in 'intercrate' (short inter-system) applications. Providing such capabilities as message passing, generalized event/interrupt communication, discrete signal control, and synchronization/semaphore operations, the VMSbus is especially significant for multiprocessor and/or fault-tolerant systems. With a 2.9Mbit/second data rate in a backplane environment, collision-free self-arbitration during frame transmission, and variable message priority, the VMSbus offers a high-speed, deterministic path for the guaranteed delivery of short, urgent messages.

SCC68173 VMSbus CONTROLLER

The Signetics SCC68173 VMSbus Controller (VMSCON) is a register-oriented peripheral device that includes a collection of the basic functional modules described in the VMSbus specification. It interfaces to the VMSbus via the SCB68171 VMSbus Interface, and to a standard 68000 family bus or other parallel data buses. The SCC68173 includes the following VMSbus functions:

Qty	Function	Functional Modules
1	Controller	Header sender plus frame monitor
1	Talker	Data sender plus header receiver
1	Listener	Data receiver plus header receiver
4	Flags	Status flip-flop plus header receiver

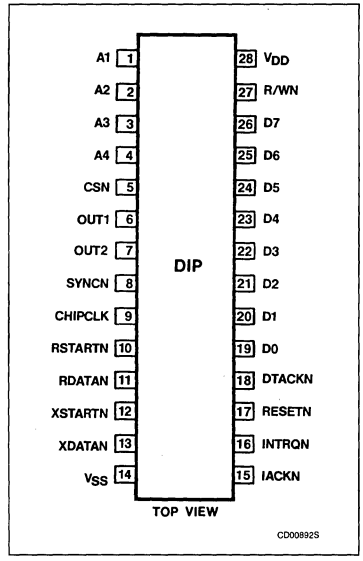
These functions are fully programmable so they can be used in a wide variety of combinations. The flags control two discrete output signal pins. All functions are designed to be driven via vectored interrupts.

FEATURES

- **Header sender paired with frame monitor**
- **Data sender with four-byte buffer**
- **Data receiver with four-byte buffer**
- **Four programmable flag modules**
- **Register blocks with sequential access**
- **Seven maskable interrupt sources**
- **Pending interrupt register**
- **Input signal can control frame transmission**
- **Interrupt vector base register**
- **Multiplexed header receivers**
- **Two direct control outputs**
- **68000 bus compatible**

The SCC68173 is designed to be used with the SCB68171 which interfaces to the VMSbus clock (SERCLK) and data (SERDATN) lines. It also provides all the 68000 equivalent parallel bus controlling signals with an 8-bit bidirectional data bus. A set of command, status, and data registers are selected by the appropriate lower address bit lines.

PIN CONFIGURATION



VMSbus Controller (VMSCON) SCB68173

ORDERING CODE

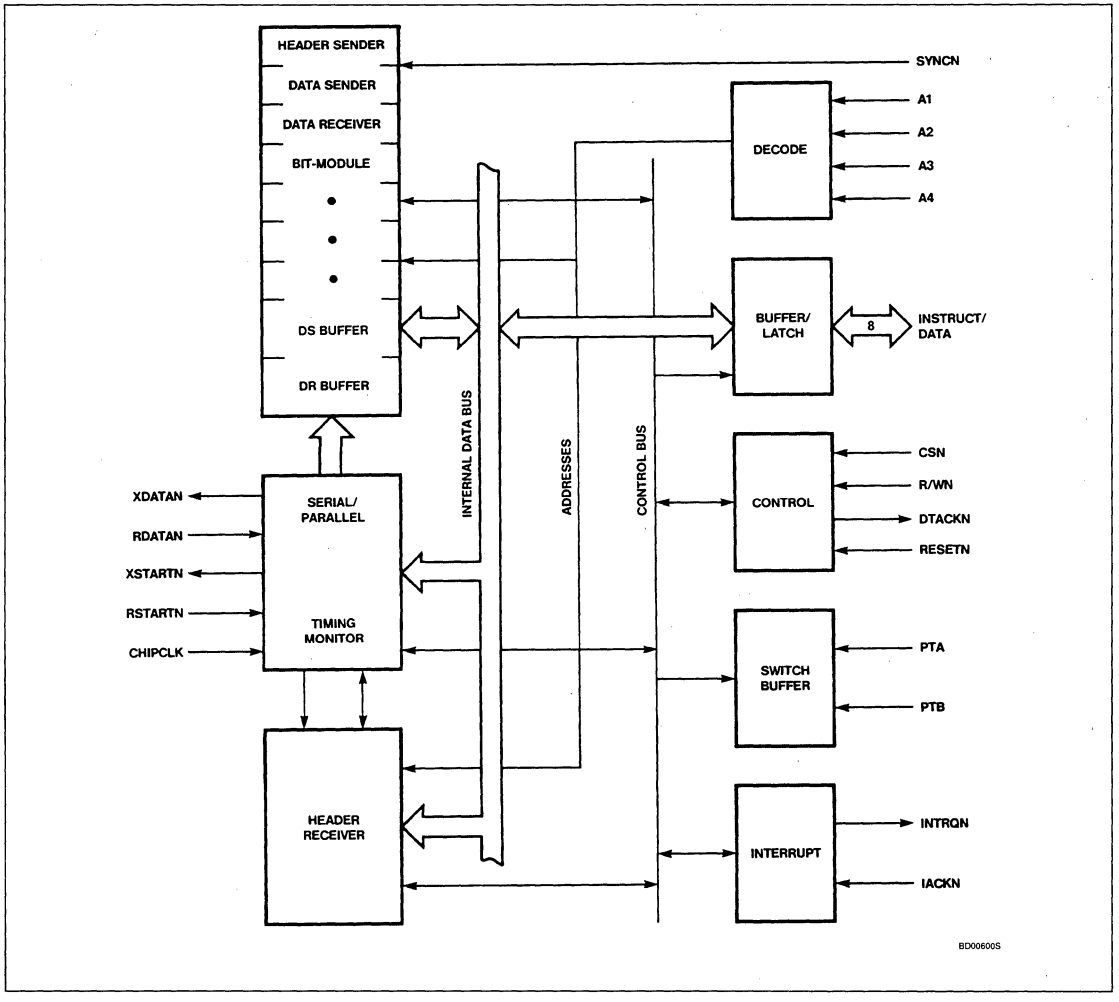
PACKAGES	V_{CC} = 5V ±5%, T_A = 0°C to 70°C
Ceramic	SCC68173CxI28
Plastic	SCC68173CxN28

BLOCK DIAGRAM

The SCC68173 VMSbus Controller consists of five major sections. The serial/parallel and parallel/serial conversion block controls all the timing of incoming and outgoing frames, does all on chip timing, and monitors the VMSbus. The header receiver block contains the programmed module addresses and incorporates the comparator logic to identify arriving S- and R-fields of a header subframe.

The register array can be loaded to control the header sender and the data sender, data receiver and bit modules.

Included in the array is a data sender buffer and a data receiver buffer. The interrupt control block provides request and acknowledge logic. The parallel bus interface includes four address lines, an 8-bit data bus, and bus control signals. There are also two direct control output lines available (OUT1, OUT2).



VMSbus Controller (VMSCON) SCB68173

SCB68171 VMSbus INTERFACE

The VMSbus serial clock (SERCLK) provides for well-defined sampling times called S1 and S2. The Signetics SCB68171, a high speed bipolar interface, releases the VMSbus Controller from the VMSbus timing requirements. It also serves as a buffering device for multiple onboard VMSbus Controllers and provides the high current required on the SERDATN line. Figure 1 shows the VMSbus interface and controller interconnect. A timing diagram is shown in figure 2.

As the SCC68173 sends data or prepares to do so, it also monitors the RDATAN and RSTARTN inputs for conflict with other transmitters. The XDATAN and XSTARTN outputs provide the signal timing required by the SCB68171. As an immediate response to a detected misplaced start bit, the VMSbus controller generates XSTARTN and XDATAN low simultaneously, which makes the SCB68171 drive SERDATN low immediately, to "jam" the VMSbus.

REGISTER BLOCKS

The SCC68173 has four address line inputs, A4 – A1, for a total of 16 register addresses. Of these, 10 are simple 8-bit registers while the other six address values (2, 3, 5, 10, 14, and 15) reference register blocks. Each register block has an internal address pointer that is used to access the "current" 8-bit register from among the several registers in the block. After the current register is read or written, the internal address pointer is incremented to point to the next register in the block. After the last register in a block is read or written, the pointer cycles back to the first register in the block.

Accessing any SCC68173 register or register block resets the internal address pointers of all other register blocks, to reference the first register in the block.

VMSbus OVERVIEW

Information is transmitted in frames on the VMSbus. A frame is a sequence of consecutive bits in a prescribed arrangement or format. Figure 3 shows the three types of frames defined for the VMSbus. The first 26 bits, called the header subframe, are the same for all three frame types, and are sent by a VMSbus functional module called a header sender.

The header subframe contains two 10-bit fields called the S and R addresses. Either or both of these addresses may select one or more VMSbus functional modules called header receivers. Header receivers may be paired with other VMSbus functional modules called data senders and data receivers, and/or may control status bits (flipflops). The combined functions are called talkers, listeners, and flags, respectively.

The first frame in figure 3 shows the case where one (or more) of the header receivers selected (by either the S or R address) is not ready for the frame. In this case, the not-ready header receiver cancels the frame by driving the value 111 in the frame type field. Such cancelled frames end immediately after the frame type field.

The second frame in figure 3 shows a non-data frame. This type of frame results when the S address does not select a talker. In normal operation, the R address of a non-data frame does not select a listener. The frame type field contains zero and is immediately followed by the frame status field, which indicates whether flags were selected by the S and R addresses respectively, and whether there were any problems with the frame.

The third and last frame in figure 3 shows a data frame. In this case, the S address

PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
XDATAN, XSTARTN	13,12	O	**Transmit Data Control:** Open drain outputs. Set up during CLK low time as follows: XDATAN XSTARTN H H Zero bit (or not transmitting a frame) L H One bit H L Start bit Also, if RSTARTN goes low during a frame (misplaced start bit), the SCC68173 drives both of these signals low during the CLK high time, to force SERDATN low and jam the bus.
RDATAN	11	I	**Serial Data Input**
RSTARTN	10	I	**Start Detect Input:** Active low whenever a start condition is detected on the VMSbus.
CHIPCLK	9	I	**Clock:** The CLK signal is derived from SERCLK – the serial bus clock – by the bipolar interfacing device.
RESETN	17	I	**Reset:** Active low signal, which clears most of the controller's logic.
OUT1,OUT2	6,7	O	**Control:** These outputs are controlled by on-chip bit modules for direct control purposes.
DTACKN	18	O	**Data Transfer Acknowledge:** Open drain, active low.
CSN	5	I	**Chip Select:** This active low signal activates transfer on D0 – D7.
A1 – A4	1 – 4	I	**Address Lines:** These lower address lines select on-chip registers.
INTRQN	16	O	**Interrupt Request:** Active low, open drain output which signals the CPU that one or more of seven maskable interrupting conditions is true.
IACKN	15	I	**Interrupt Acknowledge:** Active low input indicating an interrupt acknowledge cycle.
D0 – D7	19 – 26	I/O	**Data Bus:** Bidirectional 3-state data bus used to transfer commands, data, and status.
R/WN	27	I	**Read/Write:** Read/write control.
SYNCN	8	I	**Synchronize:** Input which controls frame transmission by the header sender module.
V_{DD}	28	I	**+ 5V ± 5% power input**
V_{SS}	14	I	**Power ground return**

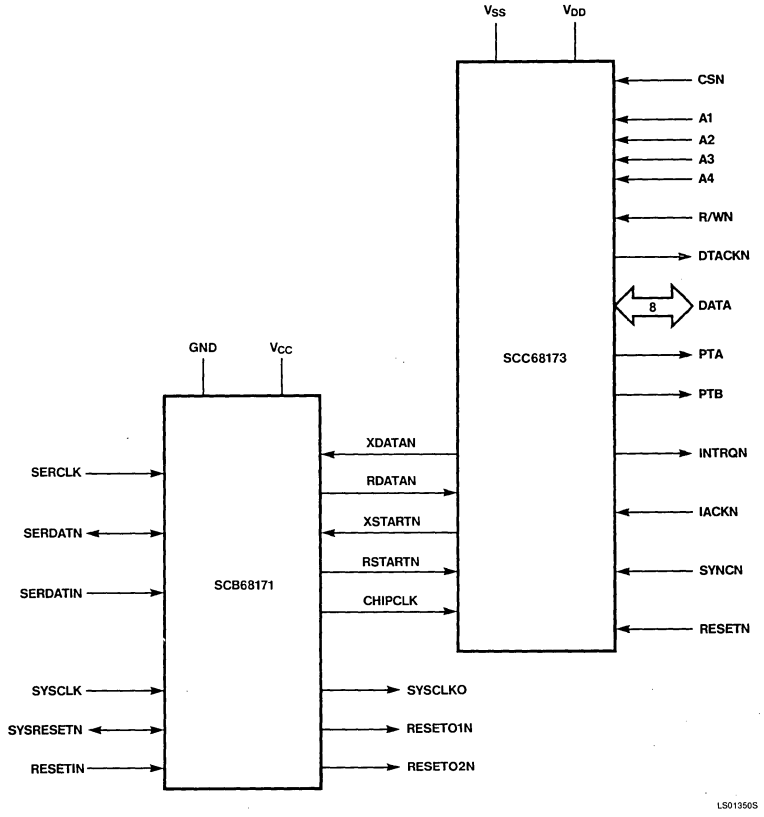


Figure 1. VMSbus Interface and Controller Interconnect

selected a talker function. The data sender sends one of the values 001 – 110 in the frame type field, indicating the number of data bytes which follow. In normal operation, the R address selects a listener function, which receives the data. The data is then followed by the frame status field, which indicates if there were any problems with the frame.

As shown in figure 3, every frame is followed by a single bit called a jam bit. The jam bit should always be high/zero. This forms a guard against desynchronization among various VMSbus modules, as to where frames begin and end.

All frames begin with a special unique start bit. Each frame monitor module tracks every VMSbus frame – if it detects a start bit within a frame, it jams the VMSbus with a string of 512 low/one bits. Since the longest VMSbus frame is 287 bits, this jam sequence is sure to affect the jam bit of any frame currently in progress.

If any VMSbus module detects a jam bit low/one, it ignores the preceding frame. The jam sequence also serves to resynchronize all VMSbus modules, which wait until it is over before transmitting any more frames.

Since a VMSbus typically includes a number of header senders, there must be a way to control when they start sending frames. As already noted, frame monitor modules track the progress of every VMSbus frame. If a header sender is signaled to start a frame while its paired frame monitor is tracking a frame, it waits until the current frame is completed.

VMSbus data is wire-ORed, so that if several modules send data in the same bit time, the result is the logical OR of their data. This fact is used to provide bus arbitration as frames are sent. Whenever a module sends a zero/high bit, it also senses the result bit on the bus. If it senses a one/low, it has lost the arbitration and stops sending data. Typically, a header sender waits until the current frame is over, and then tries to send its header subframe once again.

This arbitration method ensures that data is never lost or garbled because of contention for use of the VMSbus. If several modules send data, the data with the highest binary value gets through without distortion. Arbitration is always used by header senders; its use is optional for data senders, depending on the DSAE bit in the header subframe.

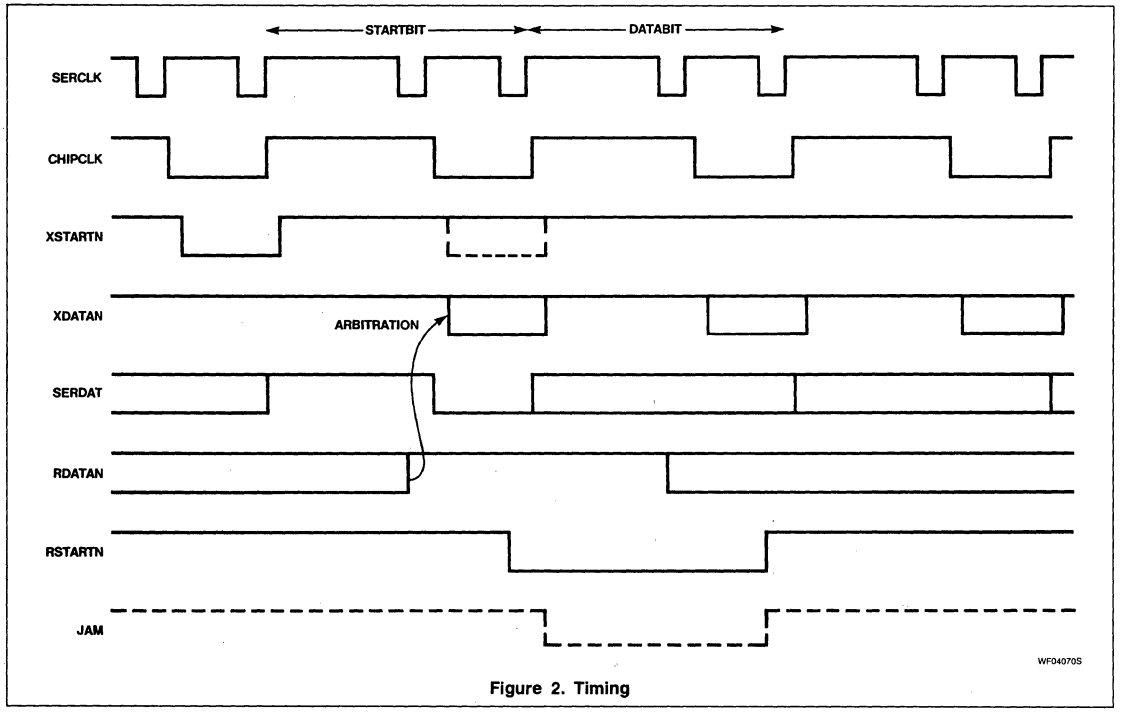


Figure 2. Timing

CONTROLLER FUNCTION (HEADER SENDER/FRAME MONITOR)

Before a header subframe can be sent, the information in it must be programmed into the SCC68173. This includes the priority field (in register A0), the S and R addresses (register blocks A2 and A3), and the DSA bit (in register A0).

If XRS in A0 is zero, then the S address is taken from register block A2 and the R address is taken from A3. If XRS is one, then the S address is taken from A3 and the R address from A2. This feature can be useful, for example, to send set and reset frames without reloading A2 and A3.

When the necessary information is loaded, the SCC68173 can be commanded to send the header subframe by software (setting the HSA bit in A0) or by its onboard hardware (SYNCN input low after software has set the SCE bit in A0). After any current frame in progress has completed, the controller function then tries to send the header subframe. This attempt may have one of several results:

1. The Controller may lose VMSbus arbitration while trying to send the header subframe. In this case it always sets the LAB bit in A1. Its other actions depend on the RTE bit in A0 and on the retry counter which can be loaded by writing A1:
   a. If RTE is zero, the SCC68173 clears the HSA bit in A0. Note that in any case where the SCC68173 clears the HSA bit in A0, it stops trying to send the header subframe, and the header sender becomes inactive. Also, if HS in A13 is set, then it sets HS in A12 and requests interrupt on IRQN.
   b. If RTE is one, the SCC68173 decrements the 4-bit retry counter. If this operation does not result in a borrow, the controller function waits for the current frame to complete, and then tries to send the header subframe again. The value programmed into the retry counter (low-order bits when writing A1) should be the maximum number of retries to attempt.
   c. If RTE is set, and decrementing the retry counter results in a borrow, the controller function clears the HSA bit in A0.
2. If the header sender wins the VMSbus arbitration through sending the HSVAL bit as low/one, it clears the LAB bit in A1 and stops sending. The frame monitor continues to track the resulting frame, through the final jam bit. If the jam bit is "one", the controller sets the jam bit in A1 and clears the HSA bit in A0. (Jam should be the overriding result, and should always be reported by the controller hardware. If this result does not preclude other bits in A1, at least jam should be the first thing checked by the software.)
3. The resulting frame may be cancelled. In this case the SCC68173 sets the CNC bit in A1 and clears the HSA bit in A0.
4. A non-data frame may result. In this case SCC68173 sets the SNT bit in A1, clears the DFR bit in A1, saves the frame status in A1 and clears the HSA bit in A0.
5. A data frame may result. SCC68173 action in this case is the same as for a non-data frame, except that it sets the DFR bit in A1.

In either case 4 or 5 above, software should check the DFR and frame status bits in A1 to see that the expected set of header receivers was selected and that the frame was otherwise correct. For cases 1 and 3 (lost arbitration and cancelled frame), the frame status field contains the monitored priority field from the bus, rather than the frame status field.

The priority, DSAE, XRS, and RTE bits can be set in A0 by the same command which sets HSA or SCE. When the SCC68173 clears HSA in A0, it does not clear SCE. To prevent additional transmissions in case SYNCN is still low, the SCC68173 requires that SYNCN

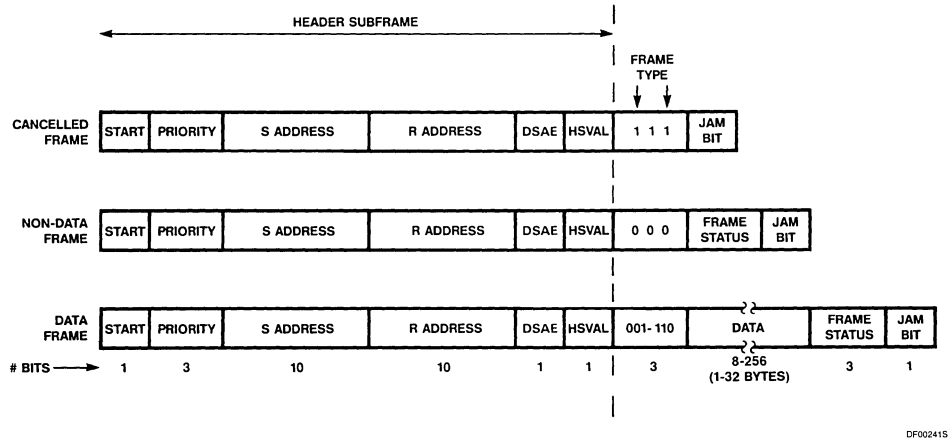


Figure 3. VMSbus Frames

go high again before it is enabled to start another frame.

System software can abort a series of retries by rewriting A0 with HSA cleared. It should then check HSA in A0 until it is zero, because if HSA is rewritten while a VMSbus frame is in progress, the new state does not become effective until the end of the frame.

HEADER RECEIVERS

There are six header receivers in the SCC68173, called HR0 through HR5. Each of them can compare a 10-bit address, assigned to it by system software, to the S and R addresses in frames on the VMSbus. When an address matches, a header receiver signals its associated module that it is selected by the frame. It is important to note that header receivers and their associated modules respond identically to VMSbus frames that are initiated by their on-chip header sender, and to frames initiated by other header senders.

Addresses are assigned to header receivers by writing register block A5. The first (reset) byte in register block A5 corresponds to the less-significant 8 bits of HR0's address, while the last (12th) byte in A5 corresponds to the 2 high-order bits of HR5's address.

Note that A5 is a write-only register block. This means that if the address assigned to a particular header receiver is to be changed, all the addresses, up to and including the desired one, must be rewritten. In general this implies that system software must maintain a memory table containing the image of register block A5.

The six header receivers select specific modules in the SCC68173 as controlled by register A6. The selection coding is shown in table 2. While each header receiver selects one module, the same address can be programmed into more than one header receiver. Each module may be selected by 0, 1, 2, or 3 header receivers.

Register A4 controls which header receivers are enabled. A module can only respond to VMSbus frames if one or more header receivers (that are enabled in A4) are programmed to select it, in A6.

A header receiver should be disabled in A4 before its selection criteria are changed in A5 and/or A6.

The six modules that are selected by the header receivers are one data sender, one data receiver, and four flags. They are described separately in the following sections. The data sender can be selected only by a match in the S address, while the data receiver can be selected only by a match in the R address. Flags can be selected by a match in either address.

DATA SENDER

The function of this module is to send data on the VMSbus (to a data receiver module) when it is selected by the S address in a frame. If the selected data sender is on the same board (in the same SCC68173) as the header sender that initiated the frame, the overall VMSbus operation is called a write frame. If the selected data receiver is on the same board (in the same SCC68173) as the initiating header sender, the operation is a read frame. If both the data sender and data receiver are on different boards (in different SCC68173's), the operation is called a move.

System software should set up a data sender for transmission as follows:

1. If necessary, establish the data sender's (S) address and the link to a header receiver via A5 and A6, and then enable the header receiver via A4.
2. If necessary, enable the data sender interrupt in register A13.
3. Write the data to be sent into register block A14.
4. Write register A8 with the frame type code for 1, 2, or 4 bytes, plus the DSE bit and optionally the DSM bit (see the discussion that follows on writing A8).
5. For a write frame, program the header sender to initiate a frame with the data sender's address in the S field.

Because the DSE bit in A8 can be set and cleared by system software and also cleared by a VMSbus frame, a special technique must be used by system software in writing A8. For bits DSE, DSM, and DST, writing a '1' in each bit position indicates that the bit should be written with the data (0 or 1) contained in the high-order bit (labeled Write Data in table 1). Also, the data sender frame type bits are written from the three least significant bits only if the next more significant bit is a '1' (labeled Write Enable in table 1).

When the data sender is selected by the S address in a header subframe, it begins operation by sending a code in the frame type field which directly follows the header subframe. If bit DSE in register A8 is zero, indicating that system software has not set up current data and enabled the data sender, it

Table 1. REGISTERS

Header sender control and monitor	A0	R/W	Priority			DSAE	XRS	RTE	SCE	HSA
	A1	R	SNT	CNC	LAB	JAM	DFR	Priority/Status		
		W	Reserved				Preset retry cntr			
	A2*	R/W	S-FIELD							LSB
			Not Used							MSB
	A3*	R/W	R-FIELD							LSB
			Not Used							MSB
Module addressing	A4	R/W	Reserved	Reserved	HR5	HR4	HR3	HR2	HR1	HR0
					Header receiver enable					
	A5*	W	HR0 address							LSB
			Not Used							MSB
		W	HR1–5 addresses							
	A6	R/W	Header receiver mux							
Module control	A7	R/W	Write Data	DRE	DRM	DRV	DRA	DR frame type		
				Write select						
	A8	R/W	Write Data	DSE	DSM	DST	Write Enable	DS frame type		
				Write select						
	A9	R/W	Write Data	Reserved			FF3	FF2	FF1	FF0
							Write select			
	A10*	R/W	FF1 Connect	FF1 mode			FF0 Connect	FF0 mode		
			FF3 Connect	FF3 mode			FF2 Connect	FF2 mode		
Interrupt control	A11	R/W	Interrupt vector (lower 3 bits read only)							
	A12	R/W	INP	HS	DR	DS	FF3	FF2	FF1	FF0
				Pending interrupts (read), reset mask (write)						
	A13	R/W	INE	HS	DR	DS	FF3	FF2	FF1	FF0
				Interrupt enable						
Buffers	A14*	R/W	Data sender buffer							
	A15*	R	Data receiver buffer							

*Indicates sequentially-accessed register block.

sends 111 in the frame type field, cancelling the frame.

If DSE in A8 is one, the data sender sends one of the values 001, 010, or 011 in the frame type field, as controlled by the low order bits in A8. These values indicate 1, 2, or 4 bytes of data, respectively, following the frame type field. (If one of the illegal values 000, 100 – 111, is inadvertently programmed into the three low-order bits of A8, the SCC68173 sends 111 in the frame type, cancelling the frame.)

While sending the frame type, the data sender uses VMSbus arbitration as described previously in VMSbus Overview. If it loses the arbitration, it stops sending and captures the winning frame type value. If the value is 111, another module (typically the data receiver) is not ready for the frame and has cancelled it. In this case the data sender ignores the frame, keeping DSE set.

If the data sender loses the arbitration to a frame type value 010 – 110, then another data sender is also selected by the S address, and it is set up to send more data than is this data sender. In this case, this data sender waits out the number of bits indicated by the winning frame type value, and then sends the error code 110 in the frame status field.

If the data sender wins the VMSbus arbitration in the frame type field, it goes on to send 1, 2, or 4 bytes of data from register block A14. If the DSAE bit in the initiating header subframe was 1, it uses VMSbus arbitration while sending the data, and if it loses the arbitration it ignores the frame, leaving DSE set. If DSAE was 0, it does not use VMSbus arbitration, but simply places each data bit on the VMSbus.

After the data, the data sender sends the code 010 in the frame status. This OR's with the code 001 from the data receiver to make the "success" code 011.

The data sender also samples the frame status and the following jam bit. If the status code is 011 and the jam bit is 0, then it sets the DST bit in register A8. Also, if the DSM bit in A8 is zero, it clears the DSE bit in A8 and, if the INE and DS bits in A13 are both one, it requests interrupt on IRQN. If DSM is one,

the SCC68173 leaves DSE set and does not request interrupt (regardless of INE and DS in A13).

The DSM bit in A8 thus controls the gross operational state of the data sender. When DSM is zero, setting DSE enables the module to send data once, then clear DSE and cancel any further frames that select it, until system software (loads new data and) sets DSE again. When DSM is one, setting DSE enables the data sender to send the data in register block A14 as many times as it is selected by VMSbus frames, until system software clears DSE.

Software cannot access register block A14 while DSE is set. This restriction applies to both reading and writing, and can be handled in one of three ways:

1. Wait until DSE is cleared by the SCC68173. This is normal operation with DSM zero, but is not applicable if DSM is one.
2. Rewrite A8 to make DSE zero. Software should then check A8 until DSE reads back as zero, because if the data sender is selected by the current VMSbus frame, the clearing of DSE does not become effective until the end of the frame.
3. Simply go ahead and access register block A14. This automatically clears DSE, but the entire access operation is delayed if the data sender is selected by the current VMSbus frame, until the end of the frame. Systems which cannot tolerate a DTACKN delay of up to 270 bit times should use method 2 above. (Since SCC68173 data senders are restricted to 1 – 4 byte data, the practical maximum delay is about 40 bit times except in the case where there is a data size conflict between a SCC68173 and some other kind of VMSbus data sender. 40 bit times is about 14μsec at a 2.9MHz data rate.)

DATA RECEIVER

A data receiver captures data from VMSbus frames in which it is selected by the R address. System software should set up a data receiver as follows:

1. If necessary, establish the DR's (R) address and the link to a header receiver, via A5 and A6, and then enable the header receiver in A4.
2. If necessary, enable the DR interrupt in register A13.
3. Write register A7 with DRE one, and optionally, DRM one.
4. For a read frame, program the header sender to initiate a frame with the data receiver's address in the R field.

Because the DRE and DRA bits in A7 can be set and cleared by system software and also cleared by a VMSbus frame, a special technique must be used by system software in writing A7. For bits DRE, DRM, DRV, and DRA, a '1' in that bit position indicates that the bit should be written with the data contained in the high-order bit (labeled Write Data, in table 1).

Table 2. HEADER RECEIVER MULTIPLEXING IN REGISTER A6

A6 BITS	HR	VALUE	MODULE SELECTED	BY MATCH IN
7 (MSB)	3	0	Bit module FF3	S, R
		1	Data receiver	R
6	5	0	Data receiver	R
		1	Bit module	S, R
5, 4	2	00	Bit module FF2	S, R
		01	Bit module FF1	S, R
		10	Data receiver	R
		11	Bit module FF2	S, R
3, 2	1	00	Bit module FF1	S, R
		01	Data sender	S
		10	Bit module FF2	S, R
		11	Bit module FF1	S, R
1	4	0	Data sender	S
		1	Bit module FF1	S, R
0 (LSB)	0	0	Bit module FF0	S, R
		1	Data sender	S

When the data receiver is selected by the R address in a header subframe, it clears the DRA bit in A7. If bit DRE in A7 is zero, it sends 111 in the frame type field that follows the header subframe, thereby cancelling the frame.

If DRE in A7 is one, the data receiver samples the frame type field and saves the result in the low-order 3 bits of A7. If the frame type is 111, the frame has been cancelled by another VMSbus module (typically a data sender). In this case the data receiver ignores the frame.

If the frame type field is 000, the data receiver then sends the error code 101 in the immediately following frame status field.

If the frame type field is 100, 101, or 110, the data receiver waits for 64, 128, or 256 bits of data (respectively) to go by, and then sends the error code 101 in the frame status field.

If the frame type field is 001, 010, or 011, the data receiver captures the following 1, 2, or 4 data bytes (respectively), and saves them in register block A15. It then sends the frame status code 001, which OR's with the code 010 from the data sender to produce the "successful" status code 011.

The data receiver also samples the frame status and the following jam bit. If the status code is 011 and the jam bit is 0, then it sets the DRV bit in A7. Also, if the DRM bit in A7 is zero, the data receiver clears the DRE bit in A7 and, if the INE and DR bits in A13 are both one, it requests interrupt on the IRQN pin. If DRM is one, it leaves DRE set and does not request interrupt, regardless of INE and DR in A13.

Thus the DRM bit governs the gross operational mode of the data receiver. If DRM is zero, once the data receiver has successfully received data, it clears DRE and cancels any further frames addressed to it, until system software (reads the data from A15 and) sets DRE again. When DRM is one, the data receiver will accept data into A15 from any frame addressed to it.

When DRM is one, system software must guard against the possibility that a new VMSbus frame could be received while it is reading data out of register block A15, which would result in reading out a mixture of old and new data. It can do this in one of two ways:

1. Rewrite A7 with DRE zero. Software should then check A7 until DRE reads back as zero, because if the data receiver is selected by the current VMSbus frame, the clearing of DRE does not become effective until the end of the frame. When DRE is zero, the software can safely read out register block A15. Typically, it then sets DRE again. This method has the disadvantage that it results in cancellation of any frame which selects the data receiver while its data is being read out.
2. Set the DRA bit in register A7, read out register block A15, then read A7. If the DRA bit in A7 is still set, the data from A15 and the frame type code in A7 are valid. If DRA has been cleared (by the SCC68173 because it started receiving a frame), then the software should set DRA and try again.

FLAGS

Flags essentially implement a flip-flop or latch which is set when the module is selected by

Table 3. OPERATING MODES FOR FLAGS IN A10

VALUE	MODULE TYPE	INTERRUPT ON
000	Disabled (Cancellation)	
001	Simple flag	Toggle
010	Simple flag	Set
011	Simple flag	Clear
100	Semaphore	Set
101	Semaphore	Clear
110	Token	Set
111	Token	Clear

the S address in a frame, and reset when the module is selected by the R address in a frame. Four flags are provided in the SCC68173, called FF1 through FF4. These modules can also be directly controlled by system software, and can be programmed to control two discrete SCC68173 output pins, OUT1 and OUT2.

Register A9 provides direct sensing and control of the state of the flags. Since the state of the modules can be changed both by system software and by VMSbus frames, a special technique must be used in writing A9. A '1' in any or all of the four least significant bit positions indicates that a flag should be written with the data contained in the high-order bit (labeled Write Data in table 1).

Register block A10 provides control of the operation of each flag. In A10, there are four bits for each module, of which the high-order bit determines whether the module is connected to the OUT1 or OUT2 pin, and the three less significant bits determine the module's operational mode, as shown in table 3.

If a flag is programmed with the 000 code and is also connected to a header receiver (in A6) that is enabled (in A4), then any frame that selects the flag (in either the S or R fields) will be cancelled.

The simple flag, semaphore, and token categories also refer to circumstances where flags cancel frames that are addressed to them.

A simple flag does not cancel frames based on its state. That is, it accepts "set frames" even if it is already set, and accepts "reset frames" if it is already reset. On the other hand, if the module's interrupt was enabled (in A13) at the time of a previous frame, and the interrupt is still pending, then any frame that selects the flag (in either the S or R fields) will be cancelled.

A semaphore module cancels a set frame if it is already set, but accepts a reset frame if it is already reset. It does not cancel frames based on its pending interrupt status.

A token module cancels a set frame if it is already set, and cancels a reset frame if it is already reset. It does not cancel frames based on its pending interrupt status.

Within each category there is a choice as to when the module requests interrupt. This choice is significant only if the module's interrupt enable bit is set in A13. System software should set up a flag as follows:

1. If necessary, disable the associated header receiver(s) in A4.
2. If necessary, program the initial state of the module, in A9.
3. Program the module's operating mode, in A10.
4. If necessary, clear any pending interrupt for the module, in A12.
5. If necessary, enable interrupt for the module, in A13.
6. Program the module's address via A5.
7. Connect a header receiver to the module, in A6.
8. Enable the header receiver in A4.

The OUT1 and OUT2 pins are high-active, and go low on reset. Each pin can reflect the state of 1 or 2 flags:

OUT1 = (FF0 and FF0 connect) or (FF1 and FF1 connect)

OUT2 = (FF2 and FF2 connect) or (FF3 and FF3 connect)

INTERRUPTS

As described in the previous sections, the SCC68173 can be set up to generate an interrupt request based on any of seven events:

1. When the header sender finishes sending a header subframe, and clears HSA.

2. When the data sender sends data in a frame, and clears DSE.

3. When the data receiver receives data in a frame, and clears DRE.

4 – 7. When flag FF0 – FF3 is set and/or reset by a frame.

Each of these seven potential interrupt sources is conditioned on a corresponding bit in register A13. When an A13 bit is one, the corresponding module's interrupt is enabled.

The most significant bit in A13, INE, controls the overall interrupt enable of the SCC68173.

Register A12 contains a corresponding set of interrupt pending bits. An interrupt pending bit is set if the appropriate VMSbus event occurs, and the module's interrupt enable bit is one in A13. The most significant bit of A12 is the OR of the other seven bits.

A pending bit in A12 can be directly reset by writing a 1 to it. Disabling an interrupt channel in A13 automatically resets the corresponding pending bit. When an interrupt service routine processes a SCC68173 interrupt, it should clear one or more of the pending bits and/or enable bits and/or make INE zero before dismissing the interrupt, to prevent a second interrupt resulting from the same event.

When the SCC68173 detects an interrupt acknowledge cycle (IACKN low) and it is requesting interrupt on IRQN (or is about to do so), it responds by placing an interrupt vector on the D7 – D0 lines and asserting DTACKN. Typically, this vector has its high-order 5 bits taken from register A11, and its low-order 3 bits in accordance with the highest-priority pending interrupt:

110	Header sender (highest priority)
101	Data receiver
100	Data sender
011	Flag FF3
010	Flag FF2
001	Flag FF1
000	Flag FF0 (lowest priority)

The exception to this vector structure is where case A11 has not been programmed since the SCC68173 was reset. In this case, the vector hexadecimal 0F is returned regardless of which modules have interrupt pending.

Jam Sequence

As described in VMSbus Overview, the SCC68173 monitors the VMSbus at all times (other than when RESETN is low), tracking the entire length of each frame and checking for start bits within each frame. If such a start bit is detected, it means that this SCC68173 is out of frame synchronization with one or more other VMSbus devices. In this case, this SCC68173 must send a string of 512 one bits to resynchronize the VMSbus, starting with the bit after the misplaced start bit. The timing of the SCC68173/SCB68171 interface does not allow this to be done in the normal (synchronous finite-state machine) way. Instead, if RSTARTN goes low from the SCB68171 during a frame, this signal makes the SCC68173 produce XSTARTN and XDATAN low combinatorially. This combination makes the SCB68171 asynchronously drive SERDATN low for a one-bit. By the next bit time, the SCC68173 is set up to send the rest of the jam sequence in the normal way. A jam

sequence maintains all SCC68173 hardware as it was at the start of the frame.

Reset

A low on the RESETN input clears the following bits to zero:

HSA and SCE in register A0

DRE and DRV in A7

DSE and DST in A8

All bits in registers A1, A4, A6, A12, A13

It also sets register A11, the interrupt vector, to the value hexadecimal 0F.

Signetics

SCB68175
Bus Controller

Preliminary Specification

Microprocessor Products

DESCRIPTION

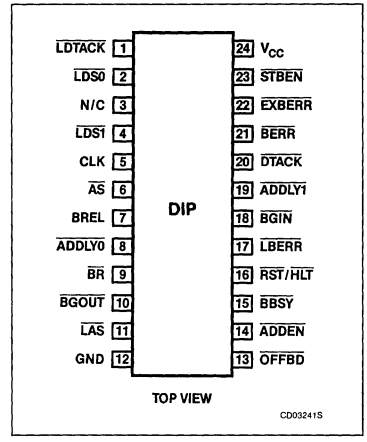
The Signetics SCB68175/8X821 Bus Controller is a high-speed bus requester and timing generator designed to interface a local master (typically, a microprocessor or DMA controller) to the VMEbus or VERSAbus™. The requester is fully asynchronous and controls all the sequencing required for acquisition and use of the bus, including both strobe and buffer timing. To permit fixed or variable length bus access, both release-when-done (RWD) and release-on-request (ROR) modes are supported by the SCB68175/8X821. In addition, an early bus-busy release increases system throughput by allowing bus arbitration to be performed concurrently with the final bus cycle of the local master. A bus error/retry circuit is also included in the SCB68175/8X821. This circuit allows the local master to rerun a cycle that was terminated by a bus error. Interface and control signals for a typical configuration are shown in Figure 1.

The SCB68175/8X821 Bus Controller is designed primarily for implementing a simple and efficient interface to the VMEbus. For more information regarding the protocol definitions, proper use, and application of this device, refer to the VMEbus Specification Manual.

FEATURES

- **Asynchronous bus controller for VMEbus and VERSAbus systems**
- **Provides transparent system interface to local master**
- **Supports both release-on-request (ROR) and release-when-done (RWD) operations**
- **Early bus-busy release for optimal bus usage**
- **Error/entry sequencing**
- **Dual port capability using 74LS764 DRAM controller**
- **High speed bipolar technology**
- **Single +5V supply**
- **24-pin slimline DIP package**

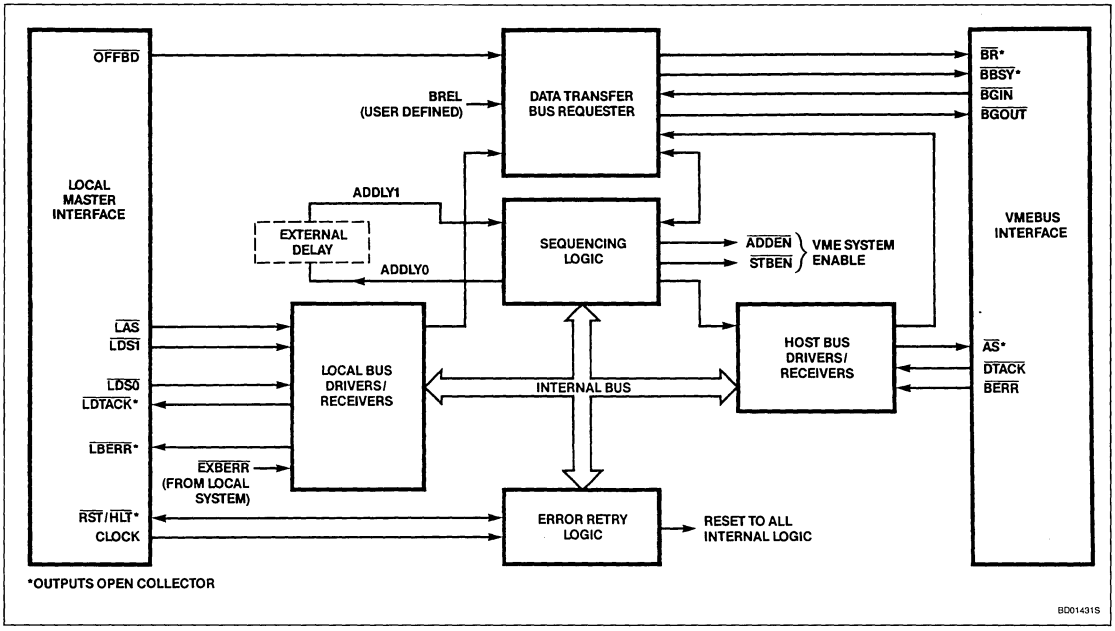
PIN CONFIGURATION



VERSAbus is a registered trademark of Motorola, Inc.

Bus Controller SCB68175

BLOCK DIAGRAM



Bus Controller | SCB68175

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C
Ceramic DIP	SCB68175C2I24
Plastic DIP	SCB68175C2N24

PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
$\overline{LDTACK}$	1	O	**Local Data Transfer Acknowledge:** Active low, open collector. $\overline{LDTACK}$ is the acknowledge signal to the local master.
$\overline{LDS0}$	2	I	**Local Data Strobe 0:** Active low (hysteresis) input lower data strobe of the local master.
$\overline{LDS1}$	4	I	**Local Data Strobe 1:** Active low (hysteresis) input upper data strobe of the local master.
CLK	5	I	**Clock:** CLK is the clock input to the retry circuitry.
$\overline{AS}$	6	I/O	**Address Strobe:** Active low, open collector. $\overline{AS}$ is the address strobe of the system bus. As an output, $\overline{AS}$ is asserted to indicate a valid address on the system bus. As an input (hysteresis), $\overline{AS}$ is monitored during the bus acquisition phase to determine when the current master has relinquished control of the bus.
BREL	7	I	**Bus Release:** Active high input used to get the SCB68175/8X821 to release the system bus.
$\overline{ADDLYO}$	8	O	**Address Delay Out:** Active low, totem pole. $\overline{ADDLYO}$ is the output to an external delay line.
$\overline{BR}$	9	O	**Bus Request:** Active low, open collector. The bus request output to the system bus; asserted when the SCB68175/8X821 is not the current master and $\overline{LAS}$ and $\overline{OFFBD}$ are asserted.
$\overline{BGOUT}$	10	O	**Bus Grant Out:** Active low, totem pole. $\overline{BGOUT}$ is the daisy-chain output.
$\overline{LAS}$	11	I	**Local Address Strobe:** Active low $\overline{LAS}$ is the (hysteresis) input address strobe signal from the local master. If $\overline{OFFBD}$ and $\overline{LAS}$ are asserted low, the SCB68175/8X821 will initiate a bus cycle requesting the bus, if necessary.
GND	12	I	**Ground**
$\overline{OFFBD}$	13	I	**Off Board:** $\overline{OFFBD}$ is an active low select input driven by an address decoder to request access to the system bus. If $\overline{LAS}$ and $\overline{OFFBD}$ are asserted low, the SCB68175/8X821 will initiate a bus cycle requesting the bus, if necessary.
$\overline{ADDEN}$	14	O	**Address Enable:** Active low, totem pole. $\overline{ADDEN}$ is an enable signal to the address and data bus drivers/receivers.
$\overline{BBSY}$	15	O	**Bus Busy:** Active low, open collector. $\overline{BBSY}$ output is driven low when the local master has been granted the system bus.
$\overline{RST}/\overline{HLT}$	16	I/O	**Reset/Halt:** Active low, open collector. (Hysteresis) input resets the SCB68175/8X821. As an output, it is asserted along with $\overline{LBERR}$ to initiate a rerun cycle upon receiving either $\overline{BERR}$ for a bus cycle, or $\overline{EXBERR}$ for a local cycle.
$\overline{LBERR}$	17	I/O	**Local Bus Error:** Active low, open collector. $\overline{LBERR}$ is the bus error signal to the local master.
$\overline{BGIN}$	18	I	**Bus Grant In:** Active low. $\overline{BGIN}$ is the daisy-chain input.
$\overline{ADDLYI}$	19	I	**Address Delay In:** Active low. Input from the external delay line.
$\overline{DTACK}$	20	I	**Data Transfer Acknowledge:** Active low. $\overline{DTACK}$ is the (hysteresis) input data transfer acknowledge signal of the system bus. During the bus acquisition phase, $\overline{DTACK}$ is monitored to determine when the current master has relinquished control of the bus.
$\overline{BERR}$	21	I	**Bus Error:** Active low. $\overline{BERR}$ is the (hysteresis) input bus error signal of the system bus. During the bus acquisition phase, $\overline{BERR}$ is monitored to determine when the current master has relinquished control of the bus.
$\overline{EXBERR}$	22	I	**External Bus Error:** Active low. $\overline{EXBERR}$ is a (hysteresis) input bus error signal from a local device.
$\overline{STBEN}$	23	O	**Strobe Enable:** Active low, totem pole. $\overline{STBEN}$ is an enable signal to the system bus data strobe drivers.
V_{CC}	24	I	**Supply Voltage:** +5V power supply.

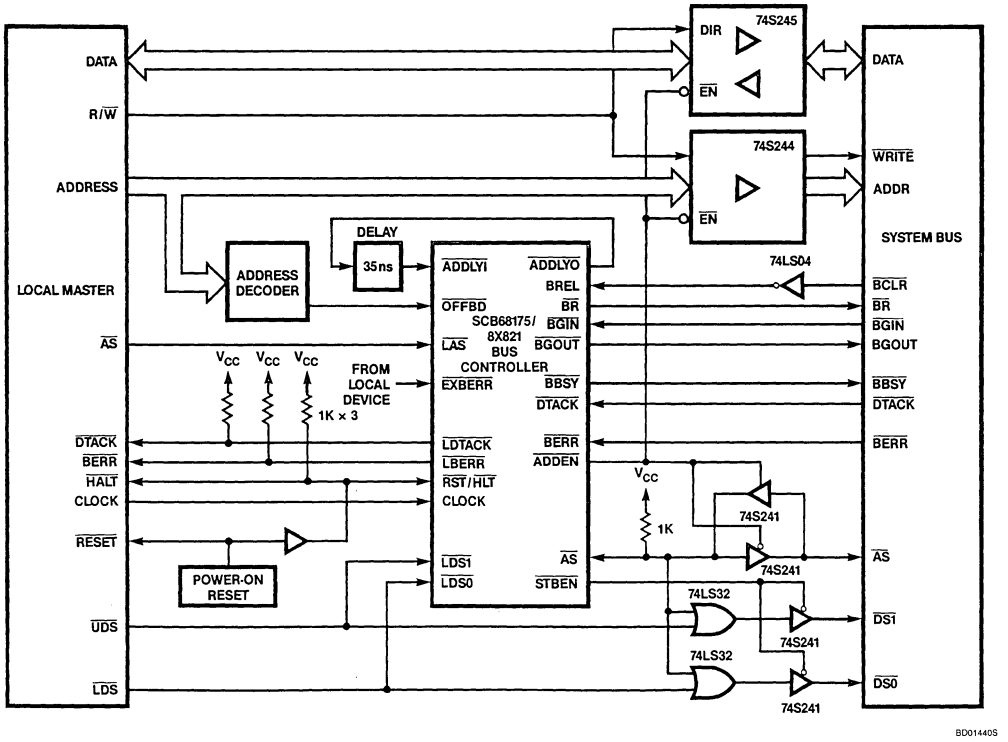


Figure 1. Typical System Configuration

FUNCTIONAL DESCRIPTION

Bus Acquisition

In a multimaster, bus oriented system, each master requires some method of gaining access to global resources such as, main memory, disk storage and I/O devices. A very precise access arbitration protocol is implemented (fully defined in the VMEbus Specification) and is basically:

1. Asserting a bus request.
2. Receiving a grant that the bus is available.
3. Acknowledging that mastership has been assumed.

Requesting the Bus

Local devices capable of becoming bus masters, for example, CPU and interrupt handler, request the system bus by asserting $\overline{OFFBD}$. This signal is typically the output from an address decoder indicating that the required resource (memory, I/O) is accessible only via the host bus. With both $\overline{OFFBD}$ and $\overline{LAS}$ low, the SCB68175/8X821 asserts the system bus request signal ($\overline{BR}$). In a VMEbus system, $\overline{BR}$ is tied to one of the four bus request signals ($\overline{BR0}$ – $\overline{BR3}$).

Receiving the Bus Grant

The BGIN/BGOUT signals comprise a daisy-chain network allowing multiple bus masters to share the same bus request level. The SCB68175/8X821 will assert $\overline{BGOUT}$ only if $\overline{BGIN}$ is received low prior to the assertion of a bus request ($\overline{BR}$). This allows the next master, in the daisy-chain, access to the bus. If $\overline{BR}$ precedes $\overline{BGIN}$, then the SCB68175/8X821 will assert $\overline{BBSY}$ and release $\overline{BR}$. This completes the bus request phase.

If $\overline{BGIN}$ precedes $\overline{BR}$, then $\overline{BR}$ remains asserted pending the next arbitration cycle. If a bus request and $\overline{BGIN}$ are asserted simultaneously, an onboard arbiter guarantees the outputs to remain stable until a random choice is made of either $\overline{BR}$ or $\overline{BGIN}$.

Bus Access

Actual bus access does not occur until the previous master has relinquished control of the bus. This is detected when the system signals $\overline{AS}$, $\overline{DTACK}$ and $\overline{BERR}$ are received high. The SCB68175/8X821 then controls the sequencing of the address, data and strobe signals required for each bus cycle.

Typical Bus Cycle

See Figure 2 for a typical bus access flowchart. For each bus access, the SCB68175/8X821 drives $\overline{ADDEN}$, $\overline{STBEN}$ and $\overline{AS}$ low. An external delay line (35ns minimum), when connected between $\overline{ADDLY0}$ and $\overline{ADDLY1}$, guarantees a minimum set-up time between the assertion of $\overline{ADDEN}$ and $\overline{AS}$.

In most cases, the local master will require multiple accesses to the host bus. To accommodate, the SCB68175/8X821 cycles $\overline{ADDEN}$, $\overline{STBEN}$ and $\overline{AS}$ for each bus access. For a local cycle (when the master asserts $\overline{LAS}$, but $\overline{OFFBD}$ is high), the SCB68175/8X821 holds $\overline{ADDEN}$, $\overline{STBEN}$ and $\overline{AS}$ high, and allows the master access to its local bus.

Read-Modify-Write Cycle

The SCB68175/8X821 also allows a read-modify-write access for system resources. A read-modify-write cycle is similar to a read cycle followed by a write cycle, except that $\overline{AS}$ is continuously driven low during both transfer cycles. However, unlike a read followed by a write, the read-modify-write cycle cannot be interrupted because $\overline{AS}$ is driven low continuously through both cycles, and

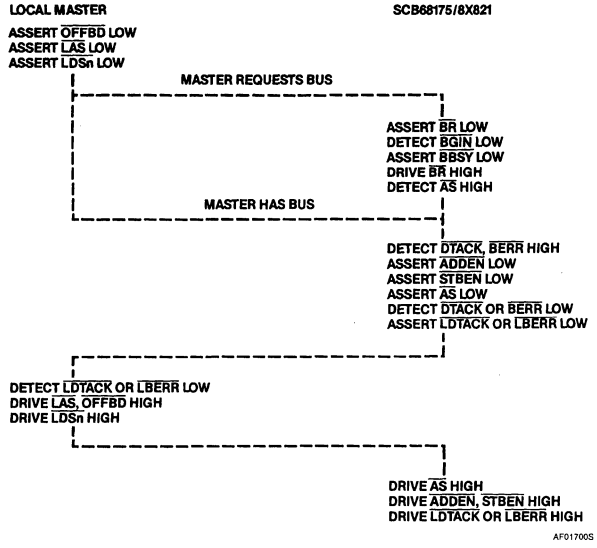


Figure 2. Typical Bus Cycle

control of the host bus may only be transferred while $\overline{AS}$ is high. The SCB68175/8X821 will cycle $\overline{STBEN}$ as required for a read-modify-write cycle.

Bus Release

Following initial bus acquisition, $\overline{BBSY}$ remains low for a minimum of 90ns, providing a 35ns delay line is connected between $\overline{ADDLY0}$ and $\overline{ADDLY1}$. If BREL is asserted high any time before the end of the current bus cycle, the SCB68175/8X821 will release $\overline{BBSY}$ allowing bus arbitration to take place. At the conclusion of the current bus cycle, the SCB68175/8X821 will release the bus by negating $\overline{ADDEN}$, $\overline{AS}$ and $\overline{STBEN}$. A subsequent offboard request will result in the start of a new bus acquisition cycle.

The bus release signal can be used to control the duration, that is the number of cycles the master will control the bus. For most simple masters, BREL can be tied high resulting in a single-cycle RWD configuration. In this case, the SCB68175/8X821 must request the bus for each and every bus cycle. A DMA controller might use the RWD mode, but would assert BREL only after performing a fixed number of cycles.

An ROR mode can be achieved by asserting BREL only when some other master is requesting bus access. Typically, this may be achieved by NANDing the appropriate $\overline{BR}$ lines together to assert BREL. BREL is not monitored by the SCB68175/8X821 until the last leading edge of either $\overline{BGIN}$ or $\overline{ADDLY1}$. This prevents the SCB68175/8X821 from relinquishing control due to its own bus request.

For improved system performance, BREL should be asserted early in the current cycle (that is, as soon as possible after $\overline{LAS}$ is asserted), to allow bus arbitration to be performed concurrently with the existing bus cycle.

Bus Error/Entry

The SCB68175/8X821 provides a bus error/entry facility which allows the local master to rerun its last cycle once, if the cycle was terminated by a bus error.

The bus error/entry sequence is initiated when a bus error is received instead of a (normal) data transfer acknowledge signal. Upon receipt of a bus error (either $\overline{BERR}$ for a bus cycle, or $\overline{EXBERR}$ for a local cycle), the SCB68175/8X821 asserts both $\overline{LBERR}$ AND $\overline{RST/HLT}$. This will allow the local master to rerun the cycle.

The local master can then rerun the bus cycle using the same address, and the same data for a write. If this cycle is also terminated by a bus error, the SCB68175/8X821 will assert $\overline{LBERR}$ but not $\overline{RST/HLT}$.

The $\overline{EXBERR}$ input is provided for local resources to initiate a retry sequence. Local devices may drive $\overline{LBERR}$ (an open collector output) directly to bypass the retry function.

Reset

The SCB68175/8X821 must be reset as specified in the reset timing in order to ensure proper operation. When $\overline{RST/HLT}$ is asserted, the SCB68175/8X821 will drive all outputs high to release the bus. If the SCB68175/8X821 has control of the system bus and the local master fails, $\overline{RST/HLT}$ should be asserted. This will ensure that the system bus is relinquished by the SCB68175/8X821 Bus Controller.

2

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
Supply voltage	−0.5 to +7.0	V
Input voltage	−0.5 to +5.5	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to+70°C, $V_{CC} = 5.0V \pm 5\%$[4,5]

PARAMETER		TEST CONDITIONS	LIMITS Min	LIMITS Max	UNIT
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
V_{IC}	Input clamp diode voltage	$V_{CC} = 4.75V$, $I_{IN} = -18mA$		−1.5	V
I_{IL}	Input low current	$V_{CC} = 5.25V$, $V_{IN} = 0.4V$		−410	μA
I_{IH}	Input high current	$V_{CC} = 5.25V$, $V_{IN} = 2.7V$		20	μA
I_I	Input high current	$V_{CC} = 5.25V$, $V_{IN} = 5.5V$		100	μA
V_{OL}	Output low voltage	$V_{CC} = 4.75V$			
	$\overline{LDTACK}$, $\overline{ADDLYO}$, $\overline{BGOUT}$, $\overline{ADDEN}$, $\overline{RST}/\overline{HLT}$, $\overline{LBERR}$, $\overline{DTACK}$	$I_{OL} = 8mA$		0.5	V
	$\overline{AS}$	$I_{OL} = 20mA$		0.5	V
	$\overline{BR}$, $\overline{BBSY}$	$I_{OL} = 64mA$		0.5	V
V_{OH}	Output high voltage				
	$\overline{ADDLYO}$, $\overline{BGOUT}$, $\overline{ADDEN}$, $\overline{STBEN}$	$V_{CC} = 4.75V$, $I_{OH} = -400\mu A$	2.5		V
I_{CEX}	Open collector leakage current				
	$\overline{LDTACK}$, $\overline{AS}$, $\overline{RST}/\overline{HLT}$, $\overline{LBERR}$	$V_{CC} = 4.75V$, $V_{OUT} = 5.25V$		100	μA
I_{OS}	Short circuit output current				
	$\overline{ADDLYO}$, $\overline{BGOUT}$, $\overline{ADDEN}$, $\overline{STBEN}$	$V_{CC} = 5.25V$, $V_{OUT} = 0V$[6]	−15	−100	mA
I_{CC}	Supply current	$V_{CC} = 5.25V$		140	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on + 150°C maximum junction temperature and thermal resistance of 60°C/W junction to ambient for ceramic package (116°C/W for plastic package).
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all inputs swing between 0.4V and 2.4V with a transition time of 3ns maximum and output voltages are checked at 0.8V and 2.0V.
6. At any time, no more than one output should be connected to ground.
7. If the asynchronous input requirement (t_{BG}) is not satisfied, then $\overline{BGOUT}$ will be asserted low for the duration of $\overline{BGIN}$ low.
8. Requires minumum 35ns delay line connected between $\overline{ADDLYO}$ and $\overline{ADDLYI}$.
9. If the asynchronous input requirement (t_{BRLS}) is not satisfied, then $\overline{BBSY}$ will be negated (driven high) on the following assertion of $\overline{LAS}$.
10. Valid only for system bus cycles not involving bus acquisition.
11. BREL is a don't care until the leading edge of either $\overline{ADDLYI}$ or $\overline{BGIN}$, whichever occurs last. If $\overline{ADDLYI}$ occurs last, then t_{DIS} is valid. If $\overline{BGIN}$ occurs last, then t_{BGS} is valid. If they occur simultaneously, then t_{DIS} is valid.
12. Valid only with 35ns external delay line and if t_{BBY} (minimum) has been met.
13. Valid from last leading edge of either $\overline{DTACK}/\overline{BERR}$ or $\overline{AS}$.
14. Only one of t_{REQ1}, t_{REQ2} is valid at once. If the falling edge of $\overline{OFFBD}$ occurs after the falling edge of $\overline{LAS}$, then t_{REQ1} is valid. If the falling edge of $\overline{LAS}$ occurs after the falling edge of $\overline{OFFBD}$, then t_{REQ2} is valid. If both $\overline{LAS}$ and $\overline{OFFBD}$ occur simultaneously, then t_{REQ2} is valid.
15. Only one of t_{ADN1}, t_{ADN2} is valid at once. If the falling edge of $\overline{OFFBD}$ occurs after the falling edge of $\overline{LAS}$, then t_{ADN1} is valid. If the falling edge of $\overline{LAS}$ occurs after the falling edge of $\overline{OFFBD}$, then t_{ADN2} is valid. If both $\overline{LAS}$ and $\overline{OFFBD}$ occur simultaneously, then t_{ADN2} is valid.
16. The SCB68175/8X821 must see two falling edges of the clock after the rising edge of $\overline{BERR}/\overline{EXBERR}$ before t_{GO} is valid.
17. Valid only if no bus request is pending.
18. The SCB68175/8X821 will only assert $\overline{LDTACK}/\overline{LBERR}$ with the high-to-low transition of $\overline{DTACK}/\overline{BERR}$. When running consecutive bus cycles, the SCB68175/8X821 will then ensure that any slow or lazy $\overline{DTACK}/\overline{BERR}$ terminating the first cycle is not transmitted through to the local master interface.
19. These parameters are guaranteed at the values listed; these values were determined either by system bench testing or by Signetics' characterization procedures. All other tabular entries are taken directly from simulation results run at a range of operational frequencies; these values are not tested or guaranteed.

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5]

PARAMETER		LIMITS		UNIT
		Min	Max	
Read/write (see figure 3)				
t_{REQ1}[14]	$\overline{OFFBD}$ low to $\overline{BR}$ low		56	ns
t_{REQ2}[14]	$\overline{LAS}$ low to $\overline{BR}$ low		56	ns
t_{BG}[7]	$\overline{BR}$ low to $\overline{BGIN}$ low set-up	0		ns
t_{BGAK}	$\overline{BGIN}$ low to $\overline{BBSY}$ low		32	ns
t_{BBY}[8]	$\overline{BBSY}$ asserted width	90		ns
t_{ASEN}[13]	$\overline{AS}$, $\overline{DTACK}$, $\overline{BERR}$ high to $\overline{ADDEN}$ low		39	ns
t_{DIST}	$\overline{ADDLYI}$ low to $\overline{STBEN}$ Low		23	ns
t_{STAS}	$\overline{STBEN}$ low to $\overline{AS}$ low		14	ns
t_{BRLS}[9]	BREL high to $\overline{LAS}$ high set-up time	0		ns
t_{ADN1}[10, 15, 19]	$\overline{OFFBD}$ low to $\overline{ADDEN}$ low		37	ns
t_{ADN2}[10, 15]	$\overline{LAS}$ low to $\overline{ADDEN}$ low		35	ns
t_{OBDH}	$\overline{LAS}$ high to $\overline{OFFBD}$ high hold time	0		ns
t_{BLBY}[12]	BREL high to $\overline{BBSY}$ high		65	ns
t_{DYEN}	$\overline{ADDLYO}$ low to $\overline{ADDEN}$ low		10	ns
t_{DIDO}	$\overline{ADDLYI}$ to $\overline{ADDLYO}$		29	ns
t_{DLY}	$\overline{ADDLYO}$ high/low to $\overline{ADDLYI}$ high/low (minimum delay line)	35		ns
t_{LAS}[19]	$\overline{LAS}$ high to $\overline{AS}$ high		11	ns
t_{LSEN}	$\overline{LAS}$ high to $\overline{ADDEN}$ high		25	ns
t_{LSST}	$\overline{LAS}$ high to $\overline{STBEN}$ high		25	ns
t_{BGS}[11]	BREL high to $\overline{BGIN}$ high set-up	0		ns
t_{DIS}[11]	BREL high to $\overline{ADDLYI}$ high set-up	0		ns
t_{DIAS}	$\overline{ADDLYI}$ low to $\overline{AS}$ low		37	ns
t_{ASST}[19]	$\overline{AS}$ high to $\overline{STBEN}$ high		14	ns
t_{ASD}[19]	$\overline{AS}$ high to $\overline{ADDEN}$ high		14	ns
t_{ASDT}	$\overline{AS}$ high to $\overline{DTACK}$ high	0		ns
t_{CYED}	$\overline{LAS}$ high to $\overline{LDTACK}$/$\overline{LBERR}$ high		45	ns
t_{LSLD}	$\overline{LAS}$ high to $\overline{LDSn}$ high	0		ns
t_{BGL}[17]	$\overline{BGIN}$ low to $\overline{BGOUT}$ low		57	ns
t_{BGH}[17]	$\overline{BGIN}$ high to $\overline{BGOUT}$ high		57	ns
t_{LDS}	$\overline{LAS}$ low to $\overline{LDSn}$ low	0		ns
t_{LDSO}[10]	$\overline{LAS}$ low to $\overline{ADDLYO}$ low		26	ns
t_{OFDO}[10]	$\overline{OFFBD}$ low to $\overline{ADDLYO}$ low		26	ns
t_{ASDO}[13]	$\overline{AS}$, $\overline{DTACK}$, $\overline{BERR}$ high to $\overline{ADDLYO}$ low		38	ns

AC ELECTRICAL CHARACTERISTICS (Continued)

PARAMETER		LIMITS		UNIT
		Min	Max	
Read-modify-write (see figure 4)				
t_{PDL}[18]	$\overline{DTACK}/\overline{BERR}$ low time, $\overline{LDTACK}/\overline{LBERR}$ low propagation delay		28	ns
t_{CYEN}	Local data strobe 0 and 1 high to $\overline{LDTACK}/\overline{LBERR}$ high		45	ns
t_{LDAS}	$\overline{LDTACK}/\overline{LBERR}$ low to $\overline{LAS}$ high	0		ns
t_{LDST}	Local data strobe 0 and 1 high to $\overline{STBEN}$ high		32	ns
t_{DTST}	$\overline{DTACK}/\overline{BERR}$ high to $\overline{STBEN}$		38	ns
t_{LDLN}	$\overline{LDTACK}/\overline{LBERR}$ low to local data strobe 0 or 1 high	0		ns
Bus error/retry and reset (see figures 5, 6 and 7)				
t_{RST}	$\overline{RST}$ low time		$3t_{CKPD}$	ns
t_{CKH}	Clock high time	20		ns
t_{CKPD}	Clock period	40		ns
t_{PDL2}	$\overline{BERR}/\overline{EXBERR}$ low to $\overline{LBERR}$ low propagation delay		28	ns
t_{PDH2}	$\overline{BERR}/\overline{EXBERR}$ high/to $\overline{LBERR}$ propagation delay		12	ns
t_{ERR}	$\overline{LBERR}$ low to $\overline{RST}/\overline{HLT}$ low		16	ns
t_{GO}[16]	Clock to $\overline{RST}/\overline{HLT}$ high		38	ns

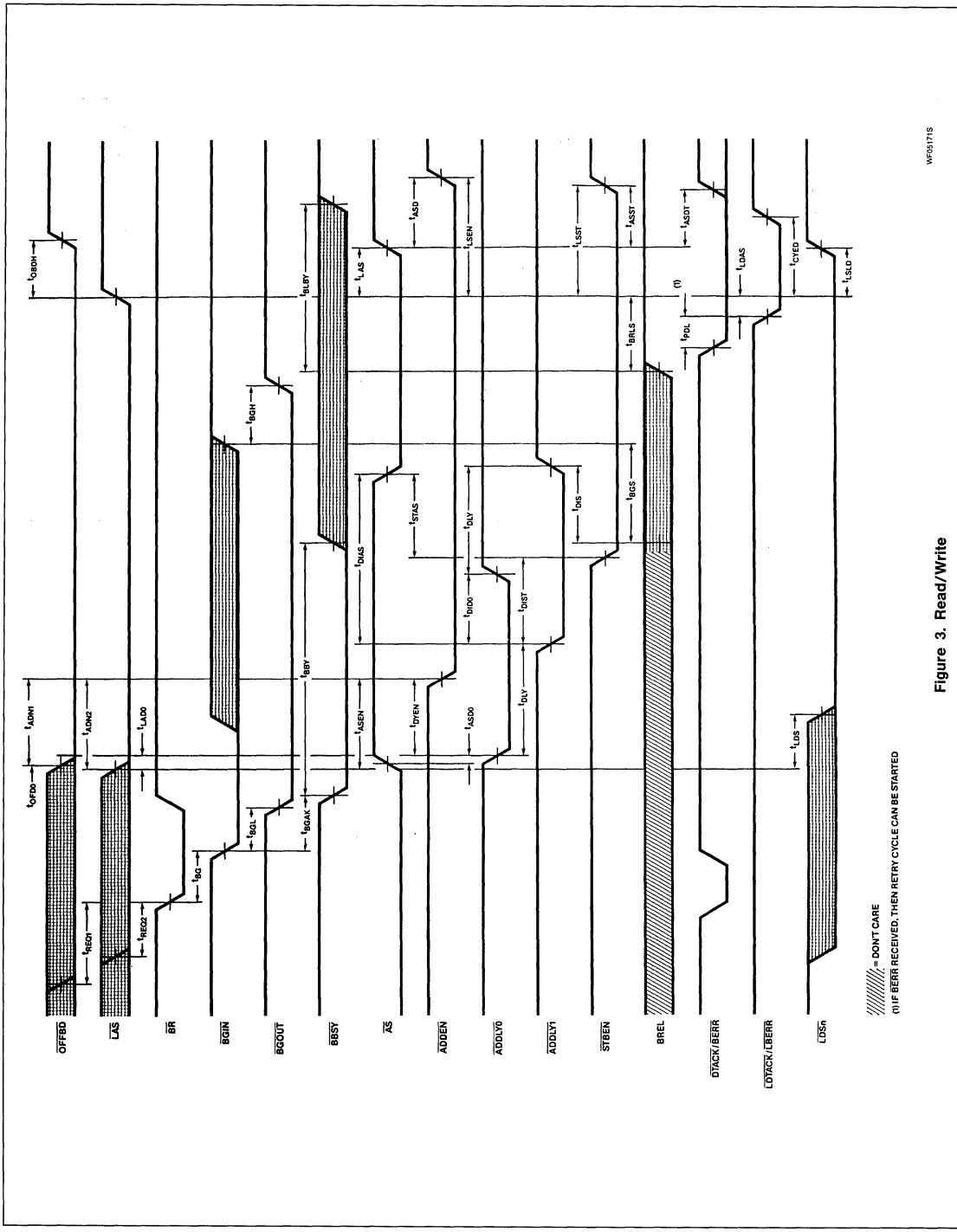


Figure 3. Read/Write

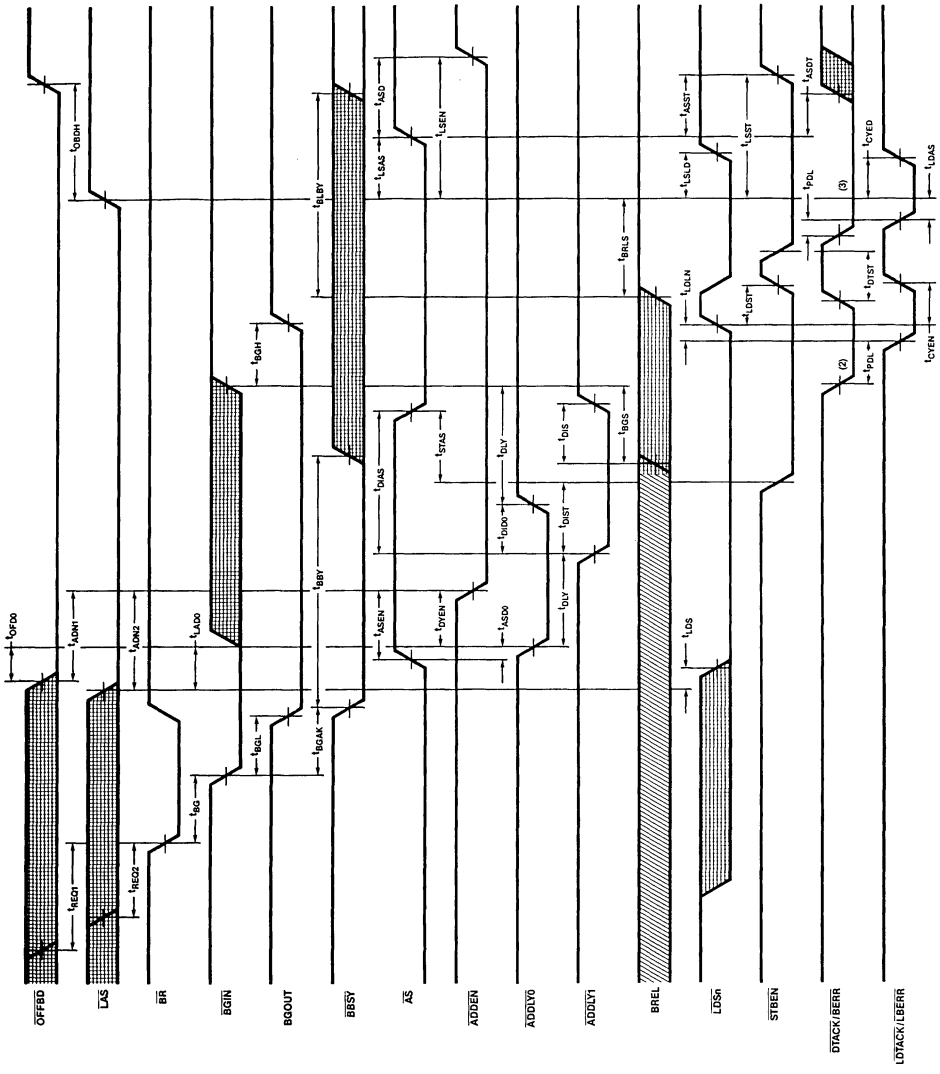


Figure 4. Read/Modify/Write

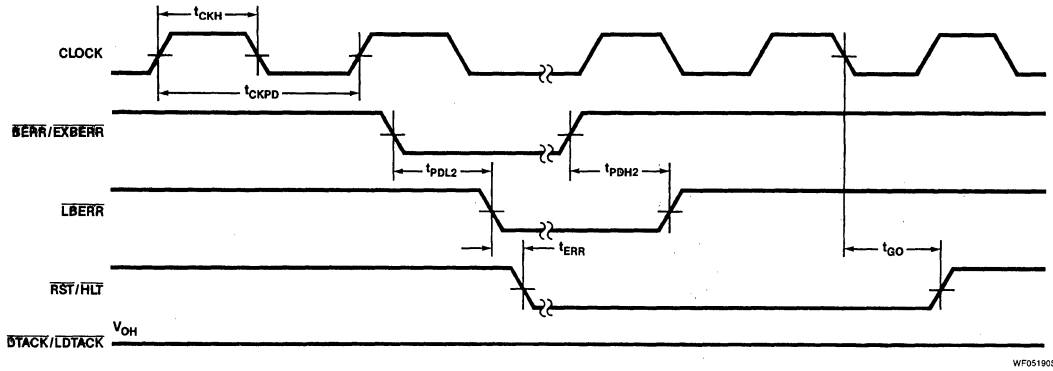


Figure 5. Bus Error/Retry (Last Cycle not Terminated by Error)

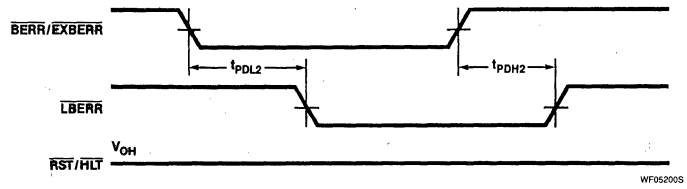


Figure 6. Bus Error/Retry (Last Cycle Terminated by Error)

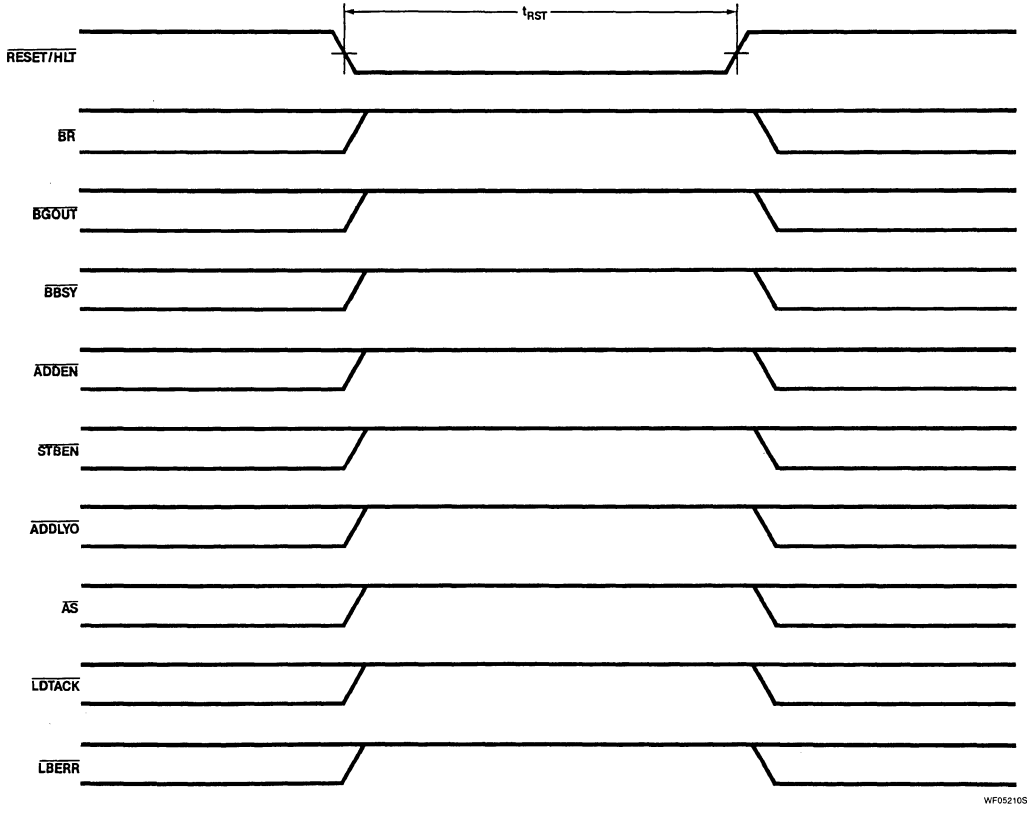


Figure 7. Reset

Signetics

SCB68430
Direct Memory Access Interface (DMAI)

Preliminary Specification

Microprocessor Products

DESCRIPTION

The SCB68430 Direct Memory Access Interface (DMAI) is a single channel interface circuit which is intended to complement the performance and architectural capabilities of the SCN68000 microprocessor. The DMAI functions by transferring a series of operands (data) between memory and a device: operand sizes may be byte, word, or long word. A block is a sequence of operands: the number of operands in the block is determined by a transfer count stored within the DMAI. The SCB68430 can be programmed to utilize single cycle (cycle stealing) or burst data transfers.

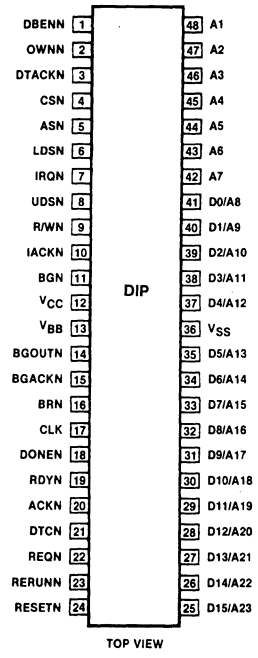
The DMAI provides two interfaces. The microprocessor interface is fully compatible with the SCN68000 microprocessor. The device interface includes lines for requesting, acknowledging, controlling, and timing the data transfers. The DMAI is a single-channel subset of the other 68000 family DMA controllers (68440 and 68450). It is software compatible with these devices and provides similar interfacing signals to both the system bus and the device.

The SCB68430 is constructed using Signetics ISL bipolar technology.

FEATURES

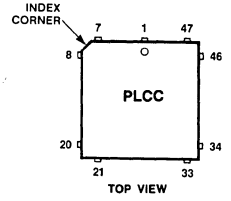
- **Bus compatible with SCN68000 microprocessor**
- **Software compatible with other 68K family DMA controllers**
- **Single address transfers**
- **Cycle steal and burst mode operation**
- **Bus arbitration daisy chain**
- **Automatic rerun on bus error**
- **Supports 32-bit transfers for VME bus**
- **Supports SCN68000 vectored interrupts**
- **24-bit address counter**
- **16-bit transfer counter**
- **Maximum transfer rate of 5 Mbytes per second**
- **Signetics ISL bipolar technology**

PIN CONFIGURATION



TOP VIEW

CD00650S



TOP VIEW

CD00652PS

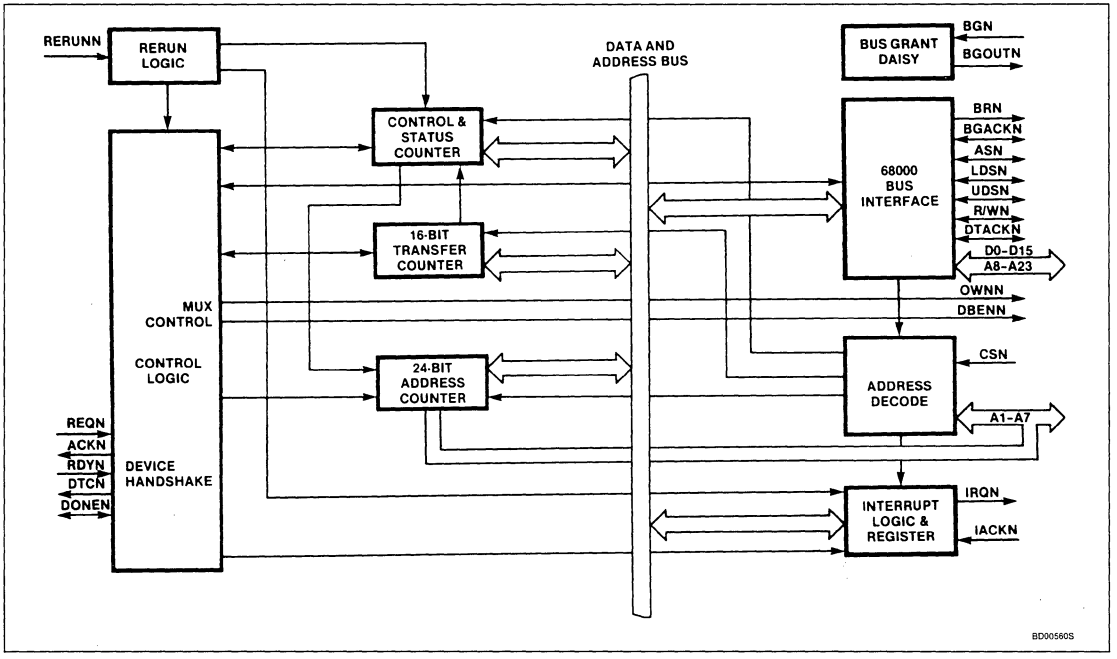
Pin	Function	Pin	Function	Pin	Function
1	NC	19	CLK	36	D7/A15
2	DBENN	20	DONEN	37	D6/A14
3	OWNN	21	RDYN	38	D5/A13
4	DTACKN	22	ACKN	39	V_{SS}
5	CSN	23	DTCN	40	NC
6	ASN	24	REQN	41	D4/A12
7	LDSN	25	RERUNN	42	D3/A11
8	IRQN	26	RESETN	43	D2/A10
9	UDSN	27	NC	44	D1/A9
10	R/WN	28	D15/A23	45	D0/A8
11	IACKN	29	D14/A22	46	A7
12	BGN	30	D13/A21	47	A6
13	V_{CC}	31	D12/A20	48	A5
14	NC	32	D11/A19	49	A4
15	V_{BB}	33	D10/A18	50	A3
16	BGOUTN	34	D9/A17	51	A2
17	BGACKN	35	D8/A16	52	A1
18	BRN				

Direct Memory Access Interface (DMAI) SCB68430

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C	
	10MHz	12.5MHz
Ceramic DIP	SCB68430CAI48	SCB68430CCI48
Plastic DIP	SCB68430CAN48	SCB68430CCN48
Plastic LCC	SCB68430CAA52	SCB68430CCA52

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
A1 – A7	48 – 42	I/O	**Address Lines:** Active high, three-state. In the MPU mode, these low order address lines specify which internal register of the DMAI is being accessed. In DMA mode, A1 – A7 are ouputs which provide the low order address bits of the location being accessed. Three-stated in IDLE Mode.
A8 – A23/ D0 – D15	41 – 37 35 – 25	I/O	**Address/Data Lines:** Active high, three-state. These lines are time multiplexed for data and address leads. The lines OWNN, RWN, CSN, and DBENN are used to control the demultiplexing of the address and data using external circuitry. In MPU mode, the bidirectional data lines (D0 – D15) are used to transfer data between the MPU and the DMAI. In the DMA mode, A8 – A23 provide the high order address bits of the location being accessed. Three-stated in IDLE mode.
ASN	5	I/O	**Address Strobe:** Active low, three-state. In MPU and IDLE modes, ASN is an input which indicates that the current bus master has placed a valid address on the bus. It is monitored by the DMAI during bus arbitration to ascertain that the previous bus master has completed the current bus cycle. In DMA mode, it is an output indicating that the DMAI has placed a valid address on the bus.
UDSN	8	I/O	**Upper Data Stobe:** Active low, three-state. In MPU and IDLE modes, UDSN is an input which indicates that the upper data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning.
LDSN	6	I/O	**Lower Data Strobe:** Active low, three-state. In MPU and IDLE modes, LDSN is an input which indicates that the lower data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning.
R/WN	9	I/O	**Read/Write:** Active high for read, low for write, three-state. In MPU mode, R/WN is an input which controls the direction of data flow through the DMAI's input/output data bus interface and, if required, through an external data bus buffer. R/WN high causes the DMAI to place the data from the addressed register on the data bus, while R/WN low causes the DMAI to accept data from the data bus. In DMA mode, R/WN is an output to memory and I/O controllers indicating the type of bus cycle. It is held three-stated during IDLE mode.
CSN	4	I	**Chip Select:** Active low. When low, places the DMAI into the MPU mode. This input signal is used to select the DMAI for programmed data transfers. These transfers take place over D0 – D15 as controlled by the R/WN and A1 – A7 inputs. The DMAI is deselected when CSN is high. CSN is ignored during DMA mode.
DTACKN	3	I/O	**Data Transfer Acknowledge:** Active low, three-state. In MPU mode, DTACKN is asserted on a write cycle to indicate that the data on the bus has been latched, and on a read cycle or interrupt acknowledge cycle to indicate that valid data is present on the bus. The signal is negated (driven high) when completion of the cycle is indicated by negation of the CSN or IACKN input, and returns to the inactive third state a short time after it is negated. In DMA Mode, DTACKN is an input monitored by the DMAI to determine when the addressed device (memory) has latched the data (write cycle) or put valid data on the bus (read cycle).
RESETN	24	I	**Master Reset:** Active low. Assertion of this pin clears internal control registers (See table 1), initializes the interrupt vector register to H'0F', and sets the status register to the default value B'0000 000X', where X is the state of RDYN. All bidirectional I/O lines are three-stated and the DMAI is placed in the IDLE mode.
CLK	17	I	**Clock:** Active high. Usually the system clock, but may be any clock meeting the electrical specifications. Used by the DMAI to synchronize device functions and external control lines, and may not be gated off at any time.
IRQN	7	O	**Interrupt Request:** Active low, open collector. This output is asserted, if interrupts are enabled, upon end of transfer, on occurrence of a bus error, and on receipt of an abort from the MPU. The CPU can read the status register to determine the interrupting condition(s), or can respond with an interrupt acknowledge cycle to cause the DMAI to output an interrupt vector on the data bus.
IACKN	10	I	**Interrupt Acknowledge:** Active low. When asserted, indicates that the current cycle is an interrupt acknowledge cycle. The DMAI normally responds by placing the contents of the interrupt vector register on the data bus and asserting DTACKN. IACKN is not serviced if the DMAI has not generated an interrupt request.
BRN	16	O	**Bus Request:** Active low, open collector. BRN is asserted by the DMAI to request ownership of the bus after a DMA request is sensed on the REQN input from the I/O device. It is negated when the bus has been granted (BGN low) and BGACKN has been asserted, or, in burst DMA request mode, if the I/O device negates its request at least one clock cycle before BGACKN is asserted.
BGN	11	I	**Bus Grant:** Active low. BGN indicates to the DMAI that it is to be the next bus master. This signal is originated by the MPU and propagated via a daisy chain or other prioritization mechanism. After BGN is asserted, the DMAI waits until DTACKN, ASN, and BGACKN have become inactive before assuming ownership of the bus by asserting BGACKN.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
BGOUTN	14	O	**Bus Grant Output:** Active low. Daisy chain output which is asserted by the DMAI when BGN is asserted and the DMAI does not have a bus request pending.
BGACKN	15	I/O	**Bus Grant Acknowledge:** Active low, three-state. As an input, BGACKN is monitored by the DMAI during the bus arbitration cycle to determine when it can assume ownership of the bus (BGACKN negated). In DMA mode, it is asserted by the DMAI to indicate that it is the bus master. Three-stated in MPU and IDLE modes.
RERUNN	23	I	**Rerun:** Active low. This input is asserted by external error detect logic to indicate a bus error. In DMA mode, the DMAI stops operation and three-states the data, address, and control lines, except BGACKN. It remains halted until RERUNN becomes inactive, and then re-tries the last bus cycle. If RERUNN is asserted again, the DMAI sets the ERR bit in the status register, stops DMA operation, releases the bus, and interrupts the CPU, if interrupts are enabled, responding with a special interrupt vector when IACKN is asserted. Not monitored in MPU and IDLE modes.
REQN	22	I	**DMA Request:** Active low. This input from the I/O device requests service from the DMAI and causes the DMAI to request control of the bus. In burst mode, the input is level sensitive, and the DMAI releases the bus after REQN is negated and the current DMA cycle is completed. In cycle steal mode, the REQN input is negative edge triggered. A negative going edge must occur at least one clock cycle before DTCN is asserted to accomplish continuous transfer cycles but not earlier than beginning of master cycle.
ACKN	20	O	**DMA Request Acknowledge:** Active low. ACKN is asserted by the DMAI to indicate that it has gained the bus and the requested bus cycle is now beginning. It is asserted at the beginning of every bus cycle after ASN has been asserted, and is negated at the end of every bus cycle.
RDYN	19	I	**Device Ready:** Active low. RDYN is asserted by the requesting device to indicate to the DMAI that valid data has either been stored or put on the bus. If negated, it indicates that the data has not been stored or presented, causing the DMAI to enter wait states. RDYN can be held low continuously if the device is fast enough so that wait states are not required.
DTCN	21	O	**Device Transfer Complete:** Active low. In DMA mode, DTCN is asserted by the DMAI to indicate to the device that the requested data transfer is complete. On a write to memory operation, it indicates that the data provided by the device has been successfully stored. On a read from memory operation, it indicates to the device that the data from memory is present on the data bus and should be latched.
DONEN	18	I/O	**Done:** Active low, open collector. As an output, DONEN is asserted by the DMAI concurrent with the ACKN output to indicate to the device that the transfer count is exhausted and that the DMAI's operation is completed as a result of that transfer. As an input, if asserted by the device before the transfer count became zero, it causes the DMAI to abort service and generate an interrupt request, if interrupts are enabled.
OWNN	2	O	**Own:** Active low, open collector. This output is asserted by DMAI during the DMA mode to indicate bus mastership. It can be used to enable external address/data and control buffers. Inactive in MPU and IDLE modes.
DBENN	1	O	**Data Bus Enable:** Active low, open collector. Asserted by the DMAI when CSN is asserted or when IACKN is asserted and the DMAI has an interrupt request pending. Can be used to enable bidirectional data buffers for D0 – D15. Inactive in IDLE and DMA mode.
V_{CC}	12	I	**Power Supply:** +5 volt power input.
V_{BB}	13	I	**Power Supply:** +1.5 volt power input.
V_{SS}	36	I	**Ground:** Signal and power ground input.

PIN DESCRIPTION

The Pin Description table describes the function of each of the pins of the DMAI. Signal names ending in 'N' are active low. All other signals are active high. In the descriptions, 'MPU mode' refers to the state when the DMAI is chip selected. The term 'DMA mode' refers to the state when the DMAI assumes ownership of the bus. The DMAI is in the 'IDLE mode' at all other times.

In this data sheet signals are discussed using the terms 'active' and 'inactive' or 'asserted' and 'negated' independent of whether the signal is active in the high (logic one) state or the low (logic zero) state. Refer to the individual pin descriptions for the definition of the active level of each signal.

REGISTERS AND COUNTERS

Register Map

The internal accessible register organization of the DMAI is shown in table 1. The following rules apply to all registers:

1. A read from a reserved location in the map results in a read from the 'null register'. The null register returns all ones for data and results in a normal bus cycle. A write to one of these locations results in a normal bus cycle but no write occurs.
2. Unused bits of a defined register are read as indicated in the register descriptions.
3. All registers are addressable as 8-bit quantities. To facilitate operation with the 68K MOVEP instruction, addresses are ordered such that certain sets of registers may also be accessed as words or long words.

Direct Memory Access Interface (DMAI) SCB68430

The operation of the DMAI is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The contents of certain control and status registers are initialized on RESET.

To provide compatibility with the other 68K family DMA controllers, control and status bits are mapped in bit positions equivalent to where they are located in the register map of the other devices. Bits which are used in the other devices but not in the DMAI are assigned default values. If upward compatibility to the other controllers is required, the programmer should use these default values when writing the control words to the registers, although they have no effect in the DMAI. When a register is read, the default value is returned regardless of the value used when the register is programmed. The default value is indicated by '(x)' in unused bit positions in the register formats, which are illustrated in table 2.

Table 1. DMAI ADDRESS MAP

ADDRESS BITS[1,2] 7 6 5 4 3 2 1 0	ACRONYM	REGISTER NAME	MODE	AFFECTED BY RESET
d d 0 0 0 0 0 0	CSR	Channel Status Register	R/W[3]	Yes
d d 0 0 0 0 0 1	CER	Channel Error Register	R	Yes
d d 0 0 0 0 1 0		Reserved		
d d 0 0 0 0 1 1		Reserved		
d d 0 0 0 1 0 0	DCR	Device Control Register	R/W	Yes
d d 0 0 0 1 0 1	OCR	Operation Control Register	R/W	Yes
d d 0 0 0 1 1 0	SCR	Sequence Control Register	R/W[4]	No
d d 0 0 0 1 1 1	CCR	Channel Control Register	R/W	Yes
d d 0 0 1 0 0 0		Reserved		
d d 0 0 1 0 0 1		Reserved		
d d 0 0 1 0 1 0	MTCH	Memory Transfer Counter High	R/W	No
d d 0 0 1 0 1 1	MTCL	Memory Transfer Counter Low	R/W	No
d d 0 0 1 1 0 0	MACH	Memory Address Counter High	R/W[4]	No
d d 0 0 1 1 0 1	MACMH	Memory Address Counter Middle High	R/W	No
d d 0 0 1 1 1 0	MACML	Memory Address Counter Middle Low	R/W	No
d d 0 0 1 1 1 1	MACL	Memory Address Counter Low	R/W	No
d d 0 1 d d d d		Reserved		
d d 1 0 0 0 d d		Reserved		
d d 1 0 0 1 0 0		Reserved		
d d 1 0 0 1 0 1	IVR	Interrupt Vector Register – Normal	R/W	Yes
d d 1 0 0 1 1 0		Reserved		
d d 1 0 0 1 1 1	IVR	Interrupt Vector Register – Error	R/W	Yes
d d 1 0 1 0 d d		Reserved		
d d 1 0 1 1 0 0		Reserved		
d d 1 0 1 1 0 1	CPR	Channel Priority Register	R/W[4]	No
d d 1 0 1 1 1 0		Reserved		
d d 1 0 1 1 1 1		Reserved		
d d 1 1 d d d d		Reserved		

NOTES:
1. A0 = 0 for UDSN asserted, A0 = 1 for LDSN asserted.
2. 'd' designates don't care.
3. A write to this register may perform a status resetting operation.
4. This register is a dummy register present only to provide compatibility with other 68K family DMA controllers. A write to this register has no effect on the DMAI.

Table 2. REGISTER BIT FORMATS

DEVICE CONTROL REGISTER

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
DCR	EXTERNAL REQUEST MODE 0 = BURST 1 = CYCLE STEAL	NOT USED (0)	NOT USED (1)	NOT USED (1)	NOT USED (*)	NOT USED (0)	NOT USED (0)	NOT USED (0)

*Should be programmed as '0' for SIZE (OCR[5:4]) = 00 and as '1' otherwise. When read, the value of this bit is OCR[5], .OR.OCR[4].

OPERATION CONTROL REGISTER (OCR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
OCR	DIRECTION 0 = MEM TO DEV 1 = DEV TO MEM	NOT USED (0)	OPERAND SIZE 00 = BYTE 01 = WORD (16 BIT) 10 = LONG WORD* 11 = WORD (32-BIT)*		NOT USED (0)	NOT USED (0)	NOT USED (1)	NOT USED (0)

*Long word and 32-bit word modes are not supported by 68440. 32-bit word mode is not supported by 68450.

SEQUENCE CONTROL REGISTER (SCR)

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
SCR	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (1)	NOT USED (0)	NOT USED (0)

CHANNEL CONTROL REGISTER (CCR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
CCR	START 0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	SOFTWARE ABORT 0 = NO 1 = YES	INTERRUPT ENABLE 0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	NOT USED (0)

CHANNEL STATUS REGISTER (CSR)

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
CSR	CHANNEL OPERATION COMPLETE 0 = NO 1 = YES	NOT USED (0)	NORMAL DEVICE TERMINATE 0 = NO 1 = YES	ERROR 0 = NO 1 = YES	CHANNEL ACTIVE 0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	READY INPUT STATE 0 = LOW 1 = HIGH

CHANNEL ERROR REGISTER (CER)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
CER	NOT USED (0)	NOT USED (0)	NOT USED (0)	ERROR CODE 00000 = NO ERROR 01001 = BUS ERROR 10001 = SOFTWARE ABORT				

CHANNEL PRIORITY REGISTER (CPR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
CPR	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)

Device Control Register (DCR)

[15] External Request Mode

This bit selects whether the DMAI operates in burst or cycle steal mode.

0 Burst mode. This mode allows a device to request the transfer of multiple operands using consecutive bus cycles. In this mode the request (REQN) line is an active low input which is asserted by the device to request an operand transfer. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. If the request line is active when the DMAI asserts ACKN, and remains active at least until the DMAI asserts device transfer complete (DTCN), the DMAI recognizes a valid request for another operand, which will be transferred during the next bus cycle. If the request line is negated before the DMAI asserts DTCN, the DMAI relinquishes the bus and waits for the next request, but the current transfer will be completed.

1 Cycle steal mode. In this mode, the device requests an operand transfer by generating a falling edge on the request (REQN) line. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN)

output. The request line must be in the inactive state for at least one clock cycle before a request is made. After a request has been asserted, it must remain at the assertion level for at least one clock cycle. If another request is received before the first operand part of a former request is acknowledged, the second request is not recognized. Normally, the DMAI will relinquish the bus after servicing a valid request. However, if the device generates a new request before the DMAI asserts DTCN for the last operand part, the DMAI will retain ownership of the bus and that request will be serviced before the bus is relinquished.

Operation Control Register (OCR)

[7] Direction

0 Transfer is from memory to device.

1 Transfer is from device to memory.

[5:4] Operand Size

The programming of these bits determine whether UDSN, LDSN, or both are generated during the transfer cycle and the increment by which the memory address counter (MAC) is changed in each transfer cycle.

00 Byte. The operand size is 8 bits. The MAC is incremented by one after each operand transfer. If the LSB of the MAC is a '0', UDSN is asserted during the transfer. If the LSB of a MAC is a '1', LDSN is asserted during the transfer. The transfer counter decrements by one before each byte is transferred.

01 Word. The operand size is 16 bits. The MAC is incremented by two after each operand transfer. The value of the LSB of the MAC is ignored and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before each word is transferred.

10 Long word. The operand size is 32 bits. The operand is transferred as two 16-bit words. The MAC is incremented by two after each 16-bit word is transferred. The value of the LSB of the MAC is ignored and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before the entire long word is transferred. Note that this mode is not implemented in the 68440.

11 Double word. The operand size is 32 bits. The operand is transferred as a single 32-bit word. The MAC is incremented by four after each operand transfer. The value of the two LSBs of the MAC is ignored (the A1 output will always be a zero in this mode) and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before the double word is transferred. Note that this mode is not implemented in the 68440 or 68450; it is included in the DMAI to support VME bus operations.

Sequence Control Register (SCR)

This register serves no function in the DMAI. It is included only to provide compatibility with the programming for the 68440 and 68450 DMA controllers.

Channel Control Register (CCR)

[7] Start Operation

0 No start pending.

1 Start operation. The start bit is set to initiate operation of the DMAI. The memory address counter and the memory transfer counter should have been previously initialized, and all bits of the channel status register (CSR) should have previously been reset. The DMAI initiates operation by clearing any pending requests, clearing the start bit, and setting the channel active bit in the CSR. The DMAI is then ready to receive requests for an operation. The channel cannot be started if any of the internal status bits in the CSR (CSR[15:11]) have not been cleared.

A pending start cannot be reset by a write to the register. START can be cleared only by the DMAI when it starts operation or by setting the software abort bit (CCR[4]).

[4] Software Abort

0 Do not abort.

1 Abort operation. Setting this bit terminates the current operation of the DMAI and places it in the IDLE state. The channel operation complete and error bits in the CSR are set, the channel active bit in the CSR is reset, and an ABORT ERROR condition is signaled in the CER. Setting this bit causes a pending start to be reset.

[3] Interrupt Enable

0 Interrupts not enabled.

1 Enable interrupts. An interrupt request is generated if the channel operation complete bit in the CSR is set. When the IACKN input is asserted, the DMAI returns the normal interrupt vector if the error bit in the CSR is not set, or the error interrupt vector if error is set.

Channel Status Register (CSR)

A read of this register provides the status of the DMAI. The COC, NDT, and ERR bits can be cleared by writing a '1' to the bit positions of the register which are to be cleared. Those bit positions which are written with a '0' remain unaffected.

[15] Channel Operation Complete

This bit is set following the termination, whether successful or not, of any DMAI operation and indicates that the DMA transfer has completed. This bit must be cleared to start another channel operation.

[13] Normal Device Termination

This bit is set when the device terminates the DMAI operation by asserting the DONEN line while the device was being acknowledged. This bit must be cleared to start another channel operation.

[12] Error

This bit is used to report that the DMAI's operation was terminated due to the occurrence of an error. The condition which caused the error can be determined by reading the channel error register (CER). This bit must be cleared to start another channel operation. When this bit is cleared, the CER is also cleared.

[11] Channel Active

This bit is set after the channel has been started and remains set until the channel operation terminates. It is then automatically cleared by the DMAI. The bit is unaffected by the write operations.

[8] Ready Input State

This bit reflects the state of the RDYN input at the time the CSR is read. The bit is a '0' if RDYN is low and a '1' if RDYN is high. This bit is unaffected by write or reset operations.

Channel Error Register (CER)

[4:0] Error Code

This field indicates the source of error when an error is indicated in CER[12]. The contents of this register are cleared when CER[12] is cleared.

00000 No error.

01001 Bus error. A bus error occurred during the last bus cycle generated by the DMAI. See rerun description in OPERATION section.

10001 Software abort. The channel operation was terminated by a software abort command. See CCR[4].

Channel Priority Register (CPR)

This register serves no function in the DMAI. It is included only to provide compatibility with the programming for the other 68K family DMA controllers.

Memory Address Counter (MACH, MACMH, MACML, MACL)

The 32-bit memory address counter is used to program the memory location where the first operand to be transferred is located or is to be transferred to, depending on the direction of transfer. The counter must be initialized prior to beginning the transfer of a block of data and then increments automatically depending on the operand length, as de-

scribed in the Operation Control Register description.

Only the least significant 24 bits of the counter (MACMH, MACML, and MACL) are implemented in the DMAI. The most significant byte of the counter, MACH, is provided only to allow compatibility with programming of the 68440 and 68450. Writing to MACH has no effect on the DMAI operation. Reading MACH always returns H'00'.

Memory Transfer Counter (MTCH, MTCL)

The 16-bit memory transfer counter programs the number of operands to be transferred by the DMAI. The counter must be initialized prior to beginning the transfer of a block of data and then decrements once per operand transfer (regardless of operand size) until it reaches the terminal value of zero. Channel operation then terminates and the COC bit in the CSR will be asserted.

Interrupt Vector Register (IVR)

The IVR contains the value to be placed on the data bus upon receipt of an interrupt acknowledge from the MPU. Only the seven most significant bits of the programmed value are used by the DMAI. The output vector from the DMAI contains a zero in the least significant bit position if a normal termination occurred (error bit not set) and contains a one in the least significant bit position if termination was due to an error (error bit set).

The contents of this register are initialized to H'0F' by a reset. The value returned will be H'0F', regardless of the error state, until the register is programmed by the MPU.

To provide compatibility with the other 68K family DMA controllers, the IVR has two addresses (see table 1). If program compatibility is required, the value written at the normal IVR address should have a zero as its LSB, and the value written at the error IVR address should be the same but with the LSB equal to one.

OPERATION

A DMAI operation proceeds in three principal phases. During the initialization phase, the MPU configures the channel control registers, loads the initial memory address and transfer count, and starts the channel. During the transfer phase, the DMAI accepts requests for transfers from the device, arbitrates for and acquires ownership of the bus, and provides for addressing and bus controls for the transfers. The termination phase occurs after the operation is complete, when the DMAI reports the status of the operation.

Operation Initiation

After having programmed the control registers, the memory address counter, and the memory transfer counter, the MPU sets the start bit (CCR[7]). The DMAI initiates the operation by clearing any pending requests, clearing the start bit, and setting the channel active bit in the CSR. The DMAI is then ready to receive valid requests for an operation.

The channel cannot be started if any of the internal status bits in the CSR (CSR[15:11]) have not been cleared. An error is not signaled if this condition occurs. The only indication of this state is that the start bit remains set in the CCR. A pending start cannot be reset by a write to the register. START can be cleared only by the DMAI when it starts operation or by setting the software abort bit (CCR[4]).

Device/DMAI Communication

Communication between the peripheral device and the DMAI is accommodated by five signal lines:

Request (REQN)

The device makes a request for service by asserting the request line. The DMAI can operate in either the burst request mode or the cycle stealing request mode, as programmed by the external request mode bit (DCR[15]).

The burst mode allows a device to request the transfer of multiple operands using consecutive bus cycles. In this mode the request line is an active low input. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. If the request line is active when the DMAI asserts ACKN, and remains active at least until the DMAI asserts device transfer complete (DTCN), the DMAI recognizes a valid request for another operand, which will be transferred during the next bus cycle. If the request line is negated before the DMAI asserts DTCN, the DMAI relinquishes the bus and waits for the next request. For long word transfers (2 x 16), the request must be asserted at least until the acknowledge for the second part of the operand has been asserted.

In the cycle steal mode, the device requests an operand transfer by generating a falling edge on the request line. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. The request line must be in the inactive state for at least one clock cycle before a request is made. After a request has been asserted, it must remain at the assertion level for at least one clock cycle. If another request is received before the first operand part of a former request is acknowledged, the second request is not recognized. Normally, the DMAI will relinquish the bus after servicing a valid request. However, if the device generates a new request before the DMAI asserts DTCN for the last operand part, the DMAI will retain ownership of the bus and that request will be serviced before the bus is relinquished.

Acknowledge (ACKN)

The DMAI asserts the acknowledge line, which implicitly addresses the device making the request, during transfers to and from the device. The line may be used to control buffering circuits between the data bus and the MPU bus.

During burst mode, REQN must not be disasserted for less than one CLK period plus four RC time constants, where R is the value of the resistor used for the pullup on BRN and C has a typical value of 20pF.

Ready (RDYN)

Ready is an active low input which is asserted by the requesting device to indicate to the DMAI that valid data has either been stored or put on the bus. If negated, it indicates that the data has not been stored or presented, causing the DMAI to enter wait states until RDYN is asserted. RDYN can be held low continuously if the device is fast enough so that wait states are not required. The current state of the ready input is reflected in CSR[8].

Done (DONEN)

Done is a bidirectional active low signal. As an output, it is asserted and negated by the DMAI concurrent with the ACKN output of the last operand part to indicate to the device that the memory transfer count is exhausted and that the DMAI's operation is completed as a result of that transfer.

The DMAI also monitors the state of the line while acknowledging a device. If the device asserts DONEN, the DMAI will terminate operation after the transfer of the current operand. In this case the DMAI clears the channel active bit and sets the channel operation complete and normal device termination bits in the CSR. If both the DMAI and the device assert DONEN, the device termination is not recognized, but the operation does terminate.

Device Transfer Complete (DTCN)

DTCN is an active low output which is asserted by the DMAI to indicate to the device that the requested data transfer is complete. On a write to memory operation, it indicates that the data provided by the device has been successfully stored. On a read from memory operation, it indicates to the device that the data from memory is present on the data bus and should be latched. DTCN is not asserted if assertion of the RERUNN input terminates the bus cycle.

Bus Arbitration

Upon receiving a valid request for a transfer from the device, the DMAI will arbitrate for and obtain ownership of the system bus.

The DMAI indicates that it wishes to become the bus master by asserting its bus request (BRN) output. This is a wire-ORed signal that indicates to the MPU that some external device requires control of the bus. The processor is effectively at a lower priority level than external devices and will relinquish the bus after it has completed the last bus cycle it has started. The processor puts the bus up for external arbitration by asserting its bus grant (BGN) output. This signal may be routed through a daisy chain (such as provided by the DMAI) or through some other priority-encoded network. When the DMAI making the bus request receives the bus grant (indicated by its BGN input being asserted), it is to be the next bus master. It waits until address strobe (ASN), data transfer acknowledge (DTACKN) and bus grant acknowledge (BGACKN) become inactive and then assumes ownership of the bus by asserting its own BGACKN output. The DMAI then negates the BRN output and proceeds with the data transfer phase. After this phase is completed, the DMAI relinquishes bus ownership by negating the BGACKN output.

In burst DMA mode, detection of an active low request input after the DMAI operation has been started will begin the bus arbitration cycle. However, if the device negates its request at least one clock cycle before the DMAI asserts BGACKN, the DMAI will negate its bus request and will not assume ownership of the bus.

Data Transfers

The actual transfer of data between the memory and the device occurs during the data transfer phase. All transfers occur during a single cycle except in the case of long word operands, in which case two cycles are used to transfer the operand as two 16-bit words. The transfers take place using a 'single address' protocol; the DMAI addresses the memory via the bus address lines, while the device is implicitly addressed via the acknowledge output.

When a request is generated using the request method programmed in the control register, the DMAI obtains the bus and asserts acknowledge to notify the device that a transfer is to take place. The DMAI asserts all S68000 bus control signals needed for the transfer and holds them until the device responds with ready. The bus cycle then terminates normally. Ready may be tied low (asserted) if the device is fast enough.

When the transfer is from memory to the device, data is valid when DTACKN is asserted by the memory and remains valid until the data strobe(s) are negated. The assertion of DTCN from the DMAI can be used to latch the data, as the data strobes are not removed until one-half clock after the assertion of DTCN.

When the transfer is from device to memory, the data must be valid on the bus before the DMAI asserts the data strobe(s). The device indicates valid data by asserting ready. The DMAI then asserts the strobes and holds them asserted until the memory accepts the data, indicated by the assertion of DTACKN. The DMAI then asserts DTCN and negates the data strobes.

Flow charts for these operations are shown in figures 1 and 2. Refer to the timing section for the equivalent timing diagrams.

Operation Termination

Termination of the block transfer occurs under the conditions detailed below.

Terminal Count

As part of each transfer of an operand, the DMAI decrements the memory transfer counter. If this counter is decremented to zero, the operand is the last operand of the block. The DMAI operation is complete and it notifies the device of completion by asserting the DONEN output during the last operand transfer cycle. When the transfer has been completed, the channel active bit in the CSR is cleared and the COC bit is set, unless an error occurs.

Device Termination

The DMAI monitors the state of the DONEN line while acknowledging a device transfer request. If the device asserts DONEN, the DMAI will terminate operation after the transfer of the current operand. When the transfer has been completed, the DMAI clears the channel active bit and sets the COC and normal device termination bits in the CSR. If both the DMAI and the device assert DONEN, the device termination is not recognized, but the operation does terminate.

Software Abort

The software abort bit (CCR[4]) allows the MPU to abort the current operation of the DMAI. The COC and error bits in the CSR are set, the channel active bit in the CSR is cleared, and an abort error condition is signaled in the CER.

Rerun Error

The DMAI provides a rerun input (RERUNN) to indicate a bus exception condition. RERUNN must arrive prior to or in coincidence with DTACKN in order to be recognized, and the DMAI verifies that the line has been stable for two clock cycles before acting on it. The occurrence of a rerun during a DMAI bus cycle forces it to terminate the bus cycle in an orderly manner.

When the assertion of rerun is verified, the DMAI stops operation and three-states the data, address, and control lines, except BGACKN, so that it retains ownership of the bus. It remains halted until rerun becomes inactive, and then re-tries the last bus cycle. If rerun is asserted again, the DMAI stops DMA operation, releases the bus, sets the error and COC bits in the CSR, clears the active bit in the CSR, and sets the error code in the CER to indicate a bus error.

While stopped due to assertion of rerun, the DMAI does not generate any bus cycles and will not honor any requests until it is removed. However, the DMAI still recognizes requests.

Error Recovery Procedure

If an error occurs during a DMA transfer such that the DMAI stops the DMA operation, information is available to the operating system for an error recovery routine.

The information available to the operating system consists of the memory address counter, the memory transfer counter, and the control, status, and error registers. The DMAI decrements the memory transfer counter before attempting a DMA operation, so the register will contain the count minus one of the attempted transfer. The memory address counter will contain the address at which the DMA operation was attempted.

Reset

The reset input (RESETN) provides a means of resetting and initializing the DMAI from an external source. If the DMAI is a bus master when reset is received, the DMAI relinquishes the bus. Reset clears the control and error registers, sets all bits of the status register except CSR[8] to zero, and initializes the interrupt vector register to H'0'F.

Interrupts

The interrupt enable bit (CCR[3]) determines whether the DMAI generates interrupt requests. When the bit is set, an interrupt request is generated if the channel operation complete bit in the CSR is set. When the IACKN input is asserted, and the DMAI has an interrupt request pending, the DMAI returns an interrupt vector on the data bus.

The interrupt vector issued is the contents of the IVR. Only the seven most significant bits of the programmed value are used by the DMAI. The vector from the DMAI contains a zero in the LSB position if a normal termination occurred (error bit not set) and contains a one in the LSB position if termination was due to an error (error bit set).

The contents of this register are initialized to H'0F' by a reset. The value returned will be H'0F', regardless of the error state, until the register is programmed by the MPU.

To provide compatibility with the other 68K family DMA controllers, the IVR has two addresses (see table 1). If program compatibility is required, the value written at the

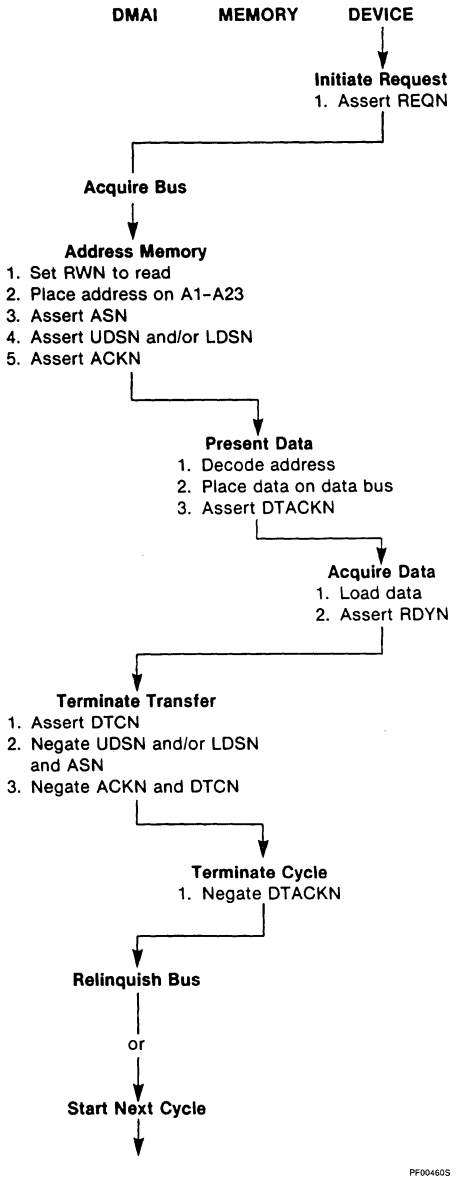


Figure 1. Transfer from Memory to Device Flowchart

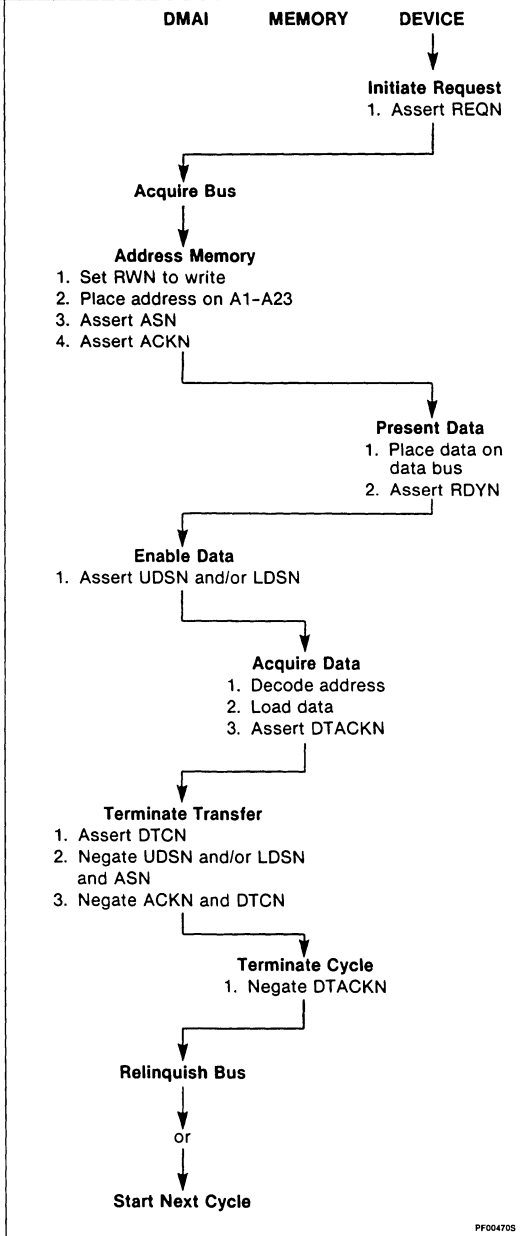


Figure 2. Transfer from Device to Memory Flowchart

normal IVR address should have a zero as its LSB, and the value written at the error IVR address should be the same but with the LSB equal to one.

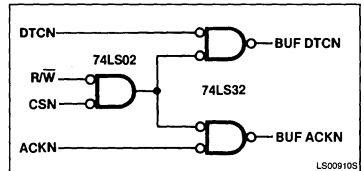
APPLICATIONS

Figure 3 illustrates a typical interconnection of the DMAI in a 68000 based system.

DESIGN NOTE

When clearing the error bit in CSR (bit 12) after a DMAI abort due to a double RERUNN, ACKN and DTCN will both go low concurrent with CSN and DTACKN for one CLK cycle.

To prevent the possibility that the device may misinterpret these signals, it is suggested that these signals be ANDed with CSN (see figure below).



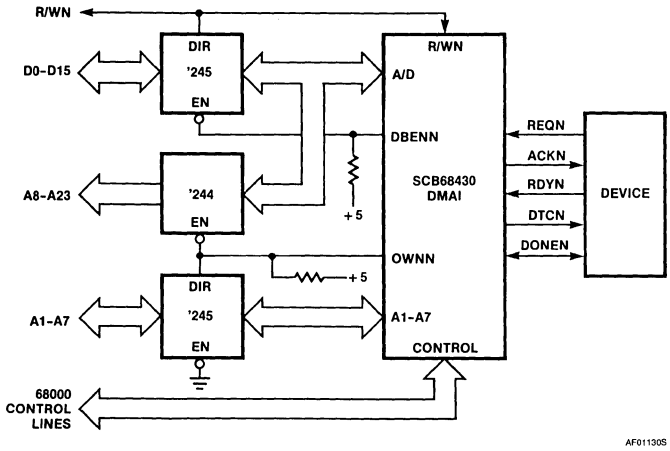


Figure 3. DMAI Application

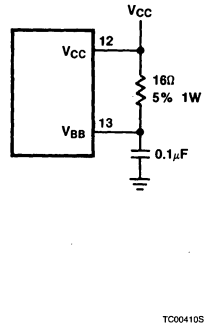


Figure 4. Recommended V_{BB} Test Circuit

2

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltages V_{CC} and V_{BB}	−0.5 to +7.0	V
Input voltage	−0.5 to +5.5	V
Operating temperature range[2]	0 to +70	°C
Storage temperature	−65 to +150	°C

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V ±5%, V_{BB} = Figure 4, T_A = 0°C to +70°C[3,4,7]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
V_{OL}	Output low voltage	I_{OUT} = 5.3mA		0.5	V
V_{OH}	Output high voltage, all outputs except open collector outputs[5]	I_{OUT} = −400μA	2.5		V
I_{IL}	Input low current	V_{IN} = 0.4V		−400	μA
I_{IH}	Input high current	V_{IN} = 2.7V		20	μA
I_{OC}	Open collector off state current[5]	V_{OUT} = 2.4V		20	μA
I_{SC}	Output short circuit current[6]	V_{CC} = max	−40	−100	mA
I_{CC}	V_{CC} supply current	V_{CC} = max		130	mA
I_{BB}	V_{BB} supply current			275	mA

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other conditions other than those indicated in the Electrical Characteristics section of this data sheet is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. Parameters are valid over specified temperature range.
4. All voltage measurements are referenced to ground (V_{SS}). For testing, all signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
5. IRQN, BRN, DONEN, and OWNN are open collector outputs.
6. No more than one output should be connected to ground at one time.
7. Capacitive test load is 100pF for all pins except DTCN which has a 35pF capacitive test load.

AC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V ±5%, V_{BB} = Figure 4, T_A = 0°C to +70°C[3,4,7]

NO.	FIGURE	CHARACTERISTICS	TENTATIVE LIMITS				UNIT
			10MHz		12.5MHz		
			Min	Max	Min	Max	
1	5	A1 – A7, ASN, RWN, set-up to UDSN, LDSN low	0		0		ns
2	5	D0 – D15 3-state to invalid data from ASN, CSN, and UDSN or LDSN low	10		10		ns
3	5	DTACKN 3-state to high from ASN, CSN, and UDSN or LDSN low	10		10		ns
4	5	CSN low after UDSN or LDSN low		25		25	ns
5	5, 6	DBENN low after ASN and CSN low		60		60	ns
6	5	D0 – D15 valid data from ASN, CSN, and UDSN or LDSN low		100		100	ns
7	5	DTACKN low after D0 – D15 valid data	−15	30	−15	30	ns
8	5	A1 – A7, ASN, RWN or CSN hold after UDSN and LDSN high	0		0		ns
9	5, 6	DBENN high from either ASN or CSN high		45		45	ns
10	5	D0 – D15 to 3-state from UDSN and LDSN high		80		80	ns
11	5	D0 – D15 to invalid data from UDSN and LDSN high	10		10		ns
12	5, 6	DTACKN high from UDSN and LDSN high		55		55	ns
13	5, 6	DTACK 3-state from either CSN or ASN high		85		85	ns
14	6	A1 – A7, ASN, RWN set-up to UDSN, LDSN low	50		50		ns/s
15	6	CSN set-up before UDSN or LDSN low	20		20		ns
16	6	DTACKN 3-state to high after CSN and ASN low	10		10		ns
17	6	D0 – D15 valid after UDSN or LDSN low		0		0	ns
18	6	DTACKN low from UDSN or LDSN low		100		100	ns
19	6	UDSN and LDSN low time	115		100		ns
20	6	A1 – A7 hold after UDSN and LDSN high	0		0		ns
21	6	ASN, RWN and CSN hold after UDSN and LDSN high	0		0		ns

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTICS	TENTATIVE LIMITS				UNIT
			10MHz		12.5MHz		
			Min	Max	Min	Max	
22	6	D0 – D15 hold after UDSN and LDSN high	0		0		ns
23	7	DBENN low from last low of ASN, IACKN, LDSN		65		65	ns
24	7	D0 – D7 valid after last low of ASN, IACKN, LDSN		105		105	ns
25	7	DTACKN 3-state to high after last low of ASN, IACKN, LDSN		100		100	ns
26	7	DTACKN low after last low of ASN, IACKN, LDSN		110		110	ns
27	7	DBENN high after first high of ASN, IACKN, LDSN		55		55	ns
28	7	D0 – D7 hold after first high of ASN, IACKN, LDSN		60		60	ns
29	7	D0 – D7 3-state after first high of ASN, IACKN, LDSN		80		80	ns
30	7	DTACKN high after first high of ASN, IACKN, LDSN		60		60	ns
31	7	DTACKN 3-state after first high of ASN, IACKN, LDSN		95		95	ns
32	8	BRN high from CLK high		65		65	ns
33	8, 11, 12	BGACKN low from CLK low		75		75	ns
34	8, 11, 12	OWNN low from CLK high		75		75	ns
35	8	BGACKN high from CLK low		75		75	ns
36	8,11,12	OWNN high from CLK high (load dependent)		50		50	ns
37	10	REQN set-up before CLK low	30		30		ns
38	10	REQN hold after CLK high	20		20		ns
39	10	BRN low from CLK high		80		80	ns
41	11, 12	ASN, UDSN, LDSN, RWN 3-state to high from CLK low		75		75	ns
43	11, 12	A1 – A23 to valid ASN	0		0		ns
44	11, 12	ASN low from CLK high		65		65	ns
45	11, 12	LDSN, UDSN low from CLK high		90		90	ns
46	11, 12	ACKN low from CLK high		65		65	ns
47	11, 12	DTACKN set-up to CLK high	30		30		ns
48	11, 12	RDYN set-up to CLK low	30		30		ns
49	11, 12	DTCN low from CLK high		70		70	ns
50	11, 12	ASN high from CLK high		75		75	ns
51	11, 12	LDSN, UDSN, high from CLK high		90		90	ns
52	11, 12	DTACKN, RDYN hold after CLK high	0		0		ns
–	11, 12	ASN, LDSN, UDSN, high from DTCN low	–20		–20		ns
53	11, 12	ACKN high from CLK high		50		50	ns
54	11, 12	DTCN high from CLK high		50		50	ns
55	11, 12	Address valid after CLK low	10		10		ns
–	11, 12	Address valid after ASN high	0		10		ns
56	11, 12	DONEN (output) low from CLK low		120		120	ns
57	11, 12	DONEN (output) high from CLK high		50		50	ns
58	11, 12	DONEN (input) set-up low before CLK low	30		30		ns
59	11, 12	DONEN (input) hold low after CLK high	0		0		ns
60	11, 12	BGACKN, ASN, UDSN, LDSN, RWN to 3-state from CLK low		75		75	ns
62	11, 12	A1 – A23 valid to 3-state from CLK high		100		100	ns
63	12	R/WN low from CLK high		65		65	ns
64	12	R/WN high from CLK high		75		75	ns
65	13	RERUNN set-up low before CLK high	30		30		ns
66	13	RERUNN hold low from CLK high	20		20		ns
67	13	A1 – A23 to idle state from CLK low		100		100	ns
68	13	A1 – A23 to valid after CLK low		85		85	ns

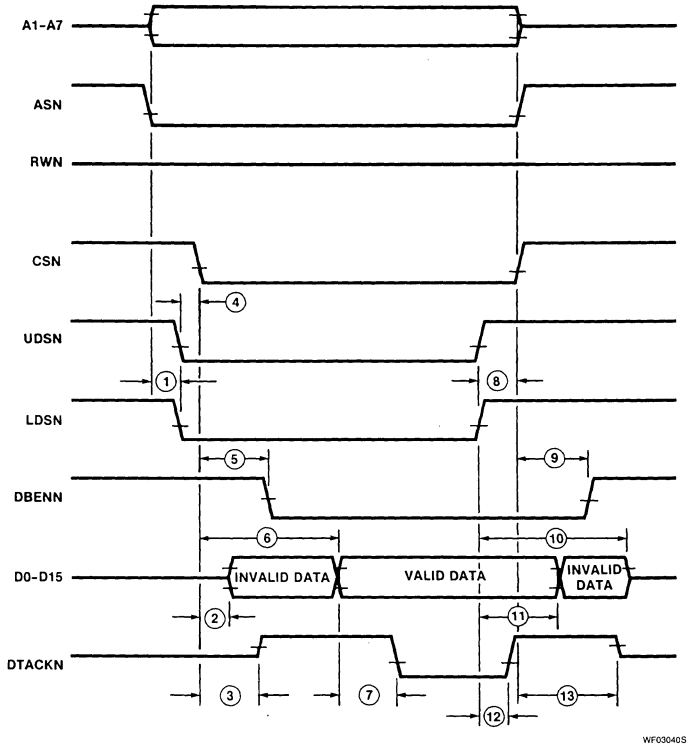


Figure 5. DMAI Read Timing

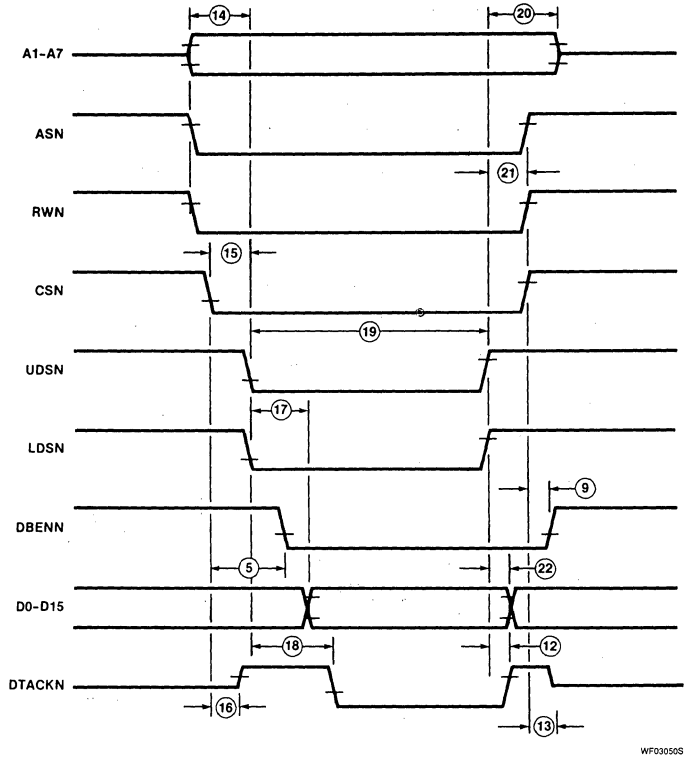


Figure 6. DMAI Write Timing

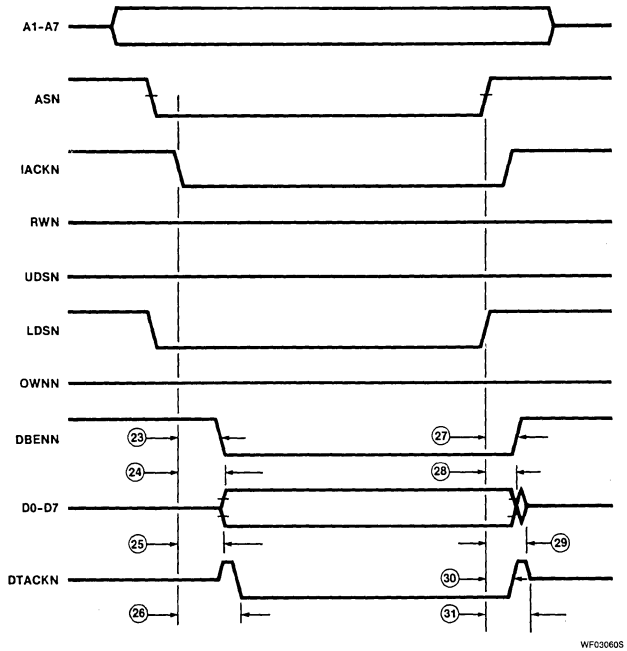
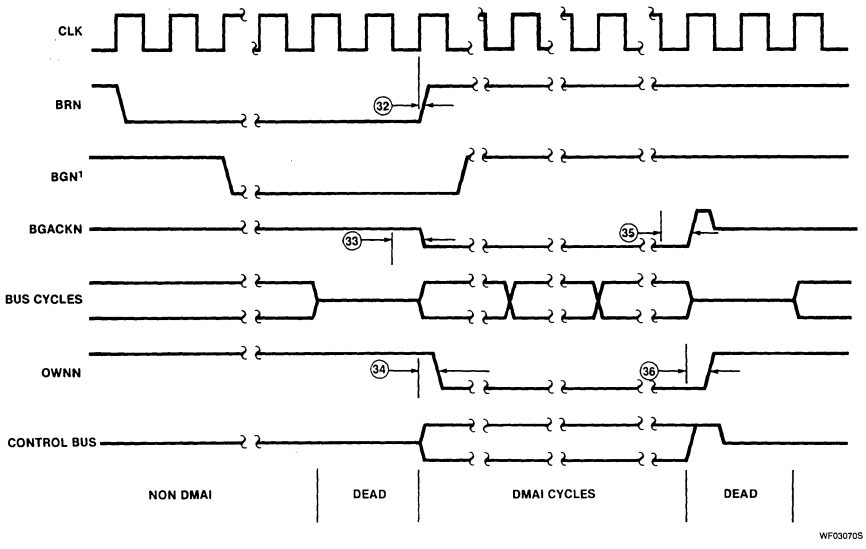


Figure 7. CPU IACK Cycle to DMAI



NOTES:
1. Device will become master if BGN is asserted concurrent with or later than REQN (same clock edge or later).
2. ASN, DTACKN and BGACKN must be negated in order for DMAI to become master. Timing assumes all these happen concurrent with BGN — if not, it is from the latest signal which is negated.

Figure 8. DMAI Bus Arbitration Timing

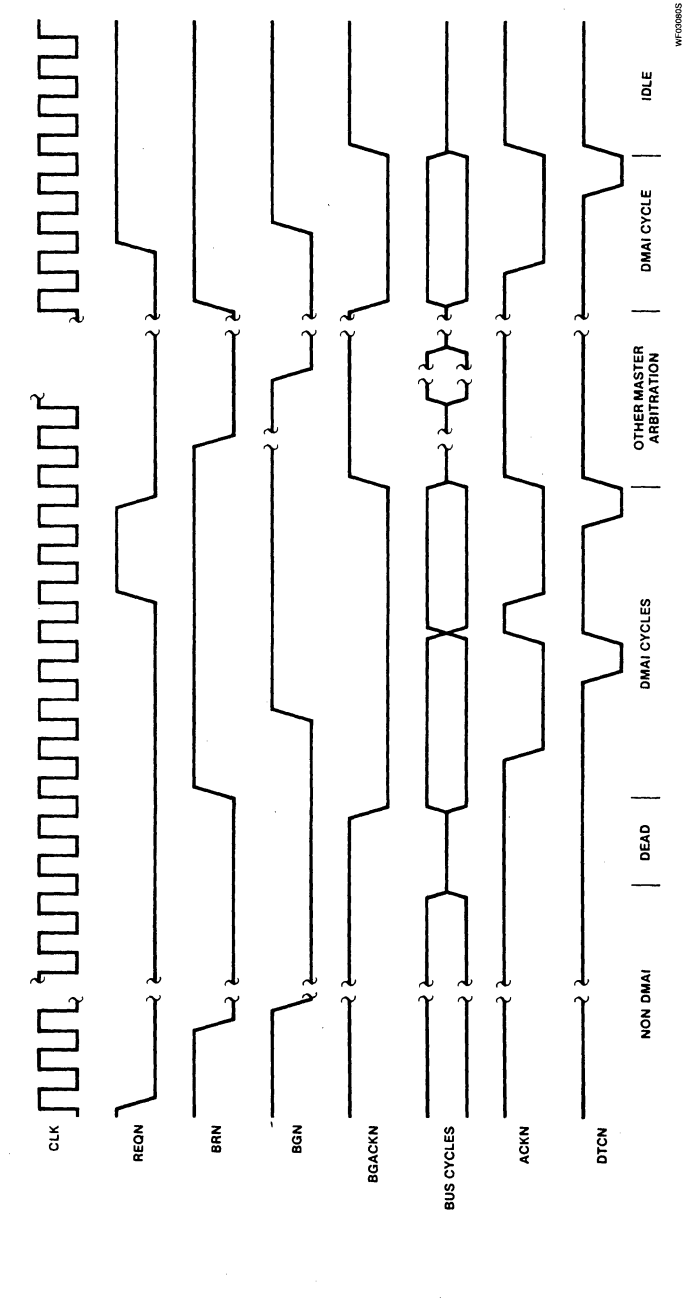
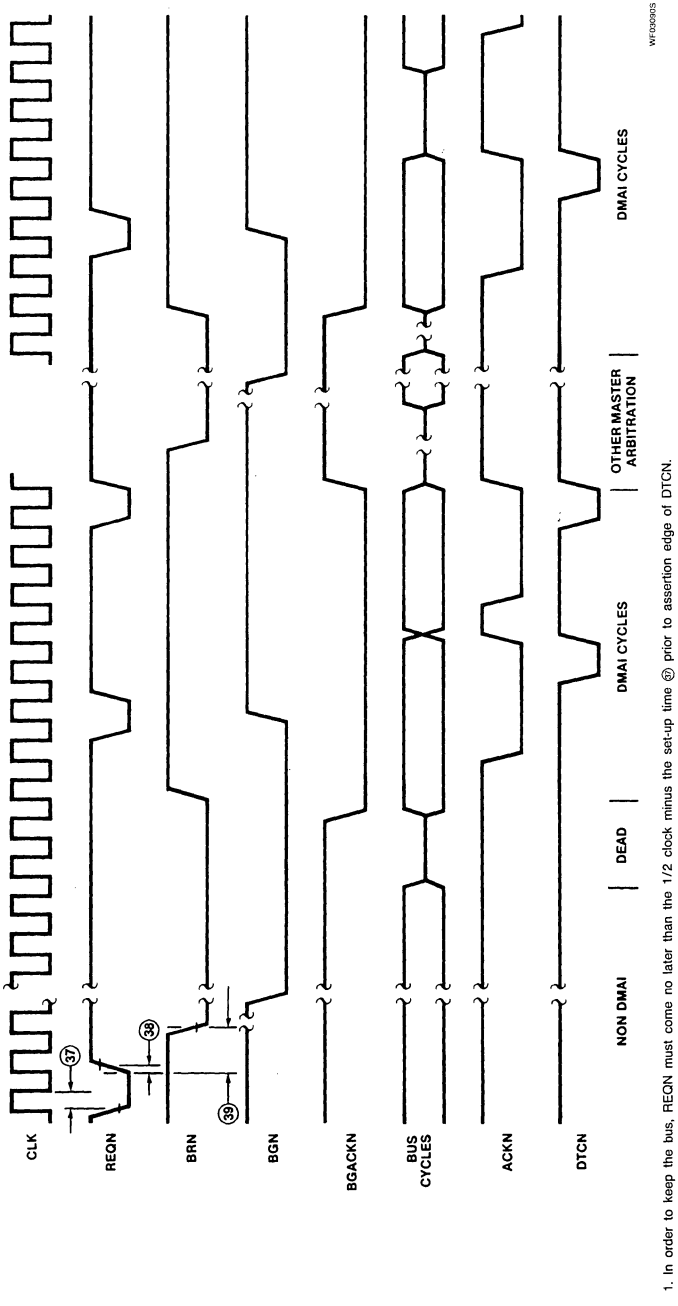


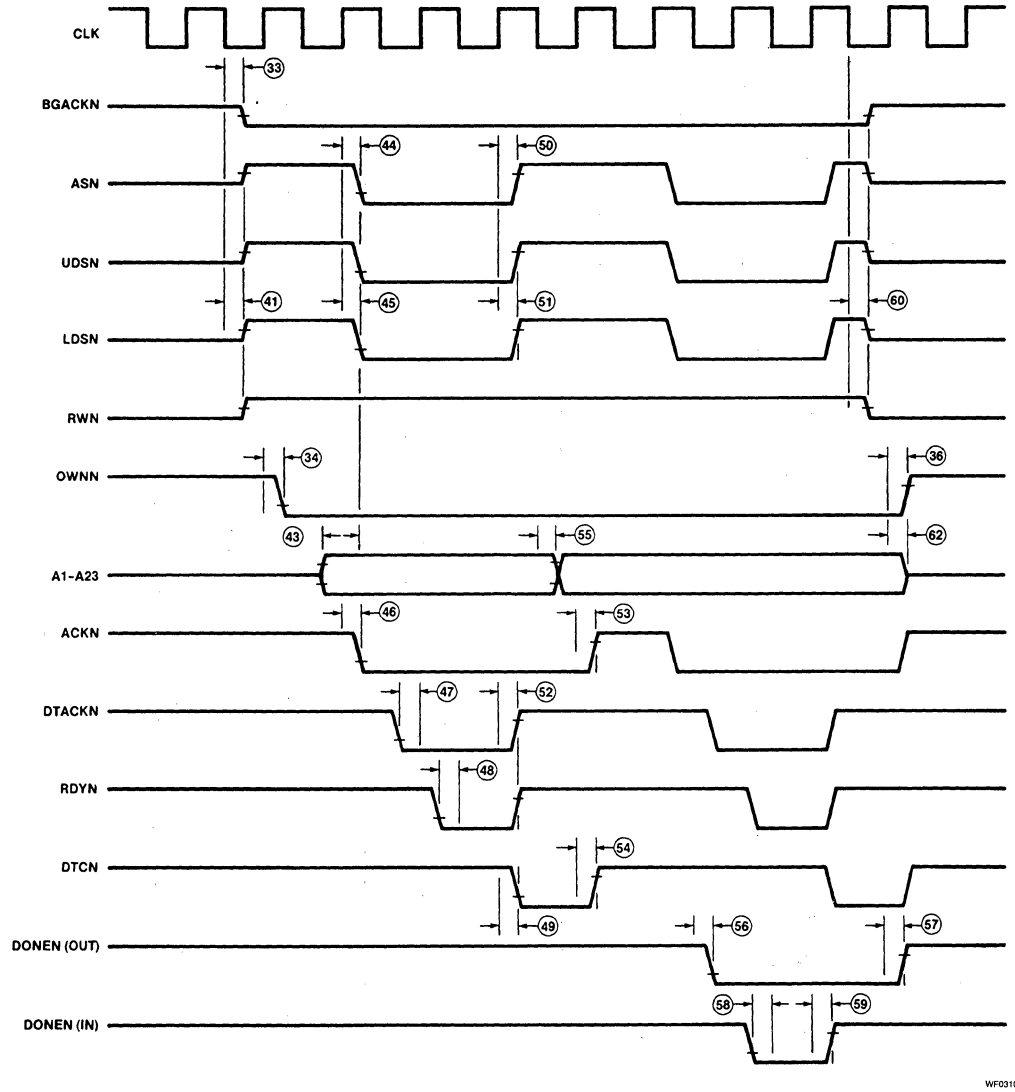
Figure 9. DMAI Burst Mode Request Timing

NOTES:
1. To maintain mastership next bus cycle, REQN must remain asserted at least until 1/2 clock cycle after DTCN is asserted.
2. To guarantee that mastership is relinquished next cycle, REQN must be negated no later than 1/2 clock cycle prior to DTCN.



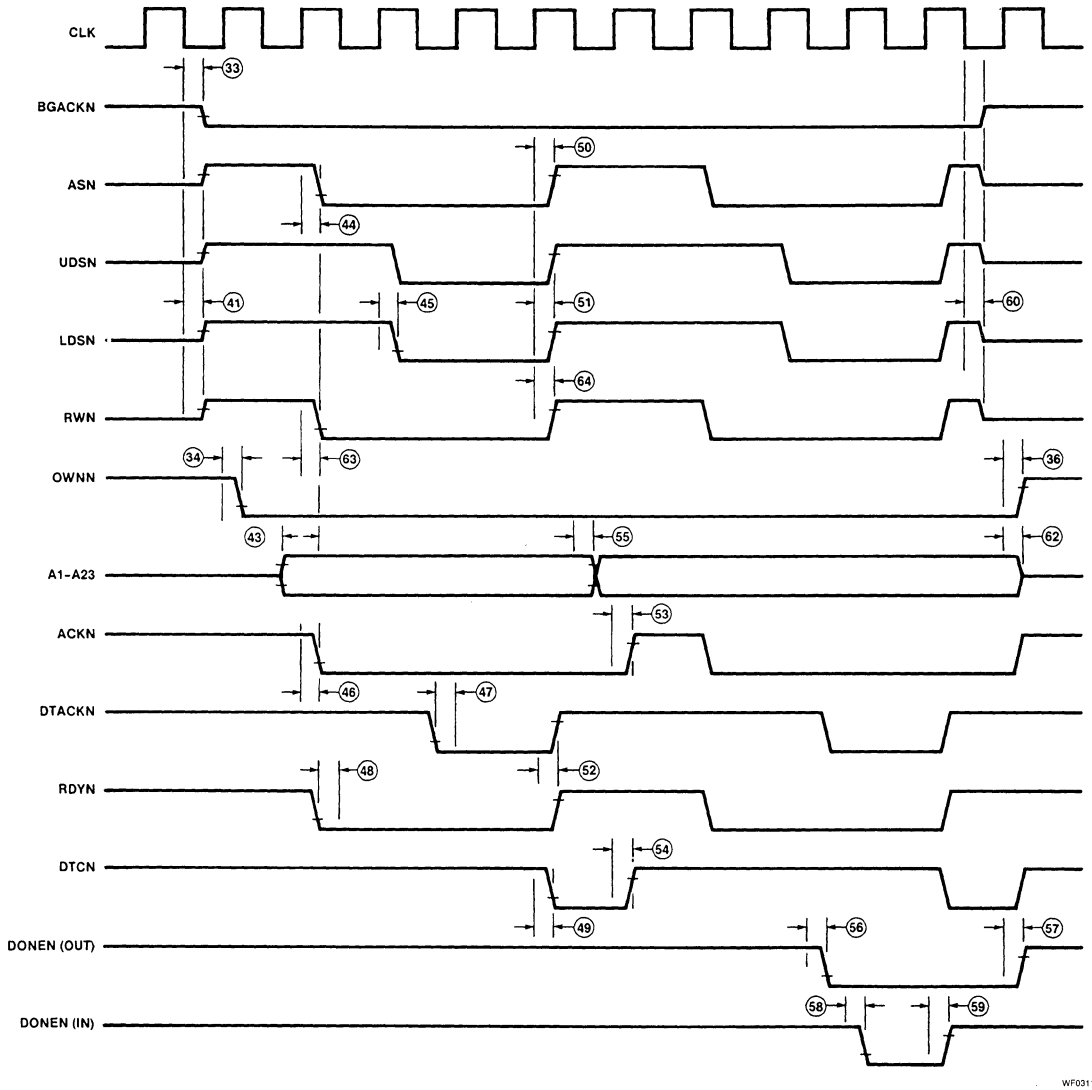
1. In order to keep the bus, REQN must come no later than the 1/2 clock minus the set-up time (37) prior to assertion edge of DTCN.

Figure 10. DMAI Cycle Steal Mode Request Timing



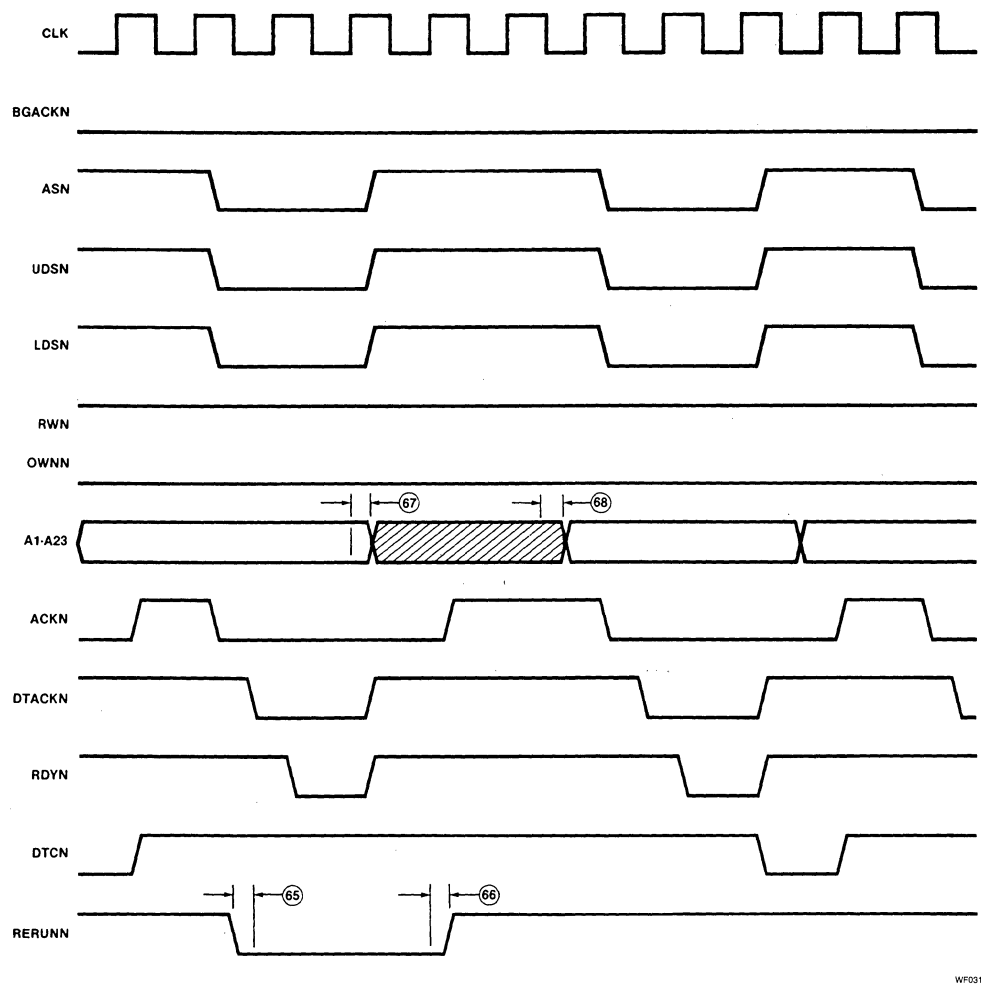
NOTE:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.

Figure 11. Read from Memory, Write to Device



NOTE:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.

Figure 12. Write to Memory, Read from Device



NOTES:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.
2. DMAI will release the bus after a RERUNN if there is no valid request. The next request will then retry the cycle which was terminated by the RERUNN signal.
3. RERUNN must be asserted no later than DTACKN and RDYN.
4. If a cycle is terminated by RERUNN, the transfer count will be one less than the actual data transferred correctly. The double RERUNN signal on the same cycle will terminate the DMAI operation with a status bit set and an interrupt generated (if enabled).

Figure 13. Rerun Asserted During Read from Memory, Write to Device

SCN68454
Intelligent Multiple Disk Controller (IMDC)

Preliminary Specification

Microprocessor Products

DESCRIPTION

The SCN68454 Intelligent Multiple Disk Controller (IMDC) provides the traditional and advanced features required to control Winchester type rigid disks and floppy disks. It can control up to four rigid or floppy disk drives, in any combination. It can be used to control, with a minimum of external hardware, any drive that has an SA1000 interface standard or an ST506 Seagate interface standard.

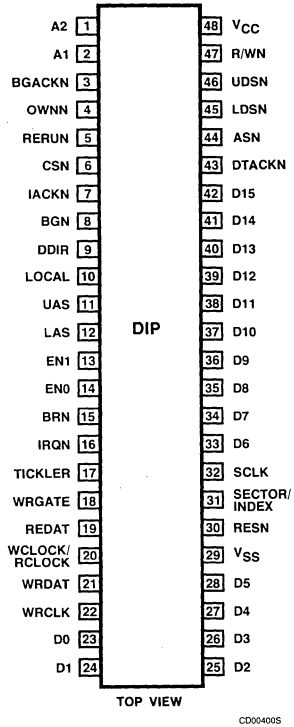
The IMDC supports soft or hard sectored disk track format and standard IBM track formats for floppy disks (both single and double density).

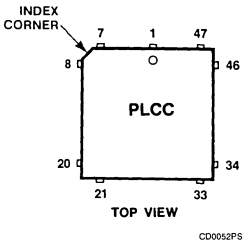
The controller is programmable via an external host processor by the use of high level commands. Data transfer on the host data bus can be 8 or 16-bit in parallel. The IMDC is capable of handling a serial data rate of up to 10Mbits per second. The SCN68454 is constructed using Signetics MOS-VLSI technology.

FEATURES

- **Bus compatible with SCN68000 microprocessor**
- **Automatic rerun on bus error**
- **Supports SCN68000 vectored interrupts**
- **31-bit address counter**
- **16/8-bit data transfers**
- **Supports up to 4 rigid disks and floppy disks in any combination**
- **Supports SA1000 and ST506 Winchester interfaces**
- **Data rates up to 10Mbits per second for MFM**
- **Data rates up to 2Mbits per second for FM**
- **Handles FM and MFM data encoding/decoding**
- **On chip DMA controller and FIFO buffer (128 bytes)**
- **Multiple sector read/write with implied seek**
- **Automatic bad sector handling**
- **32 and 40-bit ECC programmable polynomials**
- **Supports computer generated ECC polynomials**

PIN CONFIGURATION





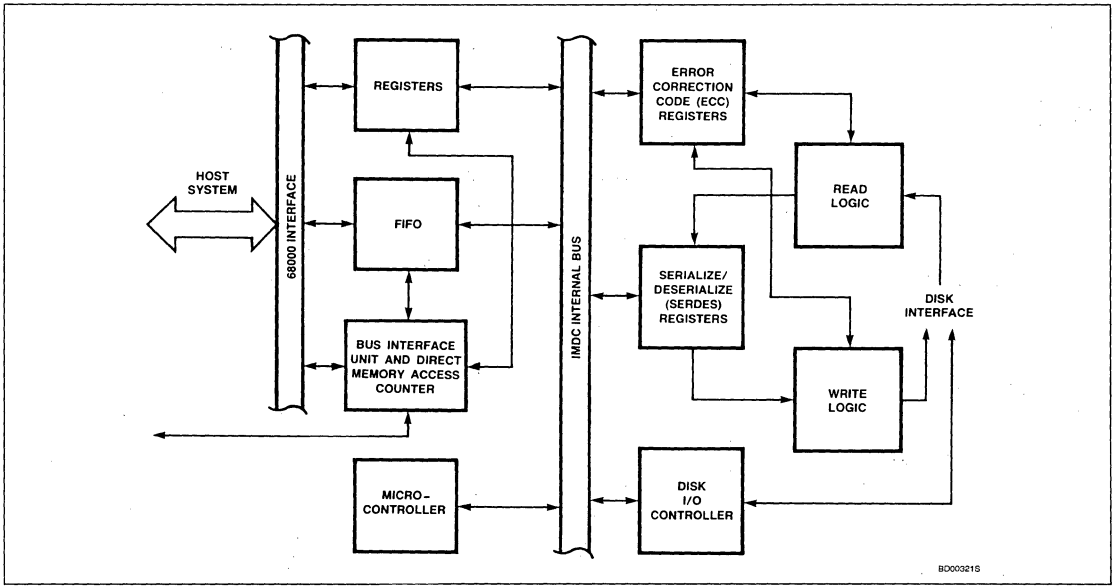
Pin	Function	Pin	Function
1	NC	27	NC
2	A2	28	D2
3	A1	29	D3
4	BGACKN	30	D4
5	OWNN	31	D5
6	RERUNN	32	V_{SS}
7	CSN	33	RESN
8	IACKN	34	SECTOR/INDEX
9	BGN	35	SCLK
10	DDIR	36	D6
11	LOCAL	37	D7
12	UAS	38	D8
13	LAS	39	D9
14	NC	40	NC
15	EN1	41	D10
16	EN0	42	D11
17	BRN	43	D12
18	IRQN	44	D13
19	TICKLER	45	D14
20	WRGATE	46	D15
21	REDAT	47	DTACKN
22	WCLOCK/RCLOCK	48	ASN
23	WRDAT	49	LDSN
24	WRCLK	50	UDSN
25	D0	51	R/WN
26	D1	52	V_{CC}

Intelligent Multiple Disk Controller (IMDC) SCN68454

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0 to 70°C
Ceramic DIP	SCN68454C6I48
Plastic DIP	SCN68454C6N48
Plastic LCC	SCN68454C6A52

BLOCK DIAGRAM



PIN DESCRIPTION

The pin description table describes the function of each of the pins of the IMDC. Signal names ending in 'N' are active low. All other signals are active high. In the descriptions, 'REG mode' refers to the state when the IMDC is chip selected. The term 'DMA mode' refers to the state when the IMDC assumes ownership of the bus. The term 'LOCAL mode' refers to the state when the IMDC is transferring data to or from the disk interface. The IMDC is in the 'IDLE mode' at all other times.

In this data sheet, signals are discussed using the terms 'active' and 'inactive' or 'asserted' and 'negated' independent of whether the signal is active in the high (logic one) state or the low (logic zero) state. Refer to the individual pin descriptions for the definition of the active level of each signal.

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
A1, A2	1, 2	2, 3	I/O	**Address Lines:** Active high, three-statable. In the REG mode, these low order address lines specify which internal register of the IMDC is being accessed. In DMA mode, A1 – A2 are outputs which provide the low order address bits of the location being accessed. Three-stated in IDLE and LOCAL mode.
D0 – D15	23 – 28, 33 – 42	25, 26, 28 – 31, 36 – 39, 41 – 46	I/O	**Data Lines:** Active high, three-statable. In REG mode, the bidirectional data lines are used to transfer data between the CPU and the IMDC registers. In LOCAL mode, the bidirectional data lines are used to transfer data between the IMDC and the disk unit (three-stated in IDLE mode). In DMA mode, the data lines carry the address information. During the first part of the cycle, D0 – D15 provide high order address bits, A19 – A31, which are latched by UAS. The data lines then provide A3 – A18 address bits which are latched by LAS.

2

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
ASN	44	48	I/O	**Address Strobe:** Active low, three-statable. In REG and IDLE modes, ASN is an input which indicates that the current bus master has placed a valid address on the bus. It is monitored by the IMDC during bus arbitration to ascertain that the previous bus master has completed the current bus cycle. In DMA mode, it is an output indicating that the IMDC has placed a valid address on the bus.
UDSN	46	50	I/O	**Upper Data Strobe:** Active low, three-statable. In REG and IDLE modes, UDSN is an input which indicates that the upper data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning. In an eight bit system this input is tied to DSN.
LDSN	45	49	I/O	**Lower Data Strobe:** Active low, three-statable. In REG and IDLE modes, LDSN is an input which indicates that the lower data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning. In an eight bit system this input is tied to A0.
R/WN	47	51	I/O	**Read/Write:** Active high for read, low for write, three-statable. In REG mode, R/WN is an input which controls the direction of data flow through the IMDC's input/output data bus interface and through an external data bus buffer. R/WN high causes the IMDC to place the data from the addressed register on the data bus, while R/WN low causes the IMDC to accept data from the data bus. In DMA mode, R/WN is an output to memory and I/O controllers indicating the type of bus cycle. It is held three-stated during IDLE and LOCAL mode.
CSN	6	7	I	**Chip Select:** Active low. When low, places the IMDC into the REG mode. This input signal is used to select the IMDC for register data transfers. These transfers take place over the D0 – D15 lines as controlled by the R/WN and A1 – A2 inputs. The IMDC is deselected when CSN is high. CSN is ignored during DMA mode.
DTACKN	43	47	I/O	**Data Transfer Acknowledge:** Active low, three-statable. In REG mode, DTACKN is asserted on a write cycle to indicate that the data on the bus has been latched, and on a read cycle or interrupt acknowledge cycle to indicate that valid data is present on the bus. The signal is negated (driven high) when completion of the cycle is indicated by negation of the CSN or IACKN input. In DMA mode, DTACKN is an input monitored by the IMDC to determine when the addressed device (memory) has latched the data (write cycle) or put valid data on the bus (read cycle).
RERUNN	5	6	I	**Rerun:** Active low. This input is asserted by external error detect logic to indicate a bus error. In DMA mode, the IMDC stops operation and three-states the data, address, and control lines, except BGACKN. It remains IDLE until RERUNN becomes inactive, and then retries the last bus cycle. If RERUNN is asserted again, the IMDC sets the error code in the main status byte, stops DMA operation, releases the bus, and interrupts the CPU. Not monitored in REG, LOCAL and IDLE modes.
RESN	30	33	I	**Master Reset:** Active low. Assertion of this pin clears the internal registers and initializes the interrupt vector register to H'0F'. All bidirectional I/O lines are three-stated and the IMDC is placed in the IDLE mode.
SCLK	32	35	I	**Clock:** Active high. Usually the system clock, but may be any clock meeting the electrical specifications. Used by the IMDC to synchronize disk functions and external control lines, and may not be gated off at any time. The frequency should be 16MHz ± 1%.
IRQN	16	18	O	**Interrupt Request:** Active low, open drain. This output is asserted at the end of each command execution. The CPU can read the status register to determine the interrupting condition, or can respond with an interrupt acknowledge cycle to cause the IMDC to output an interrupt vector on the data bus.
IACKN	7	8	I	**Interrupt Acknowledge:** Active low. When asserted, indicates that the current cycle is an interrupt acknowledge cycle. The IMDC normally responds by placing the contents of the interrupt vector register on the data bus and asserting DTACKN.
BRN	15	17	O	**Bus Request:** Active low, open drain. BRN is asserted by the IMDC to request ownership of the bus for a DMA transfer. It is negated when the bus has been granted (BGN low) and BGACKN has been asserted.
BGN	8	9	I	**Bus Grant:** Active low. BGN indicates to the IMDC that it is to be the next bus master. After BGN is asserted, the IMDC waits until DTACKN, ASN, and BGACKN have become inactive before assuming ownership of the bus by asserting BGACKN.
BGACKN	3	4	I/O	**Bus Grant Acknowledge:** Active low, open drain. As an input, BGACKN is monitored by the IMDC during the bus arbitration cycle to determine when it can assume ownership of the bus (BGACKN negated). In DMA mode, it is asserted by the IMDC to indicate that it is the bus master. Three-stated in REG, LOCAL, and IDLE modes.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
EN0	14	16	O	**Enable 0:** Active high. This signal is asserted during LOCAL mode when the IMDC transmits disk control signals on the data lines D0 – D15. The assertion or negation of EN0 is used to latch the control signals at the disk interface.
EN1	13	15	O	**Enable 1:** Active low. This signal is asserted during LOCAL mode when the IMDC reads the status of the disk drive. The assertion of EN1 is used to enable the status information onto data lines D0 – D15.
UAS	11	12	O	**Upper Address Strobe:** Active high. UAS is active only during DMA mode. It is used to latch the upper address bits A19 – A31 from the address/data lines.
LAS	12	13	O	**Lower Address Strobe:** Active high. LAS is active only during DMA mode. It is used to latch the lower address bits A3 – A18 from the address/data lines.
REDAT	19	21	I	**Composite Read Data:** This signal is the composite disk data synchronized to the RCLOCK signal generated by the external PLL.
WRGATE	18	20	O	**Write Gate:** Active high. This signal is asserted during disk write operations. Disasserted during all other modes.
WCLOCK/ RCLOCK	20	22	I	**Write Clock/Read Clock:** This clock input is generated by external logic such as the SCB68459 DPLL. During disk write operations, the input is defined as WCLOCK which provides the bit-cell frequency to the IMDC. The input is defined as RCLOCK during disk read operations. RCLOCK is twice the data frequency and is synchronized to REDAT.
WRDAT	21	23	O	**Write Data Pattern:** Active high. This signal provides the write data pattern for external logic, like the SCB68459 DPLL, to use with WRCLK and WCLOCK to generate the write data pulse stream to a disk unit. WRDAT is an non-return to zero (NRZ) signal which changes state on the rising edge of WCLOCK.
WRCLK	22	24	O	**Write Clock Pattern:** Active high. This signal provides the write clock pattern for external logic, like the SCB68459 DPLL, to use with WRDAT and WCLOCK to generate the write data pulse stream to a disk unit. WRCLK is an NRZ signal which changes state on the rising edge of WCLOCK.
TICKLER	17	19	O	**Tickler:** Active high. This signal is used to control an external PLL. When TICKLER is asserted, the external PLL should output the crystal controlled clock to the WCLOCK/ RCLOCK input. With TICKLER is low, the PLL should synchronize on the incoming disk data and generate read clock to the WCLOCK/RCLOCK input.
SECTOR/ INDEX	31	34	I	**Sector/Index:** Active high. The IMDC uses the rising edge of the signal to generate the read/ write sequence. Hard sectored disk drives output a pulse for each sector boundary for the IMDC. Soft sectored disk drives have a once around index pulse to define the start of a track.
LOCAL	10	11	O	**Local:** Active high. During LOCAL mode, this line controls the output enable on the bidirectional buffers of the address/data lines.
OWNN	4	5	O	**Own:** Active low. This output is asserted by the IMDC during the DMA mode to indicate bus mastership. It can be used to enable external address/data and control buffers. Inactive in REG and IDLE modes.
DDIR	9	10	O	**Data Direction:** Active high. This signal is active during the DMA and REG modes. It controls the direction of the data through the bidirectional buffers on the address/data bus. DDIR is asserted during a read operation of the IMDC.
V_{CC}	48	52	I	**+5 volt ± 5% power input.**
V_{SS}	29	32	I	**Power ground input.**

REGISTERS

Register Map

The IMDC is a memory transfer oriented device with minimal information transferred to the registers. The internal accessible register organization of the IMDC is shown in table 1. Register bit formats are shown in table 2. Each is 8-bits wide to allow interface to either 8 or 16-bit host systems. When the IMDC is interfaced to an eight bit system, A0 is tied to LDSN and the data strobe (DSN) is tied to UDSN.

Interrupt Source Register (ISR)

These bits are used to indicate which drive was the source of a command completion interrupt. Bit 4 reflects drive 0 as the source and bit 7 is for drive 3. The assertion of RESN will initialize all four bits to zero. The IMDC will not initiate the next command, if pending, until the host resets the interrupt source bit. The host should also reset the appropriate busy bit in the status and configuration register.

Drive Status and Configuration Register (DSCR)

[0]8/16 Bit Mode

This bit sets the length of memory and register data transfers. Byte transfers are initiated when bit 0 is set to zero. This bit is set to zero when RESN is asserted.

[7:4]Drive Busy

These bits are used by the host system to initiate an IMDC command operation for a particular drive. Bit 4 is used for drive 0 and bit 7 for drive 3. After the host system has set a busy bit to activate a drive, the IMDC

Table 1. IMDC ADDRESS MAP

ADDRESS BITS[1]	ACRONYM	REGISTER NAME	MODE	AFFECTED BY RESET
2 1 0				
0 0 0	EPH	ECA pointer high	R/W	Yes
0 0 1	EPMH	ECA pointer middle high	R/W	Yes
0 1 0	EPML	ECA pointer middle low	R/W	Yes
0 1 1	EPL	ECA pointer low	R/W	Yes
1 0 0	IVR	Interrupt vector register	R/W	Yes
1 0 1	ISR	Interrupt source register	R/W	Yes
1 1 0[2]	DSCR	Drive status and configuration register	R/W	Yes
1 1 1		Reserved		

NOTES:
1. A0 = 0 for UDSN asserted, A0 = 1 for LDSN asserted for 16- bit mode.
2. In 16-bit systems, the data for this register must be in data bits D0 – D7.

Table 2. REGISTER BIT FORMATS

INTERRUPT SOURCE REGISTER

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DR3 INT	DR2 INT	DR1 INT	DR0 INT	<——— NOT USED ———>			

DRIVE STATUS AND CONFIGURATION REGISTER

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DR3 BUSY	DR2 BUSY	DR1 BUSY	DR0 BUSY	<——— NOT USED ———>			8/16 MODE

responds by accessing the corresponding event control area (ECA) and performing the requested action. The host system can abort the current drive operation by resetting the busy bit before the operation is completed. The assertion of RESN will initialize the busy bits to zero (only time the IMDC writes into this register).

Event Control Area Pointer Registers (EPH/EPMH/EPML/EPL)

These four registers are used by the host system to direct the IMDC to a table of pointers. The table consists of four 2 word addresses that point to the location of the four ECA blocks. The pointers are arranged in ascending order by drive number. EPL is the least significant byte and EPH the most significant byte. All four registers are cleared to zero by RESN asserted.

Interrupt Vector Register (IVR)

The IVR contains the value to be placed on the data bus upon receipt of an interrupt acknowledge from the CPU. The contents of this register are initialized to H'0F' by a reset.

OPERATION

The IMDC is an intelligent controller with on-chip microprogrammed CPU and interface circuitry. Two interfaces are provided, one to the host CPU and the other to the disk drive. The host interface contains a complete DMA interface compatible with the SCN68000 bus. After the IMDC accepts a command, the DMA interface handles all transfers between the system memory and the IMDC. The IMDC signifies through an interrupt signal that the command is completed.

Operation Initiation

The host must initialize the IMDC before it can process any command requests. The initialization information must be passed from the host to the IMDC control registers. The information consists of the data byte or word transfer mode selected in the drive status and configuration register. The host must also load an interrupt vector to the interrupt vector register and load the event control areas registers with the start location of the ECA pointer table in system memory.

A new command is requested using the drive status and configuration register. The user sets a bit corresponding to the drive to be serviced. This causes the IMDC to start execution of the command for the requested drive. This can be performed as long as the bus is available. The IMDC accepts the request but does not necessarily begin to process it. If the IMDC is currently doing disk data transfers to another disk drive, it will accept the request and treat it as a pending request for disk drive service. It does this in order to prevent forcing the host processor to wait for it to complete its current processing.

The IMDC will execute parallel seek operations for floppy and SA1000 type disk drives.

Because four different drives can be on line simultaneously, the possibility exists of more than one pending request to appear in the drive status and configuration register. In this situation the IMDC will process the drive requests in ascending order (i.e., 0, 1, 2, and 3) and then start again with drive zero.

It is also the responsibility of the host to insure the drives that are to be controlled are in a ready state prior to issuing commands. Commands issued to drives that are not ready will result in the command being aborted. Also if a command is requested on a drive that goes from ready to not ready, the IMDC will abort the execution of the command. In both cases, an interrupt cycle will be initiated.

When RESN is asserted, the IMDC will go through an internal initialization program which clears the control registers and the interrupt vector will be set to H'0F'. All bus and control lines will be cleared and the IMDC will enter an idle loop.

DMA Operation

After a command is started, it is executed without further communication with the host system. All memory data transfers are handled automatically by the on chip DMA controller. The IMDC indicates that it wishes to become the bus master by asserting its bus request (BRN) output. The processor ac-

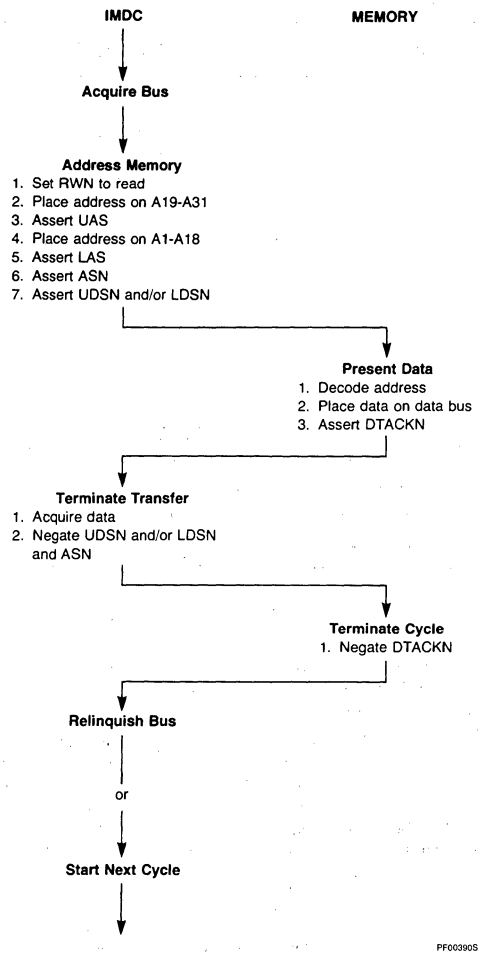


Figure 1. Transfer from Memory to IMDC

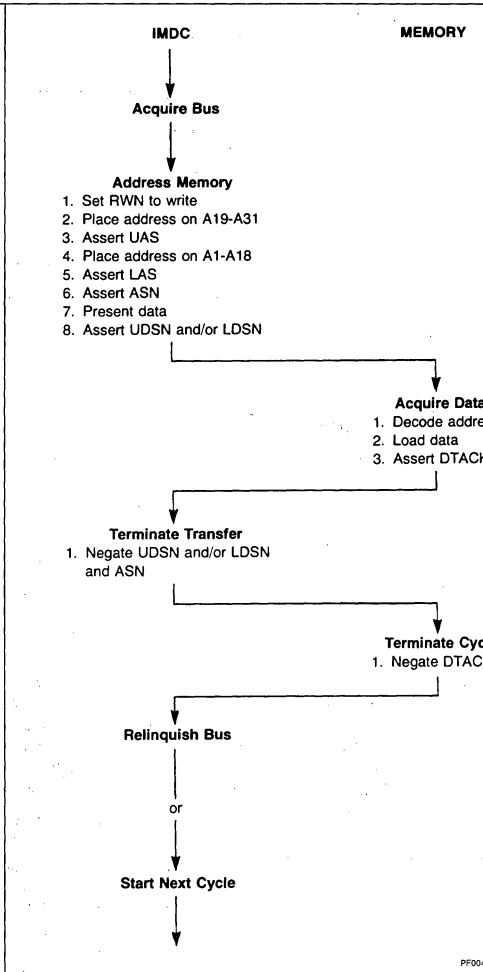


Figure 2. Transfer from IMDC to Memory

knowledges the request by asserting its bus grant (BGN) output which puts the bus up for arbitration.

The IMDC will be the next bus master when its BGN input is asserted. The IMDC then waits for address strobe (ASN), data transfer acknowledge (DTACKN), and bus grant acknowledge (BGACKN) to become inactive. It then asserts the BGACKN output to become the bus master and negates the BRN output. The IMDC then proceeds with the transfer of data between itself and memory. After the data transfer phase, the IMDC relinquishes bus mastership by negating the BGACKN output. Flow charts for the transfer operations are shown in figures 1 and 2. Refer to the timing section for the equivalent timing diagrams.

DMA Transfer Rates

The serial read/write I/O portion of the drive interface is a dedicated slave of the IMDC disk drive interface; that is, a serial string is sent or received by the interface without interruption. The DMA interface, however, must surrender the bus for arbitration after the requested number of operands have been transferred. If the number of transfers does not provide the IMDC with sufficient bus access time, it is conceivable that the DMA may fail to keep track with the disk I/O operation. This situation can occur if the bus is lost for a 'long' period of time. At the IMDC, the serial disk transfer rate is 1.6μsec per word at a 10MHz data rate. The IMDC processor operates using a 312.5nsec period clock. Three cycles or 937.5nsec are required to perform one DMA transfer. If the system bus is unavailable to the IMDC DMA for an average time greater than the difference between the drive and DMA transfer rates (662.5nsec), overflow or underflow of the FIFO buffer can result.

As an example, consider a Winchester disk transfer data of 5Mbits/second. If the transfer count is set to 16 words per transfer, the IMDC will transfer each 32 byte block in 15 microseconds. The transfer time will be 51.2 microseconds for the disk data to put 32

bytes into the FIFO. The difference between the two transfer block rates is 36.2 microseconds. This is the maximum time which the IMDC does not require the system bus.

To avoid this situation the DMA must not lose the bus for a period exceeding the difference in transfer rates. This can be accomplished in a variety of ways. One approach is to allow the DMA to have the bus mastership for the entire sector transfer. A second approach is delay the DMA until a sector has been read into the FIFO buffer and to delay writing to the disk until the FIFO is filled with one sector of data. The first approach severely restricts the availability of the system bus to other devices.

The latter approach requires a minimum FIFO buffer equal to the size of a sector and introduces a delay into the IMDC operation.

As a result, both approaches place stringent limitations on the system and reduce the throughput of the device.

Another alternative is to assign the bus arbitration such that the IMDC always has highest priority and that the bus must be returned in a time segment shorter than the transfer rate difference. This latter solution requires external circuitry to implement and still does not guarantee the IMDC bus access time is sufficient. To provide the user with maximum system flexibility, without severely restricting the bus to other devices, the IMDC has been structured to allow the user to specify the number of operands that the DMA can be master of the bus. The ECA Command Option field permits the user to select the number of operands that can be transferred before the bus is returned for arbitration. This technique allows the user to customize the DMA operation to the needs of the system.

I/O DESCRIPTION

The disk interface consists of two sections. The input and output ports that sense and generate slow changing or static control signals and the serial read/write data section. The parallel data is transferred over the address/ address/data lines (A3/A19/ D0 – A18/A31/D15) during LOCAL mode. Table 3 shows the data bus assignment during an input port bus cycle and table 4 shows the output port bus cycle assignment.

Table 3. INPUT PORT DEFINITION

BUS LINE	SIGNAL NAME	DEFINITION
D0	INDEX	Rigid disk index signal (hard sector mode) must be greater than 500ns
D1	READY	Drive ready
D2	SEEKC	Seek completed
D3	TR0A	Track zero first signal
D4	TR0B	Track zero second signal
D5	WFA	Write protection for floppies
D6	WFB	Write fault
D7 – D15		Not used

Table 4. OUTPUT PORT DEFINITION

BUS LINE	SIGNAL NAME	DEFINITION
D0	HEAD1	Head select 2**0/side select for floppy disks set to '0' for single sided floppy disks
D1	HEAD2	Head select 2**1
D2	HEAD4	Head select 2**2 that can be transferred
D3	HEAD8	Head select 2**3
D4	HEAD16	Head select 2**4
D5		Not used
D6	STEP	Step pulse 10μsec long (period set in ECA)
D7	DIR	Direction of head movement (a high corresponds to head movement toward the disk spindle)
D8	LWC	Low write current
D9	MOT	Motor on
D10	PRECOM	Precompensation enable
D11	HDL	Head load for floppy disks
D12	SEL0	Drive 0 select
D13	SEL1	Drive 1 select
D14	SEL2	Drive 2 select
D15	SEL3	Drive 3 select

The two signals enable 0 (EN0) and enable 1 (EN1) are used to gate the signals off or onto the bus. EN0 is used to latch the bus during the output cycle and EN1 is used to enable the input data onto the bus.

Input Port

The input port consists of eight lines to input disk status information to the IMDC. All signals are active high. To accommodate different drives, signals TROA, TROB, WFA and WFB are used. The IMDC will respond to any one of the signals which becomes active high. The drive will not activate these signal simultaneously.

Output Port

The output port consists of the signals corresponding to bus lines D0 through D15. All signals are positive logic. A practical implementation of the output port would consist of standard octal registers.

EVENT CONTROL AREAS

The host system communicates with the IMDC through the event control areas (ECAs) which reside in system memory. An ECA parameter block is set up for each disk drive (up to four) to be controlled. These areas contain information that is required by the IMDC to execute a disk command. This information includes data about the requested command and the disk drive. The host prepares for IMDC operation by loading the ECA pointer table in system memory with the address of each of the ECA blocks. The disk drive to ECA block assignment is determined by the relative position of the pointer in the table. The first table entry corresponds to drive zero, the second entry to drive one, etc. (see figure 3).

The IMDC microprogram will use the data contained in the ECA block to generate the disk interface signals and perform the requested drive I/O. Prior to informing the host of a command completion, the

IMDC writes the return status information to the appropriate ECA block memory. Table 5 describes the format of the ECA block.

The communications between the IMDC and the host are established through the ECA. Alteration of the ECA by the host after a command has been accepted by the IMDC is not allowed.

The static and dynamic (see ECA fields) ECA command parameters should be verified by the host before new command execution is requested. During execution of a command involving multiple sectors, it is possible that dynamic values generated by the IMDC could be invalid. The host does not have access during this time to alter or even check these values, therefore, it is the responsibility of the

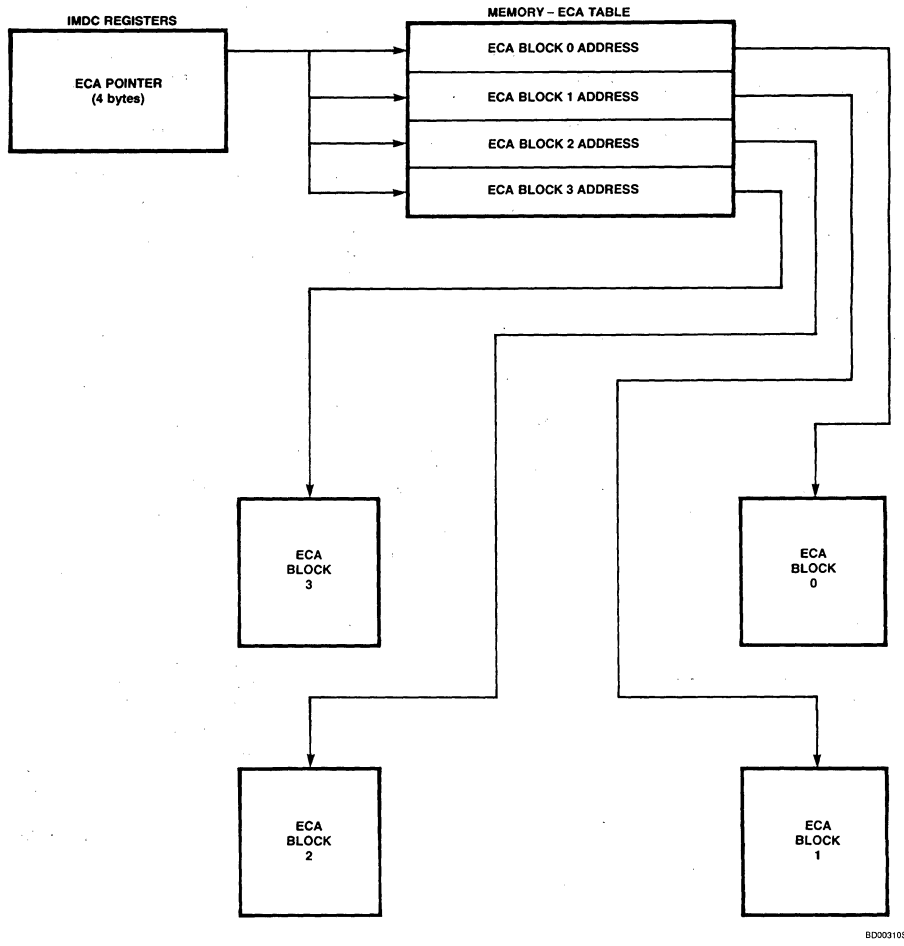


Figure 3. ECA Block Addressing

host to not only guarantee the integrity of the initial parameters but also to insure that values calculated by the IMDC during the life of a command remain valid. For example, the IMDC cannot know that a calculated DMA address is valid until it attempts to transfer data.

Implied in the execution of any disk operation is reading of the ECA data by the IMDC to load the drive control parameters. At the conclusion of a disk operation, the IMDC performs a write operation to the corresponding ECA parameters to store the results of the command. Both reading and writing of the ECA fields by the IMDC utilize the host/DMA interface to arbitrate for the system bus and to perform the required handshaking.

ECA FIELDS

As an IMDC command proceeds through its execution; it references and alters various fields of the ECA. To gain a better understanding of the interactions between the IMDC and the various ECA fields, the ECA data can be separated into four general categories.

1. Command, command status, and execution parameters
2. Programmable record processing fields
3. Disk Format Fields
4. Drive Control Parameters

These categories differentiate the ECA fields not only in content but in terms of the access and alterability.

Command, Command Status and Execution Parameters

These parameters consist of the ECA data fields required by the IMDC to execute a command. The fields can be further separated into static fields and dynamic fields. The static fields are valid for the duration of a command; the IMDC does not alter any information in these fields as long as the command is being processed. The dynamic fields provide the 'local storage' needed to execute multiple sector commands. The IMDC uses the dynamic fields of the ECA to maintain the current execution status of the on-line disk drives.

Table 5. ECA BLOCK FORMAT

WORD NO.	15 8	7 0	
00	Command code	Main status	
01	Extended status		
02	Max # of retries	Actual # of retries	
03	DMA count	Command options	
04	Buffer address most significant word		
05	Buffer address least significant word		
06	Buffer length request		
07	# of bytes transferred		
08	Cylinder number		*
09	Head number	Sector number	*
10	Current cylinder position		
11	PRP command control word		**
12	Location of SCWT, most significant word		**
13	Location of SCWT, least significant word		**
14	Scan terminator	Reserved	**
15	Maximum record length – 1		**
16	N0 – Pre index gap	N1 – Post index gap	***
17	N2 – Sync byte count	N3 – Post ID gap	***
18	N4 – Post data gap	N5 – Address mark cnt	***
19	Reserved		***
20 – 22	ECC mask (3 words)		
23	Motor on delay	# of heads	
24	Ending sector #	Stepping rate	
25	Head setting time	Head load time	
26	Seek type	Phase count	
27	Low write Current boundary track		
28	Precompensation boundary track		
29 – 31	ECC remainder (3 words)		****
32	Maximum number of cylinders per surface		
33	First sector/sector length	Flag byte	
34 – 35	B – tree pointer (2 words)		**
36 – 45	IMDC working area 10 words		

* Physical starting sector number.
** Programmable record processing parameters.
*** Track format fields.
**** ECC remainder will be aligned to the MSB byte of this field.

Command Code (1 Byte)
The command code indicates the command to be executed. Table 6 lists the valid IMDC commands and their associated codes. For a complete description of each command refer to the command description in the Command section.

Main Status (1 Byte)
This field contains the general oriented status information about the command execution. This encoded byte is updated at the completion of the current command and before the interrupt is asserted. Table 7 shows the possible values which may be returned by the IMDC.

Extended Status (2 Bytes)
This field contains specific bit oriented status information about the command execution. If errors result during the execution, the corresponding bit will be ORed into extended status and kept there. This OR function can be used by the host system for error logging. The extended status will be reset to zero by

Table 6. COMMAND CODES

COMMAND MNEMONIC	HEX CODE	DESCRIPTION
WRMS	00	Write multiple sector
WRDD	01	Write with deleted data flag
VER	10	Verify
REMS	11	Read multiple sector
PRP	12	Programmable record processing
TSR	13	Transparent sector read
RETD	20	Read identifier
FORM	40	Format
CALB	41	Recalibrate to track zero
CORR	81	Correct data
DIAG	80	Diagnostic

Table 7. MAIN STATUS CODES

DECIMAL VALUE	DESCRIPTION
0	Correct execution without error
1	Irrecoverable error which cannot be completed (auto retries are attempted, see extended status)
2	Drive not ready
3	PRP operation unsuccessful
4 – 5	Not used
6	Command rejected
7 – 9	Not used
10	Command abort (busy bit reset by host)
11 – 255	Not used

Table 8. EXTENDED STATUS

DATA BIT	DESCRIPTION
0	Write fault
1	CRC/ECC error on data or ID
2	FIFO overrun/underrun
3	No identifier found
4	Not used
5	Deleted data address mark
6	Write on write protected diskette
7	Positioning error
8	Data part timeout
9	Not used
10	Uncorrectable data error (ECC)
11	Not used
12	Not used
13	Positioning timeout
14	Not used
15	Bus Error Fault (DMA operation)

the IMDC at the beginning of the command execution. The error definitions are shown in table 8.

Maximum Number of Retries (1 Byte)
This parameter specifies the maximum number of retries per command (not per sector) that the IMDC attempts, after a disk operation error. Loading a zero value indicates that no retries should be attempted. This byte will not be used if a bus error or FIFO under/overrun error occurs during the command execution. These two errors will cause an immediate command abort. If an identifier is not found, 16 revolutions of the disk will be tried before exit.

Actual Number of Retries (1 Byte)
This byte is set by the IMDC to the actual number of retries executed per command. All retries are accumulated by the IMDC on a sector by sector basis.

DMA Count (1 Byte)
This byte contains a DMA transfer count. If a zero value is specified, the IMDC transfers only one operand and surrenders the bus. A transfer amount of 1 to 16 can be specified.

Command Options (1 Byte)
This byte contains options that are to apply to the current command. Valid options for handling of deleted-data address marks are described in table 9. Sectors with the deleted-data address mark (bits 0, 1 of the option field) will be handled as shown for read operations, excluding those associated with the PRP command. Some retries may be necessary in these steps. They all start with a new identifier search.

Buffer Address
This is a 31-bit starting buffer address for the DMA transfers. The LSB of the address field defines the even or odd address of the 16 bit words when in 16-bit mode. For the 68000, only the first 23 bits are used for addressing. The additional bits can be used to implement the function codes. It should be noted that the IMDC does not protect the upper address lines, AD25 – AD31, from counter overflow. It is up to the operator to prevent transfers between the IMDC and memory greater than the 68000 addressing range. For PRP commands, this field contains the pointer to the content of the matched record data (see Programmable Record Processing section).

Buffer Length Requested (2 Bytes)
This field contains the requested number of bytes to be transferred. This field implicitly defines the number of multiple sector transfers to be performed. A seek only will occur when a zero length is specified. For the format track command, this value is used to terminate the operation.

Table 9. COMMAND OPTIONS

BIT	FORMAT	OPTION DESCRIPTION
1 – 0	00	FM, IBM single density format
	01	MFM, IBM double density format
	10	Programmable disk format, 1 byte cylinder
	11	Programmable disk format, 2 byte cylinder
2		Not used
4 – 3	00	Select CRC-CITT (CRC-16) polynomial
	01	Select 32 bit ECC polynomial
	10	Select 40 bit ECC polynomial
	11	Not used
5	0	The sectors with deleted-data address mark will be skipped as if it did not exist. A successful CRC/ECC check is not required.
	1	The data of the sector with deleted-data address mark will be transferred to the host system and the operation terminated
6		Not used
7		Hard sectored disk

Actual Number of Bytes Transferred (2 Bytes)
This field indicates how many bytes have been actually read or written during a command execution. The value can be used, for example, for a location of a problem sector with an irrecoverable error. The value of this field can be changed by the IMDC during the execution of a command and used as an intermediate field.

Physical Starting Sector Number (4 Bytes)
Cylinder, head, sector – This field contains the physical disk sector location at which the command is to begin execution. The format of the field is shown below:

Cylinder High	Cylinder Low
Head Number	Sector Number

Current Cylinder (2 Bytes)
These two bytes will be updated by the IMDC after each positioning. At the beginning of the IMDC reset, they should be set by the host system to zero, because most of the hard disks have built-in automatic recalibration.

Programmable Record Processing Fields (10 Bytes)

The programmable record processing fields are described in the Programmable Record Processing section.

Disk Track Format Fields (8 Bytes)

The disk track format fields are described in the Disk Track Format section.

ECC Mask (6 Bytes)
These six bytes define the error correction polynomial. The standard notation of a polynomial is: $X^{40} + k_{39} X^{39} + \bullet\bullet\bullet + k_1 X' + k_0 X^0$

The coefficients k_{39} through k_0 may be any combination of 1s and 0s. For a given polynomial, an equal combination of bits in the mask register will initialize the IMDC logic to generate or check the same polynomial on the disk.

Drive Control Parameters

These fields contain information that define the disk interface to the IMDC. This information is provided by the host system and is not altered by the IMDC. The IMDC microprogram, that controls the disk drive I/O issued, continually references this information during the execution of a command. Because of the potentially large number of possible variations of interfaces, the IMDC cannot provide extensive verification of the parameters contained in these fields. Protection and verification of these fields are primarily the responsibility of the a host operating system and not the IMDC.

Motor on Delay (1 Byte)
This field contains the motor on delay (period) in ten millisecond units. A value of zero represents a zero delay timeout.

Number of Heads (1 Byte)
This value is the number of heads on the disk unit. The maximum value is 128.

Ending Sector Number (1 Byte)
This value is the last sector on the track (cylinder) for the disk unit.

Stepping Rate (1 Byte)
This field contains head stepping rate (period) in 500μsec units if applicable.

Head Settling Time (1 Byte)
This field contains head settling time in 500μsec units. If a non-zero head settling time is specified, the IMDC assumes that a seek complete is not available from the drive.

Head Load Time (1 Byte)
This field contains head load time in 500μsec units. For all hard disks, this value must always be set to zero by the host system.

Seek Type (1 Byte)
This field defines what type of seek positioning is to be performed.

VALUE	DESCRIPTION
0	Normal single step seek
1	ST506 with accelerated seek
2	Disk with buffered seek

Phase Counter (1 Byte)
This field contains the phase counter which is a status of the command execution set by the IMDC.

Low Write Current Active Track (2 Bytes)
This field defines, to the IMDC, the track at which low write current (LWC) output signal, on the output port (bit 8), is to be asserted.

Precompensation Active Track (2 Bytes)
This field defines to the IMDC the track at which precompensation signal (PRECOM) is to be asserted at the output port (bit 10).

ECC Remainder (6 Bytes)
This field contains the ECC remainder generated by the IMDC read operation. This returned value will be zero unless an error is detected.

Maximum Number of Cylinders (2 Bytes)
These bytes define, to the IMDC, the maximum cylinder count for the disk unit on line. If an operation is specified beyond this boundary, the IMDC will abort the command.

Sector Length (1 Byte)
This byte defines the sector length as a power of two multiple of 128 bytes, i.e., sector length = $(2^{\#})$ * (128 bytes). The minimum sector length of 128 bytes is specified by a zero value while the maximum length of 4096 is specified by 5. The most significant bit of this byte is used for the IMDC's starting sector number, either 0 or 1. The three least significant bits are used for the sector length.

Flag Byte (1 Byte)
This byte is placed in the ECA by the IMDC during a transparent sector read (TSR) command operation. It is the flag byte for the data field part of the sector.

B-Tree Pointer (4 Bytes)
The IMDC loads the physical sector number of the last record match when in the PRP command mode and in the B-tree scan function (see the Programmable Record Processing section for further details).

IMDC Work Area (20 Bytes)
These ten words are reserved for the IMDC microprogram. The meaning of certain bytes of this area will be determined by the command being executed.

COMMANDS

The following is a generalized description of the sequence performed by the IMDC to execute a command:

a. The IMDC checks if the drive is available. If it is not, the IMDC generates an error status and a completion interrupt.
b. The IMDC executes a track seek if necessary. For drives which have a seek complete signal (as indicated by a zero value in the head settling time field of the ECA), a command termination can be caused by the timeout of the drive signal SEEK COMPLETE. Drives for which the head settling time is non-zero, the IMDC waits the specified time after issuing the stepping pulses (that is, there is no timeout on the seeks).
c. The head will be selected, or for a floppy disk, loaded. The IMDC will wait four byte times to insure head switching has occurred.
d. The IMDC reads the sectcr identifier to locate the requested sector. If the IMDC is unable to perform the sector identifier read, a retry is attempted for as many times as specified in the ECA maximum number of retries parameter. If the retries are all unsuccessful or if no retries are specified, the command is terminated (step g) and the status 'no identifier' is returned. If the matched sector identifier indicates a bad sector (see Track Format description), the IMDC uses the replacement information to perform steps b and c for the new sector. This bad sector replacement is not counted as a retry.
e. After the requested sector has been located, the IMDC performs the sector disk I/O (read or write) and DMA operations. If the operation cannot be completed, the retry processing described in step d is attempted. A failure in this portion of the command is indicated by the return status 'data part time out', which distinguishes it from the read identifier failure of step d.
f. If multiple sectors have been specified in the command, the IMDC assumes contiguous physical sectors: steps b, c, d and e are repeated for each sector. The IMDC automatically performs any track seek required if the next physical sector is located on the next cylinder. When the sectors are interleaved, the IMDC repeats step d until the correct sector is located.
   In the case of a bad sector replacement, the IMDC continues the multiple sector operation, after the replacement sector, at the sector physically located after the bad sector. It performs all the necessary track repositioning required to return the drive head to the next physical sector.
g. Upon command completion, all relevant ECA fields are updated and the appropriate bits of the 'interrupt state status' and 'drive status and configuration' registers are set and an interrupt signal is generated by the IMDC. Upon interrupt acknowledge, the IMDC presents the interrupt vector from the interrupt vector register on the data bus.

Command Description

In the command descriptions, the term FIFO is used to refer to the internal memory of the IMDC when it is used as a FIFO buffer.

Write Multiple Sector with Implied Seek (WRMS)
After the desired sector is located, the IMDC will write the complete data part; preamble, address mark, flag, data, CRC or ECC and postamble. The precise format is given by the drive type. The FIFO will be continuously filled with the new data from the buffer. The number of data bytes written in one sector is determined by the IMDC using information from the ECA fields. This operation is repeated until the number of sectors, implied by the buffer length field of the ECA, have been written.

Write with Deleted Word (WRDD)
This command differs from the normal write only in that the flag or address mark written is the deleted data address mark or flag.

Verify (VER)
Verify with implied seek is the same functionally as the read command, except the IMDC compares the ECC or the CRC for accuracy. No data is transferred to the system.

Read Multiple Sector with Implied Seek (REMS)
After a successful seek to the desired sector, the IMDC will start to read the data of the sector, fill the FIFO buffer and check the CRC or ECC code. This sector data is transferred through the DMA interface to the host system memory. This procedure is repeated until the number of sectors, implied by the buffer length field of the ECA, have been read. For sectors with the deleted data address mark, the IMDC will process them according to the options selected in the command option field of the ECA.

Programmable Record Processing (PRP)
Refer to the Programmable Record Processing section.

Transparent Sector Read (TSR)
This command will read a single error sector and transfer the data to the FIFO buffer regardless of a CRC or ECC check. Only the extended status will be correspondingly filled with all errors encountered. The transparent sector read can be used for diagnostic purposes and, with some manual help, for recovery of damaged data. In case of an incorrect CRC/ECC remainder, the IMDC discards the remainder into four bytes reserved in the ECA for this value.

Read Identifier (RETD)
This command will read identifiers as they come from the disk and fill the whole buffer with records consisting of:
- flag byte
- ID data
- CRC
- remainder generated by IMDC

A CRC error will not terminate the execution of the command. For soft sectored formats, the command execution will begin and end with the detection of an index pulse. For the hard sectored formats, the IMDC will detect and count sector pulses to determine the command termination.

Format a Track with Implied Seek (FORM)
The IMDC supports four different media formats (see Disk Track Format section). This command allows the user to write the format information, as specified by the ECA track format fields, to the recording media.

Recalibrate to Track Zero (CALB)
The function of this command is to retract the heads to track 0. The IMDC issues one step 'IN' and then steps 'OUT' until the signal track 0 becomes active or the maximum number of steps equal the number of cylinders from the ECA. The check on seek completed timeout will be made for the hard disk. The function recalibrate can be initiated automatically by two conditions:
- Encountered error in cylinder number by reading or writing.
- Writing zero into the current cylinder in ECA.

Correct (CORR)
This command provides an error mask which is used to correct the data in the memory. It uses the ECC remainder field of the ECA. The IMDC will put the error correction vector and its relative position from the end of the data buffer into the ECA.

The 24-bit ECC correction mask will be placed in the flag byte and first half of the B-tree pointer of the ECA block. This is the least significant byte of the 33rd word and the 34th word of the ECA block in memory as shown in table 5. The offset count will be placed in the second half of the B-tree pointer or the 35th word of the ECA block. This is the relative offset from the last byte transferred to memo-

ry, plus four for a 32-bit ECC or five for a 40-bit ECC.

Chip Diagnostic (DIAG)
The IMDC will exercise the 16-bit counters; the record counter, the record length counter, the buffer length counter, and the time out counter. If everything is functioning properly, the IMDC places a zero byte into the main status. Otherwise, an irrecoverable error bit value is set in the main status.

DISK TRACK FORMAT

The IMDC supports four different track formats:

- IBM 3740 single density
- IBM System 34 double density
- Programmable soft sectored
- Programmable hard sectored

In each of the formats, data on the physical media is separated into blocks of information or sectors. The format serves several purposes in this arrangement; it defines the structure of the sector, the location of the actual data, and provides gaps and sync bytes that allow an interval for any switching required by the drive hardware. These intervals, in turn, allow the IMDC to compensate for any variations in either the recording media and/or the drive hardware. Although each of these formats is well defined, variations of parameter values within a given format require that the IMDC provide the user with the capability to program them.

Because there is no one standard which defines track format parameters, a description of them and other pertinent definitions are included in this section. See table 10 for track format definitions. Tables 11 and 12 provide a summary of the formats and the programmable values. The following definitions apply:

The format command operation is performed by the IMDC as integral operations on a per track basis, as opposed to normal disk I/O which is on a sector basis. This technique was selected as a compromise to satisfy two conflicting – requirements command efficiency versus equal access of on-line drives to the IMDC resources. Obviously, these commands would be most efficient if allowed to monopolize the IMDC resources. However, this situation would prevent any other drive from being serviced until the command had been completed. Normally these commands are background tasks with the other disk operations having a higher priority. For this reason, allowing the IMDC to concentrate completely on either command is not a good system practice.

The IMDC becomes available for other processing after a full track has been processed. In the worst case situation for the format command, the time the IMDC will not be able to process the other drives will not exceed two disk revolutions. This would occur for disks in which the index pulse was just missed and a complete revolution is required to find the pulse.

The programmable soft sectored disk format, which is used mainly for hard disk drives, is a MFM format which is nearly identical to the IBM double density format. The hard sectored disk format is used for rigid disks with an internal sector clock and is similar to the soft sectored format, except that each sector starts with the sector pulse rather than a related byte count from the index pulse.

The format track command allows the user to write formatting information to the recording media. Specification of format parameters is accomplished by changing appropriate ECA fields. This structure gives the user a tremendous flexibility to accommodate, through changes in the ECA, variations that occur within a given format.

For non-hard sectored disk formats, the IMDC writes the sector identifier and fills the data part with the fill byte starting with the leading edge of the first index pulse and ending with the leading edge of the next index pulse. For hard sectored disk formats, the format information is written between sector pulses.

Tables 11 and 12 show the possible layout of the track information in table format. Table 11 is for the floppy parameters and table 12 is for the rigid disk format parameters. In table 11 each of the different fields on each track is described. Associated with each field is where the IMDC gets the data, either from the ECA register locations or through the DMA process from the system memory.

ECA Track Format Fields

For the four formats supported by the IMDC, eight parameters are required to specify the format of the recording media to the IMDC. The format parameters are programmed by changing the values of the appropriate ECA fields. The layout of the track format portion of the ECA is given in table 13.

N0 and N1 are ignored by the hard sectored format and only N1 is applicable to the programmable soft sectored format. N5 contains the number of address marks contained in the address mark subfields for the active

Table 10. TRACK FORMAT DEFINITIONS

NAME	SUBFIELD	FORMATS	DESCRIPTION
N0	Index	IBM only	Pre-index Gap. This gap represents the number of bytes that appear prior to the index pulse.
N1	Index	All except hard sect	Index Gap. This gap represents the number of bytes that appear after the index pulse and prior to the ID subfield.
N2	ID, data	All	Preamble count or sync. This is the number of index sync bytes that precede the address mark.
N3	ID	All	Post ID gap. This count is the number of bytes that separate the ID subfield from the data subfield.
N4	Data	All	Post data gap. This count is the number of bytes that separate the data subfield to the beginning of the ID subfield of the next sector.
N5	ID, data	IBM	Address Mark Count. This contains the number of index address marks contained by the subfields. For single density formats, the count is one and for double density the count is three.
	ID, data	Prog	The number of data part address marks is a 1, 2 or 3. The number of ID address marks is always 1.

Table 11. SUMMARY OF FLOPPY FORMAT PARAMETERS

DESCRIPTION	IMDC USES	HEX DATA VALUE	FM CNT	HEX DATA VALUE	MFM CNT	
Pre-index gap	N0	FF	40	4E	80	
Sync field	N2	00	6	00	12	
Index mark	N5	FC/D7[1]	1	C2[2]	3	
Index flag		–	–	FC	1	
Index gap	N1	FF	26	4E	50	
Sync field	N2	00	6	00	12	▲
ID address mark		FE/C7[1]	1	A1[3]	3	
ID address flag		–	–	FE	1	
Cylinder	DMA		1		1	
Side	DMA		1		1	
Sector	DMA		1		1	
Record length	DMA	01[4]	1	01[4]	1	
CRC-CCITT			2		2	Repeat as required
Post ID gap	N3	FF	11	4E	22	
Sync field	N2	00	6	00	12	
Data address mark	N5	FB/C7[1]	1	A1[3]	3	
Data address flag		–	–	FB	1	
DATA (see note 4)		Fill byte	256	Fill byte	256	
CRC-CCITT			2		2	
Post data gap	N4	FF	27	4E	54	▼
Inter-record gap[5]		FF	170	4E	598	

NOTES:
1. Shows data pattern and clock pattern (clock pattern normally FF).
2. Shows data pattern; clock pattern should suppress clock bit between data bit 3 and 4.
3. Shows data pattern; clock pattern should suppress clock bit between data bit 4 and 5.
4. This example is for a 256 byte sector, others will have different values.
5. This is an approximate count.

Table 12. SUMMARY OF RIGID FORMAT PARAMETERS

DESCRIPTION	IMDC USES	HARD SECTOR		SOFT SECTOR		
		Hex Data Value	Pgm Cnt	Hex Data Value	Pgm Cnt	
Index gap	N1		22	4E	22	
Sync field	N2	00	13	00	13	▲
ID address mark	N5	A1[1]	1	A1[1]	1	
ID address flag		FE	1	FE	1	
Cylinder	DMA		1or2		1or2	
Head	DMA		1		1	
Sector	DMA		1		1	
CRC-CCITT			2		2	
Post ID gap	N3	00	3	00	3	Repeat as required
Sync field	N2	00	13	00	13	
Data address mark	N5	A1[1]	1	A1[1]	1	
Data address flag		FB	1	FB	1	
DATA (note 2)		Fill Byte	256	Fill Byte	256	
CRC-CCITT		note 3	2,4,6	note 3	2,4,6	
Post data gap	N4	00	3	00	3	
Inter-sector gap		4E	15	4E	15	▼
Inter-rec gap[4]				4E	346	

NOTES:
1. Shows data pattern; clock pattern should suppress clock bit between data bit 4 and 5.
2. This example is for a 256 byte sector; others will have different values.
3. Can be either a CRC-CCITT, 32-Bit ECC, or 40-Bit ECC Field.
4. Approximate count.

format; valid values are either one, two or three. N2 must be at least four.

Format Parameters

Although exhibiting some differences, the parameters that constitute each of the four formats supported by the IMDC are basically similar. In each format the data on the diskette or disk is separated into a logical data block or sector. The sector becomes the smallest block of information that can be addressed directly by the IMDC.

The programmable soft sectored and the two IBM formats organize the physical disk into a circular path or track, which, in turn, is separated into several sectors. In this scheme, tracks on the media are referenced with respect to a physical index mark. An index mark pulse is generated by the media to indicate the beginning of a track. By specification of a track and sector, the location of a sector on the media is uniquely addressed (for at least one side of the media). The programmable hard sectored format also uses this track and sector structure, however, it differs from the other formats in that in addition to the index pulse, each sector is preceded by a sector pulse.

In each of the formats, a sector can be further separated into two parts or subfields – an ID and a data subfield. The ID subfield contains a unique identifier (or address) and description of the sector. The data subfield contains the actual data of the sector. Both subfields contain three types of information:

1. Address mark – unique character which precedes the data in the subfield. For the MFM formats, (non IBM single density) it is followed by a single character, the address mark flag, which further describes the subsequent data.
2. Data (either actual or about the sector).
3. Error report – pattern generated by the IMDC that is used to verify the data transmitted.

Both fields contain sequences of characters called 'gaps' and 'syncs' that are used to differentiate the sector subfields and to provide an interval that allows any hardware switching to be performed. As a result, these sequences provide any timing compensation due to variations in either the recording media or the disk drive.

For the programmable hard sectored and IBM formats, (formats that use the index pulse), a third subfield, the index subfield, is utilized.

Table 13. ECA TRACK FORMAT

Word	15 – 8	7 – 0
16	N0 – Pre Index Gap	N1 – Post Index Gap
17	N2 – Sync Byte Cnt	N3 – Post ID Gap
18	N4 – Post Data Gap	N5 – Address Mark Cnt

This subfield appears prior to the first physical sector of a track on the recording media and has subfields which contain gap and sync sequences. Unlike the other two fields it occurs only once per track and contains only the address mark information.

Format Table

Table 14 contains the tables used by the IMDC during the format operation. Each table section is unique for the four different format types the IMDC can execute. The table is addressed by the SCWT pointer in the ECA block in RAM.

Provisions for a Bad Sector Substitution by Formatting

The IMDC command provides a convenient mechanism for handling bad sectors on the recording media. For example, consider the identifier (ID subfield) layout for a good sector using the programmable soft sectored format.

	Bytes	Coding Data/Clock
Preamble(sync)	N2	00
Address mark	1	A1/A0
Flag	1	FE (normal)
Cylinder number	1 or 2	
Head number	1	
Sector number	1	
Sector length	1	
CRC-CCITT	2	
Postamble	3	
Post ID gap	N3	

The data part of the sector is:

	Bytes	Coding Data/Clock
Preamble(sync)	N2	
Address mark	N5	A1/A0
Flag	1	FB (normal) or F8 (deleted)
Data	128X**L	L is the number between 0 and 5.
ECC	N6	
Postamble	3	
Post data gap	N4	

The media sectors with defects cannot be used for recording data. The host can replace any bad sectors it encounters during formatting with good sectors. Sector replacement is accomplished via the coding of identifiers (ID subfield) during the formatting. A bad sector is identified by the host placing an X'FF' in the sector length field of the ID part of the sector. The identifier for the bad sector is then extended with information pointing to the replacement sector.

Usually, there will be more than one identical bad sector identifier on the track for the same media defect. In this way, a later correct reading will be possible independent of the position of the media defect. During the read and write, the IMDC will automatically issue a seek to the replacement sector cylinder. There is no restriction on the placement of the substituting sectors. The identifier for a bad sector for the previous example would then appear as:

	Bytes	Coding Data/Clock
Preamble	N1	00
Address mark	1	A1/A0
Flag	1	FE (normal)
Cylinder number	1 or 2	
Head number	1	MSB set to 1 for bad sector
Sector number	1	
Cylinder number	2	
Head number	1	
Sector number	1	
CRC-CCITT	2	
Postamble	4	
Post ID gap	N3	

Table 14. FORMATS

IBM FM	
15	0
01	00
FE	00
80	N3-3
FF	Sector length
01	00
FB	00
Fill byte	00
FF	N4-5
FF	00
01	N5-1
DC	00
FF	00

IBM MFM	
15	0
00	02
A1	FE
80	N3-3
4E	Sector length
00	02
A1	FB
Fill byte	00
4E	N4-5
4E	00
00	N5-1
C2	FC
4E	00

PROG 1 AND 2 CYL	
15	0
00	00
A1	FE
00	N3-3
00	Sector length
00	N5-1
A1	FB
Fill byte	00
00	N4-2
4E	00
4E	00
00	00
00	00

Note that these tables are aligned to an even address boundary.

PROGRAMMABLE RECORD PROCESSING

The following definitions apply:

Key
Character string that the IMDC is to locate and match.

Scan
Search operation performed by the IMDC in trying to match the key.

Field (data item)
Smallest unit of named data. It consists of a string of characters that has a user defined significance. Records are formed by joining several fields together.

Record
Named collection of data items (fields) that has significance to the user.

Key Field
Field that contains the key data.

File
Named collection of all occurrences of given type of record.

The IMDC can be instructed to locate a specified string of characters within a logical data block on the recording media. After the string has been matched, the IMDC can perform the following additional functions:

- Retrieve and store the content of the logical data block
- Process a pointer from the data block
- Locate all data blocks that satisfy some search criteria

These primitive functions constitute the programmable record processing (PRP) capability of the IMDC, which can be used to form the basic support of more sophisticated applications such as:

- Directory processing
- Data block retrieval for data base management systems
- Processing of complex file structures (linked and mapped file structures)
- Searching of multilevel tree or network data structures

The programming for the PRP functions consists basically of two parts. One part involves loading of ECA fields with parameters that define the PRP operations and the physical representation of the data to be processed. The second part consists of table(s) which describe the search criteria on a character by character level.

PRP ECA Fields

The IMDC requires the following information to define the PRP operation:

1. PRP command control
2. Location of scan control word table
3. Location of matched record storage (not a separate PRP field; the IMDC uses the ECA buffer address field for this purpose)
4. Scan termination character
5. Maximum record length

PRP Command Control (1 Word)
This field contains the execution control parameters for the PRP command. The format of the command control field is:

B-tree scan (bit 1) — This parameter enables the B-tree scan function on the IMDC.

No data transfer — Returns pointer in ECA B-tree pointer.

Record type (bit 2) — This parameter contains a code which specifies the record type:

0 Fixed length. The record length is given in the maximum record length (MAXLEN) field of the ECA.

1 Variable length. The record is of variable length and uses the content of the scan terminator field of the ECA to determine the end of the record. If this terminator is not found, the IMDC uses the MAXLEN parameter to terminate the record search.

Location of Scan Control Word Table (2 Words)
This field contains the address of the start of the scan control word table in system memory.

Scan Terminator (1 Byte)
This character specifies the end of the scan field in the record.

Maximum Record Length (MAXLEN – 1 Word)
This field plus one is the maximum length of variable length records and is the length of the fixed length records.

Scan Control Word Table

The scan control word (SCW) table's primary task is to provide the character by character comparison template. It also is used to locate the key field and provide processing instructions for the IMDC as shown below. It must be located on an even word boundary.

15	0
Program length	Record start offset
Number of record to be scanned	
Scan control word(s)	

Bits 8 – 15
DATA – This 8 bit field is used in the following manner depending on the usage defined by the operation.
1. It contains the data to be matched against the key
2. It contains a count for repeating a previous instruction.
3. It contains the character used with the continue opcode.

Bit 5
SUB – This bit is used to indicate the last character of a subkey or the last character of the key to the IMDC. This flag is used by AZFF and SUCFF to control generation of the automatic initialization state for these devices. This bit must be set for the last byte of the key field, i.e., when bits 6,7 are 0,0.

Bit 4
ORENA – This bit is used to indicate to the PRP processor that the result of the previous subkey comparison is to be 'ORed' with the next logical comparison. This bit must be set for the last byte of the key field, i.e. when bits 6, 7 are 00.

Bits 6, 7
OP – This 2 bit field is a command to the IMDC to perform one of the following operations independent of the resultant logical processing being performed.

7 6
0 0 Indicates the last byte of the key field.
0 1 Read this character (data) into system memory from this character until either:
  1. Stop read command is issued, or
  2. End of key, and no match has occurred.
1 0 Stop reading
1 1 No-op

Bits 0 – 3
OP CODE – This field is used to indicate either the logical comparison or some special control function is to be performed.

CODE	OPERATION
0	Repeat previous instruction (data + 1) times
3 – 1	Not used
4	SUCFF is unchanged
5	Continue scanning, no-op until 'data' byte is read
6	Not used
7	If ORFF is set, then store this character into the FIFO. If ORFF is not set, then use last point in FIFO as address of next sector to be scanned on the next level of the tree or plex.
8	Reset SUCFF if (CHAR GT Data) • (AZFF = 1).
9	Reset SUCFF if (CHAR GT or EQ Data) • (AZFF = 1).
10	Reset SUCFF if (CHAR LT Data) • (AZFF = 1).
11	Reset SUCFF if (CHAR LT or EQ Data) • (AZFF = 1).
12	Reset SUCFF if CHAR = Data.
13	Reset SUCFF if CHAR NEQ Data.
14, 15	SUCFF is unchanged

The number of scan control word tables required to describe a PRP function is determined by the data structure organization. For records organized in a relational structure, a single SCW table is required to provide the pertinent parameters. A complete description of the PRP operation is defined by the single SCW table and the ECA fields.

For records organized into a tree or plex structure, each level generally requires a new SCW table. Because each level may have a different record structure, certain ECA parameters describing the record structure must be updated (see PRP ECA fields discussion).

Note that a repeat operation cannot follow a continue operation.

Location of Matched Record Storage (2 Words)
The IMDC will use the previously defined 'buffer address' field.

PRP Status Flip Flops

To understand the operation of these flip flops, one must differentiate between the 'automatic initialization state' and the 'logical operation' of each of these internal devices. If this distinction is not made, some apparent conflicts in their logical state appears to exists. For example, a contradictory condition appears to occur for the accumulated zero flip flop (AZFF) for the last character of the subkey. According to the description of the flip flop, it is reset if the comparison fails. However, in the same description, it is stated that the AZFF is set if the character is the last of the subkey string. This apparent contradiction is easily explained; it is simply a problem of failure to recognize the sequence of events that is involved in execution of the 'logical operation' and the 'automatic initialization state'. The logical operations are performed by the flip flops prior to the automatic initialization state.

In order to help in making this distinction, the operation of each of the flip flops is described in terms of the logical operation and the automatic initialization state rather than in terms of the set/reset conditions.

Accumulated Zero Flip Flop (AZFF)
This flip flop is used to report the status of each character by character comparison. It is used to identify the first unsuccessful character by character match.

Logical Operation — The flip flop is reset on the first unequal character comparison.

Automatic Initialization State — The operation of the flip flop is initialized to the set condition:
1. At the beginning of a new record.
2. At the end of a subkey prior to the next subkey. The SUB bit of the SCWT indicates this condition.

Success Flip Flop (SUCFF)
This flip flop can be used to indicate the logical status of the subkey match operation. The AZFF cannot be used for this purpose since its primary task is to respond to the character by character search operation.

Logical operation — The SUCFF is reset by the interaction of the SCWT operation (OP code) and the status of the AZFF.

Automatic initialization state — This flip flop is set for the following conditions:
1. At the beginning of a new record.
2. At the end of a subkey if the ORENA bit of the SCWT is set.

OR Flip Flop (ORFF)
This flip flop is used to indicate the resultant scan status for the entire key field. The ORFF will always be in the set condition at the conclusion of a successful key scan, regardless of the logical operation performed between the subkeys.

Logical operation — The ORFF is set at the end of a successful subkey comparison; this condition is indicated by the SUCFF and the ORENA bit of the SCW table both being set.

Automatic initialization state — RFF is reset for the following conditions:
1. At the beginning of the first record.
2. At the beginning of all records to be scanned.

Programmable Record Processing Operation

Although the programmable record processing command utilizes most of the basic disk read sequence described earlier, its execution involves significantly more character processing than is performed for the basic read operation. For the PRP command, the function of the IMDC memory is split; one portion is allocated to store the PRP program contained in the SCWT table and the remaining portion is used as a FIFO buffer. The term 'FIFO' for this discussion of the PRP command refers to the memory available after the SCWT has been loaded. The command is 'normally' executed in the following sequence:
1. Transfer the SCW table data from system memory to a portion of the IMDC memo-

ry. The remainder of the IMDC memory is used as a FIFO to store the data from the record.
2. Perform disk positioning I/O to locate the drive read/write heads to the first sector at which the PRP is to operate.
3. Perform a read of the sector ID part to locate the data part.
4. Read an operand from the data part of the sector. The IMDC performs processing of this operand in order to locate the key field. This step is repeated until the key field is located.
5. After the key field is located, it is compared character by character with the SCWT template. This step is repeated until a match or no match with the key is determined.
6. If the scan does not produce a match the current record is skipped and the next record is located for processing by repeating steps 3, 4 and 5.

Programming Parameters Supplemental Information

The following describes programming parameters in more detail.

Record Format Types

Records to be processed by the IMDC, regardless of the overall data organization, can be formatted in one of two types.

1. Fixed – Records consist of a fixed number of characters. The user must pass a record length parameter value to the IMDC.
2. Variable – Records are of variable length. Requires the user to specify a unique termination character to identify the end of each record.

Since all records organized in the relational data structure are identical, the selected record format is applicable to all the records being processed. For data organized in a tree or plex structure, each level may have a different format type, therefore, each requires this parameter to be specified.

Key Field Location

The IMDC allows the user to specify the location of the key field in the record. The SCW allows the user to program the necessary information to locate the key field. The record format type determines how the key field is specified.

1. For fixed length records, the key start occurs at some fixed number of characters from the beginning of the record. For these records, the user must provide the IMDC with this count.
2. For variable length record, the key field location is specified by the number of data fields occurring prior to it in the record. Data fields consist of data that is bordered by a 'marker' character. Normally, this character will be some unique control character like a tab, line feed or return code. The PRP hardware will count the occurrence of the markers to locate the key field. Specification of the variable length field requires the user to provide the 'marker' as a parameter to the PRP.

Record Offset Start

The recording media often contains some header or identification information prior to the first record. The IMDC allows the user to specify where the PRP control is to become active. This allows the IMDC to skip any information that is not to be processed by the PRP. This parameter is referred to as the 'record start offset'.

Number of Records to Process

Because a user does not always know how many records are to be processed, the IMDC provides mechanisms to limit the number of records that are to be processed. First, the user is required to specify the 'number of records to process' parameter which is a count of the maximum number of records to be processed. In addition to this count, the user must also specify either an 'end of record' character for variable length fields or a record length count for fixed length records. The IMDC will monitor the number of records processed and will terminate the processing when the total equals the 'number of records to process', or if the 'buffer length' is exceeded.

Programming the PRP Feature

One of the most powerful aspects of the PRP feature is the capability it provides the user to effectively specify the search criteria on a character by character basis. The PRP feature not only allows the user to establish the criteria for success, but also to specify the action to be taken after completion of a search. The PRP program consists of two major parts. The first part essentially defines the basic requirements of the processing and the structure of the records; it is given once during initialization of the feature to the IMDC. It consists of loading several ECA fields dedicated to defining the parameters of the PRP operation to the IMDC.

Scan Control Word Table

The second part of the PRP program consists of a template which is used to perform the character by character match of the record data as it read from the media. The template consists of the character string data to be matched and processing instruction for each character of the string. The IMDC utilizes a 16-bit word to implement this comparison/instruction. It is loaded by the host into system memory and is transferred by the IMDC into its memory before execution of the PRP command. The SCWT is used in conjunction with three internal IMDC statuses to control and monitor the operation of the PRP search. Because of the interaction, they will be described prior to presentation of the table format.

PRP Key Status

To implement the PRP feature, the IMDC performs a character string search of the key field against the scan control word data. The IMDC generates and stores the status of this comparison in three flip flops dedicated to reporting the status of this operation. The logical operation of these flip flops is as follows:

For the most general case, the key field is composed of substrings. For convenience these substrings shall be referred to as subkeys for the remainder of this section. The desired logical result of the scanning of the entire key field is the logical combination of the constituent subkeys. For this general situation, there is the possibility of three levels of comparison statuses that occur in attempting to match a single key field – character, subfield, and entire key field each has a status that must be monitored by the IMDC. The IMDC has three flip flops that generate a status that allows these conditions to be monitored. Consider the example:
YYYYYY985DEF12580XXXXXXXXXXXX

This key field consists of five subkeys; YYYYYY, 985, DEF, 12580 and 'XXXXXXXXXXXXX', referenced arbitrarily as subkeys A, B, C, D, and E, respectively. Suppose that each subkey has the following logical condition associated with it.
A – Value is not processed by the PRP
B – Value greater than 980
C – Value equal to DEF
D – Value less than 20000
E – Value is not processed by the PRP

To perform the scan operation for this example requires that the IMDC be able to generate and monitor the condition of the character by character comparison (level 1), each subkey (level 2) and finally the entire key (level 3). The SCWT has been structured to allow specification of such combinations.

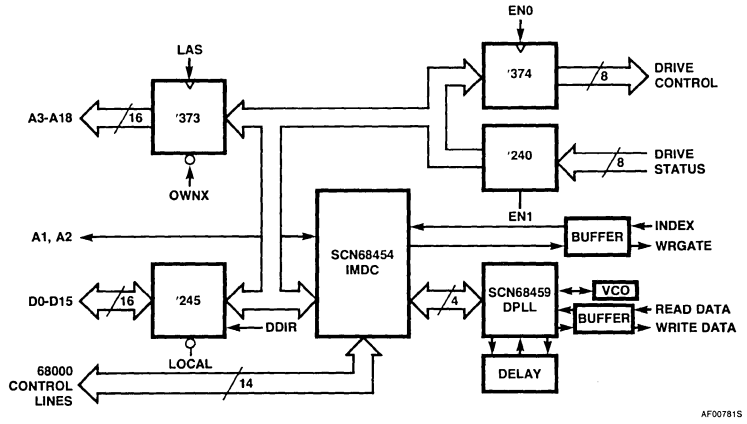


Figure 4. IMDC Application

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltage	−0.3 to +7.0	V
Input voltage[2]	−0.3 to +7.0	V
Operating temperature range[3]	0 to +70	°C
Storage temperature	55 to +150	°C

DC ELECTRICAL CHARACTERISTICS $V_{CC} = 5V \pm 5\%$, $V_{SS} = 0V$; $T_A = 0°C$ to $+70°C$[4,5]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IH}	Input high voltage	$I_{IL} = 20\mu A$	2.0	V_{CC}	V
V_{IL}	Input low voltage		GND − 0.75	0.8	V
I_{IN}	Input leakage current RERUNN, RESN, SCLK, IRQN, IACKN, BRN, BGN, BGACKN, EN0, EN1, UAS, LAS, REDAT, WRGATE, WCLOCK/RCLOCK, WRDAT, WRCLK, TICKLER, SECTOR/INBEN, LOCAL, OWNN, DDIR	5.25V		20	μA
I_{TSI}	Three-state (off state) input current A1, A2, D0 − D15, ASN, UDSN, LDSN, R/WN, DTACKN	2.4V/0.4V		20	μA
V_{OH}	Output high voltage A1, A2, D0 − 15, ASN, UDSN, LDSN, R/WN, DTACKN, EN0, EN1, UAS, LAS, WRGATE, WRDAT, WRCLK, TICKLER, LOCAL, OWNN, DDIR	$I_{OH} = -400\mu A$ $I_{OL} = 6.3mA$	2.4	V	
V_{OL}	Output low voltage A1, A2, D0 − D15, ASN, UDSN, LDSN, R/NN, DTACKN IRQN, BRN, BGACKN, EN0, EN1, UAS, LAS, WRGATE, WRDAT, WRCLK, TICKLER, LOCAL, OWNN, DDIR			0.5	V
P_D	Power dissipation			1.5	W
C_{IN}	Capacitance	$V_{in} = 0V$, $T_A = 25°C$ $f_o = 16MHz$		10	pF

NOTES:

1. Stresses above those listed under absolute maximum rating may cause permanent damages to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltage larger than the rated maximum.
3. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8 and 2.0V as appropriate.

2

AC ELECTRICAL CHARACTERISTICS $V_{CC} = 5V \pm 5\%$, $V_{SS} = 0V$; $T_A = 0°C$ to $+70°C$[4,5] (see figures 5 – 18)

NO.	FIGURE	CHARACTERISTICS	TENTATIVE LIMITS		UNIT
			Min	Max	
1	5	Cycle time		62.5	ns
2	5	Clock pulse width low	21		ns
3	5	Clock pulse width high	21		ns
4	5	Rise time		10	ns
5	5	Fall time		10	ns
6	6	Reset pulse width	1		μs
7	7, 8	A1 and A2 set-up to CSN low	0		ns
8	7, 8	Local low after CSN low		1.2	μs
9	7	D0 – D15 valid data from ASN, CSN, and UDSN or LDSN low		1.2	μs
10	7, 9	DTACKN low after D0 – D15 valid data	62.5		ns
11	7	CSN high after ASN, UDSN, LDSN, A1 and A2		10	ns
12	7, 8	D0 – D15 hold after CSN high	0		ns
13	7, 8	Local high after CSN high		100	ns
14	7	DTACKN high after CSN high		100	ns
15	7, 8	CSN low time	1.2		μs
16	7, 8	A1 and A2 HOLD after CSN high	0		ns
17	8	DDIR low after CSN low		1	μs
18	8	DTACKN low after CSN low		1.4	μs
19	8	R/WN low before CSN low	0		ns
20	8	R/WN low after CSN high	0		ns
21	9	IACKN low after last of ASN AND LDSN		30	ns
22	9	IRQN high after IACKN low		140	ns
23	9	Local low after IACKN low		1.2	μs
24	9	D0 – D15 valid after last low of ASN, LDSN, IACKN		1.2	μs
25	9	IACKN low time	1.2		μs
26	9	D0 – D7 hold after LDSN high	10		ns
27	10	BRN high after BGACKN low		1.2	μs
28	11	OWNX low after BGACKN low		1.2	μs
29	11, 12	D0 – D15 valid before either UAS or LAS low	65		ns
30	11	DDIR low after OWNX low		500	ns
31	11, 12	D0 – D15 valid after either UAS or LAS low	5		ns
32	11, 12	LAS low before ASN low	0		ns
33	11	ASN, data strobes width low (read)/ASN write	440		ns
34	11	Local low after data strobes low		20	ns
35	11, 12	D0 – D15 valid after data strobes high	0		ns
36	11	Local high after data strobes high	0		ns
37	11	DDIR high after ASN high	20		ns
38	11, 12	OWNX high after ASN high		400	ns
39	11, 12	BGACKN high after OWNX high		30	ns

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTICS	TENTATIVE LIMITS		UNIT
			Min	Max	
40	12	Data strobes with low	80		ns
41	12	R/WN low before ASN low	35		ns
42	12	D0 – D15 valid before data strobes low	62.5		ns
43	12	R/WN low after data strobes high	30		ns
44	13	D0 – D15 valid before EN0 high	40		ns
45	13	D0 – D15 valid after EN0 low	40		ns
46	14	D0 – D15 valid after EN1 low		40	ns
47	14	D0 – D15 valid after EN1 high	0	50	ns
48	15	WRGATE high after SECTOR/INDEX high	TBD		
49	15	WRDAT or WRCIK valid after WRGATE high	TBD		
50	16	REDAT high before WCLOCK/RCLOCK high	20		ns
51	16	REDAT high after WCLOCK/RCLOCK high	20		ns
52	18	RERUNN low before DTACKN low		20	ns
53	18	RERUNN low after DTACKN high	320		ns
54	11	DTACKN low after D0 – D15 valid		-62.5	ns
55	8	D0 – D15 valid after data strobes low		62.5	ns
56	12	Local low before data strobes low	50		ns

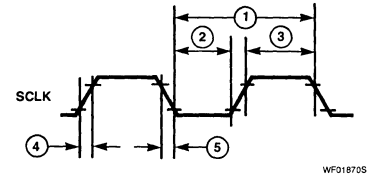


Figure 5. Input Clock Timing

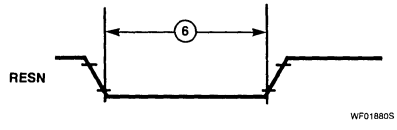


Figure 6. Reset Timing

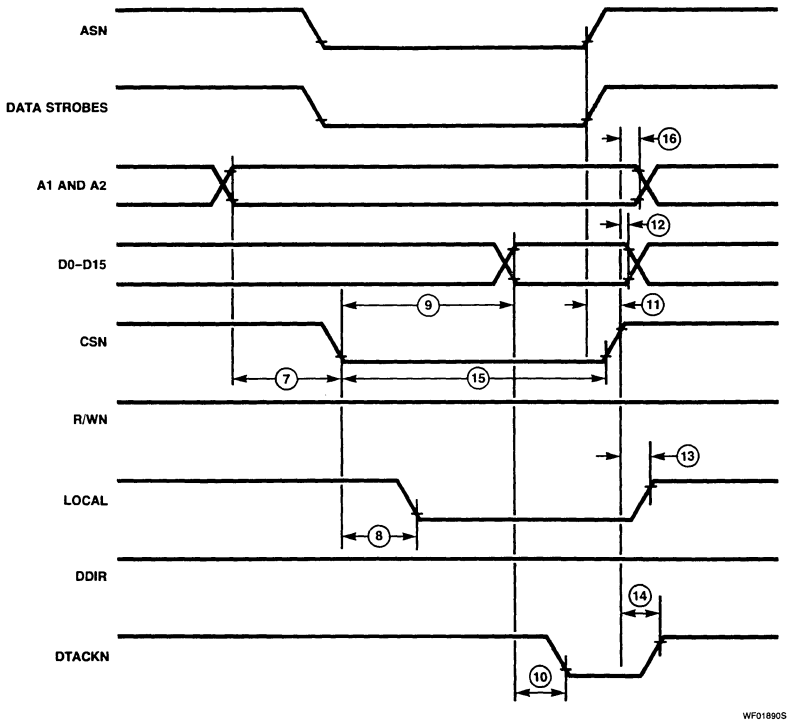


Figure 7. IMDC Register Read Cycle

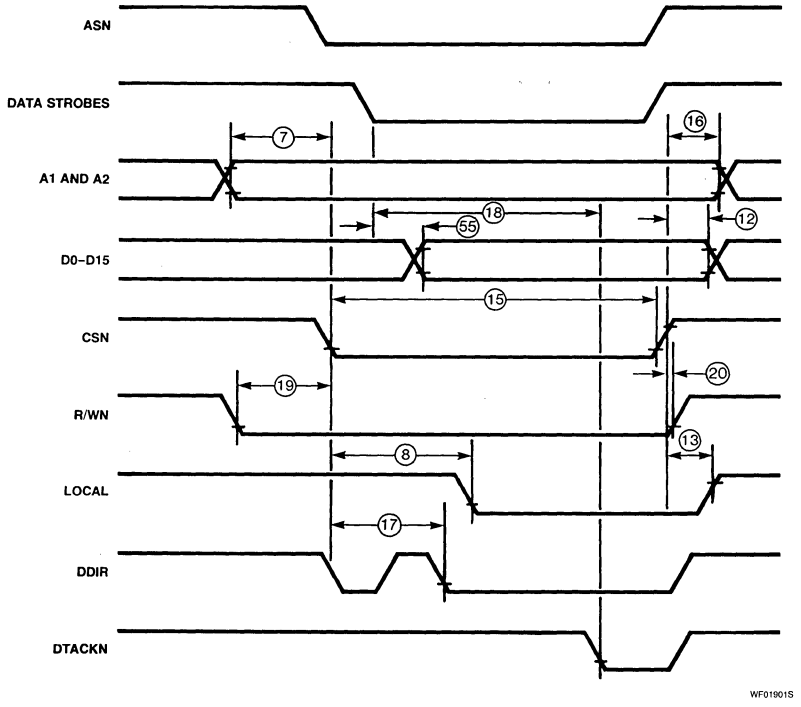


Figure 8. IMDC Register Write Cycle

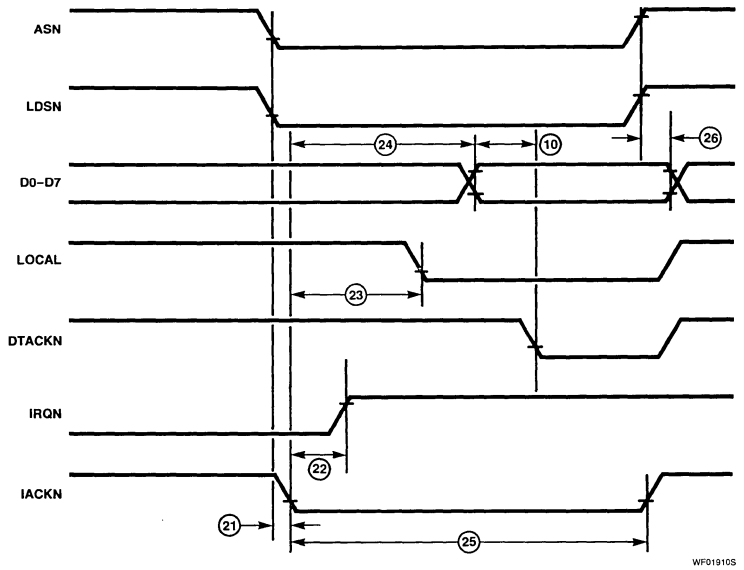


Figure 9. Interrupt Acknowledge Cycle

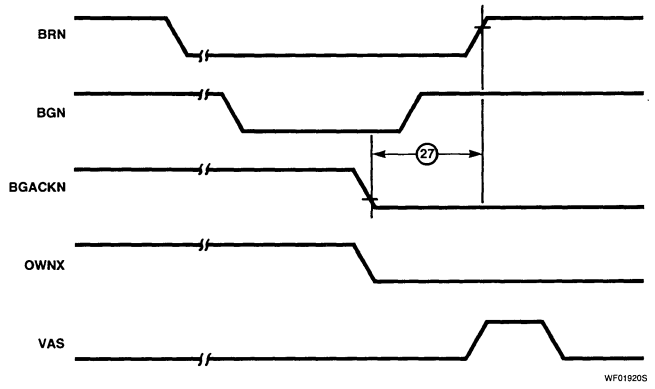


Figure 10. Bus Arbitration

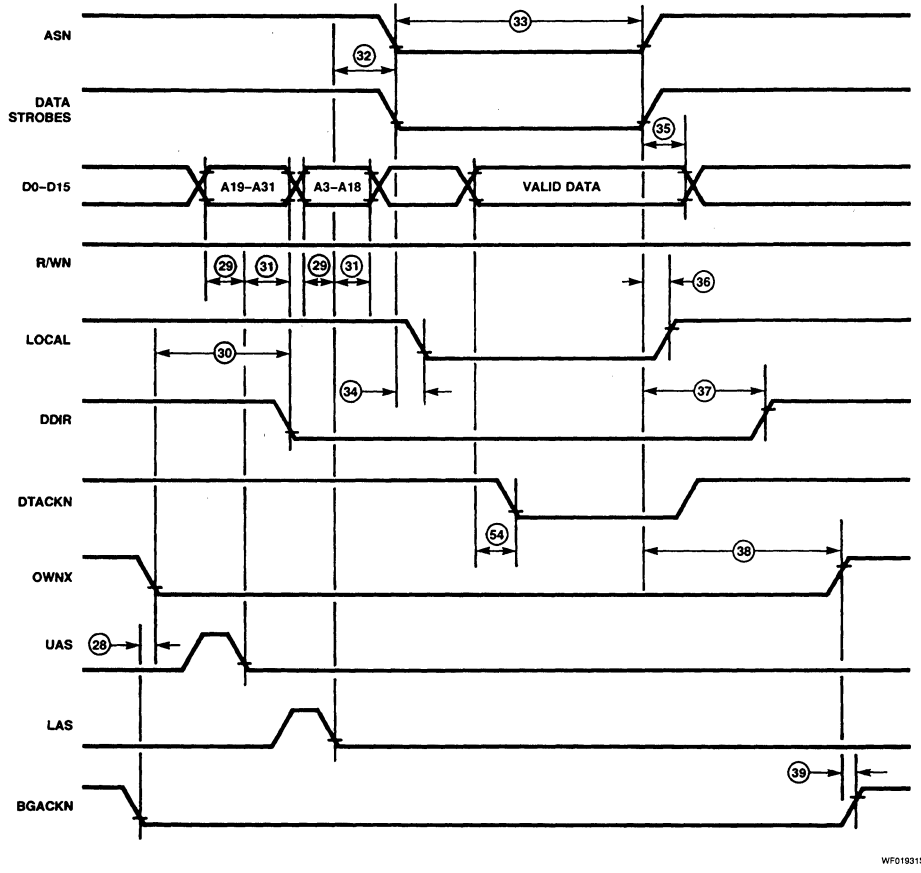


Figure 11. Single 16-Bit DMA Read From Memory Cycle

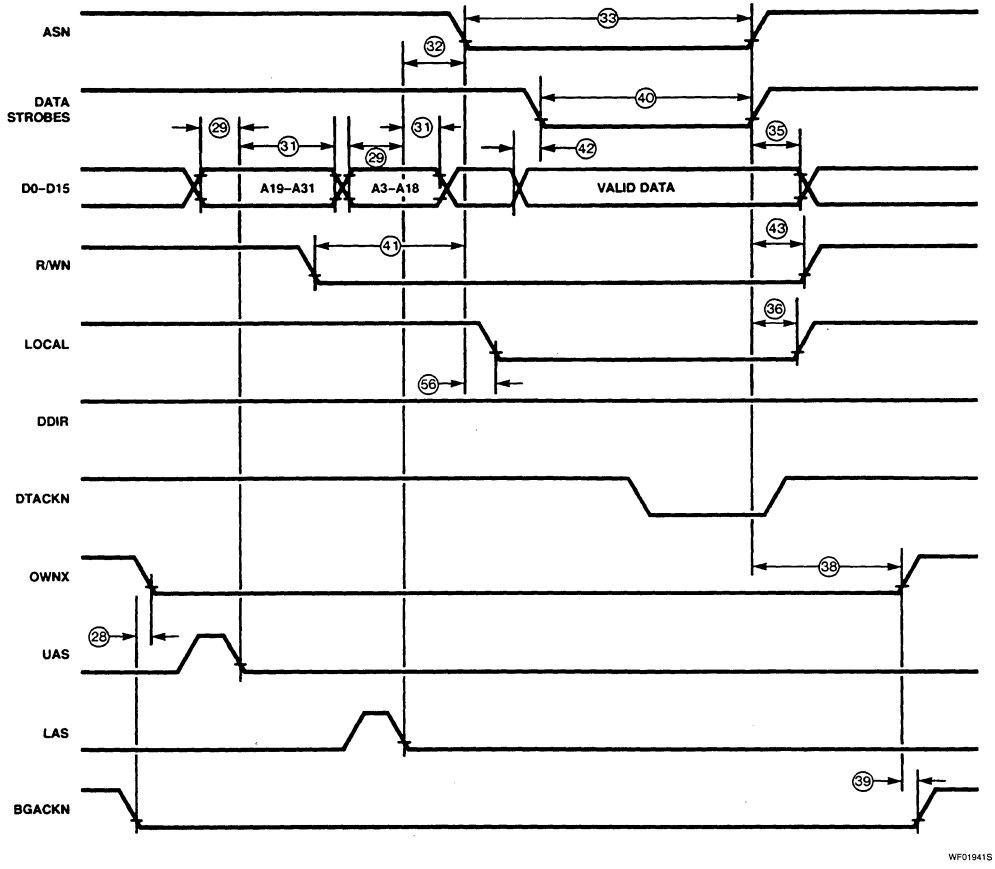


Figure 12. Single 16-Bit DMA Write to Memory Cycle

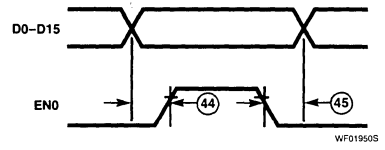


Figure 13. IMDC Output Port Timing

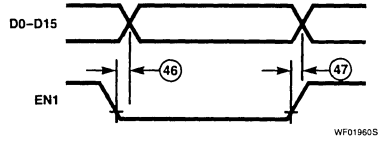


Figure 14. IMDC Input Port Timing

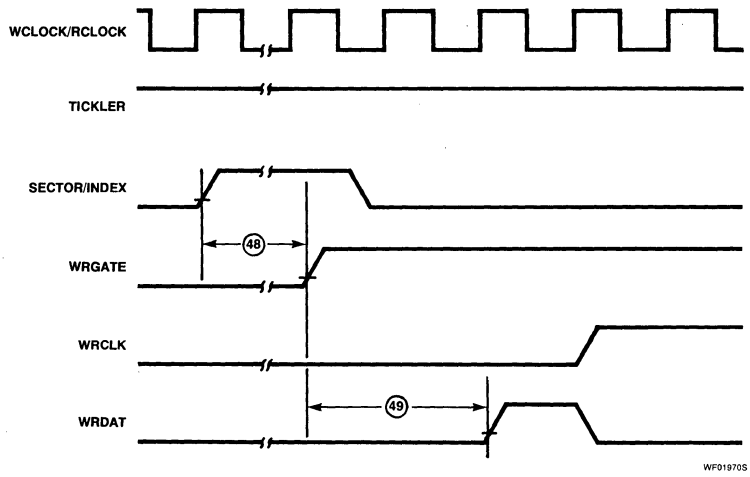


Figure 15. Write Operation

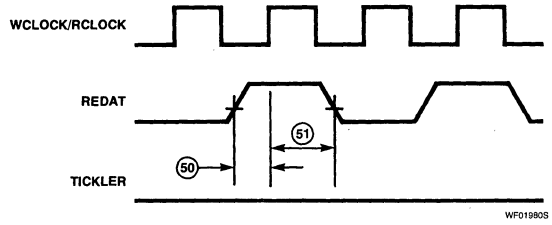


Figure 16. Read Operation

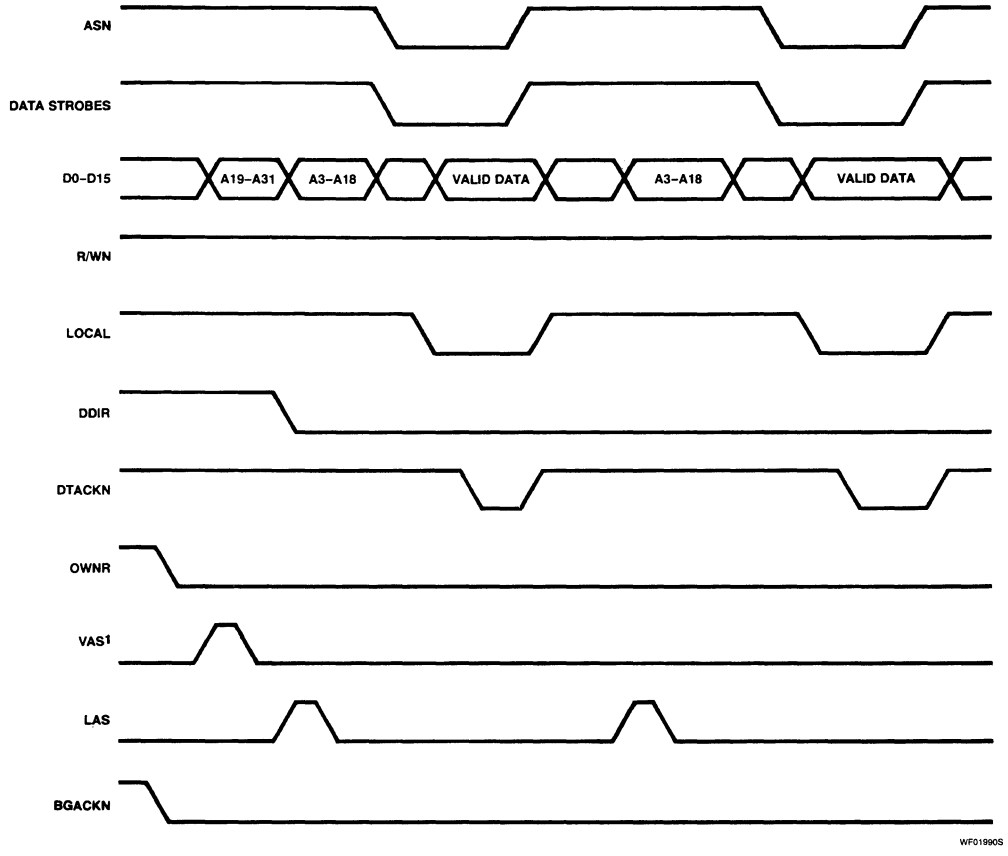


Figure 17. Example of Multiple DMA Read Cycles

[1]If the high order address does not change, VAS will not be asserted.

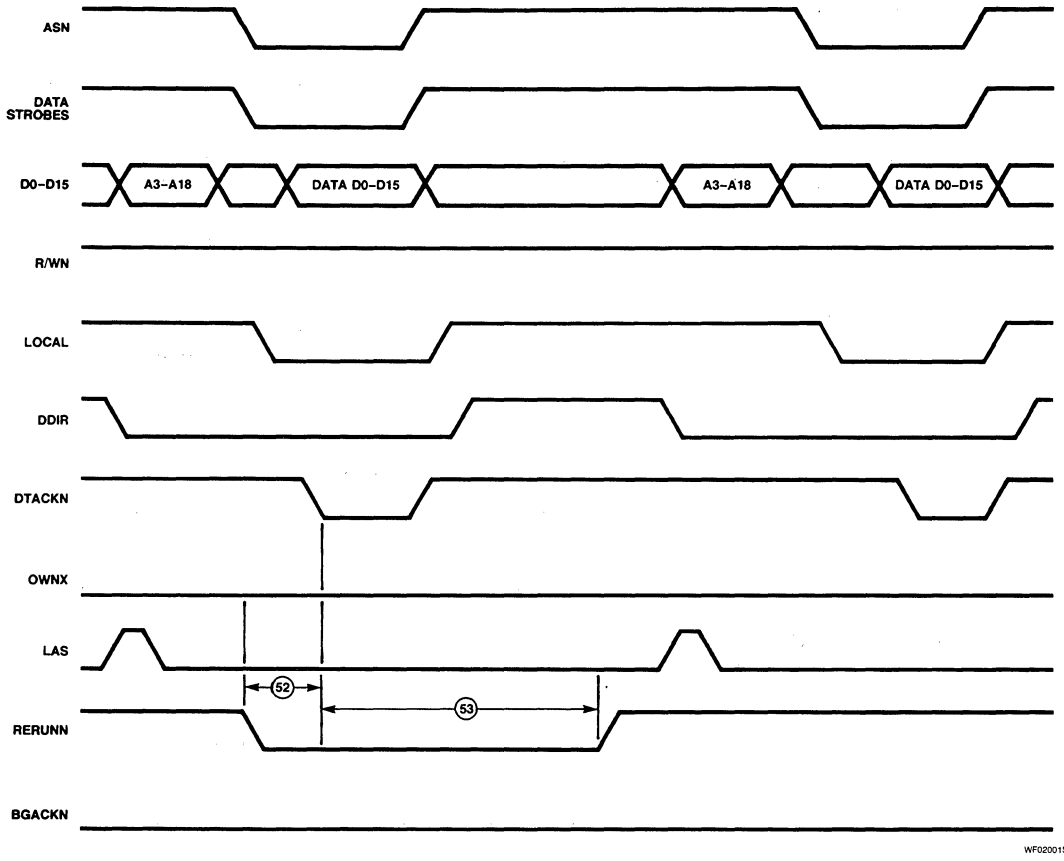


Figure 18. RERUNN Asserted During DMA Read Cycle

SCB68459
Disk Phase Locked Loop (DPLL)

Preliminary Specification

Microprocessor Products

DESCRIPTION

The SCB68459 Disk Phase Locked Loop (DPLL) is a bipolar device that complements the SCN68454 Intelligent Multiple Disk Controller (IMDC). Together, with an external voltage controlled oscillator (VCO), the DPLL and IMDC provide all the functions required to control up to four disks with SA800, ST500, or SA1000 type interfaces.

The DPLL uses an external VCO for the variable clock rate which tracks the read data from the disk unit. This VCO can be any device that can interface to the DPLL. The IMDC can control up to four disks; if these are all the same, only one DPLL is needed. If different disk types are driven by the same IMDC, then a DPLL is required for each disk drive type.

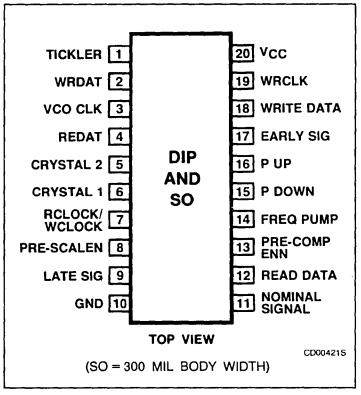
The SCB68459 DPLL operates by producing an oscillator frequency to match the frequency of an input signal. In this locked condition, any slight change in the input frequency (called jitter) will appear as a change in phase between the input frequency and the VCO frequency. This phase shift then acts as an error signal to change the frequency of the local DPLL VCO to match the input frequency.

The SCB68459 is constructed using Signetics extended performance logic (EPL) bipolar technology.

FEATURES

- **Supports composite data rate of 100KHz to 10MHz**
- **On chip multiplexer for read and write clock**
- **Supports MFM and FM data formats**
- **Generates synchronous clock for read data**
- **Built in pre-compensation circuit**
- **Phase detector and frequency detection for improved operation**
- **External loop gain control**
- **Minimal amount of external components required for operation**
- **Single +5 volt power supply**
- **Crystal controlled write clock**

PIN CONFIGURATION

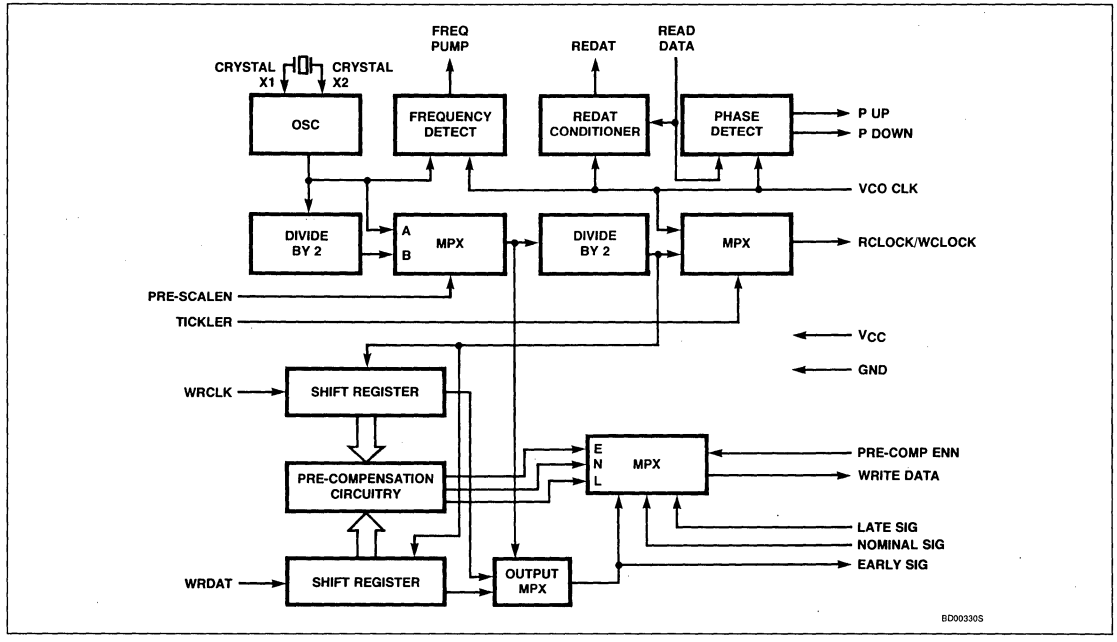


Disk Phase Locked Loop (DPLL) SCB68459

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0 to 70°C
Ceramic DIP	SCB68459CAI20
Plastic DIP	SCB68459CAN20
Small Outline	SCB68459CAD20

BLOCK DIAGRAM



Disk Phase Locked Loop (DPLL) SCB68459

PIN DESCRIPTION

The pin description table describes the function of each of the pins of the DPLL. Signal names ending in 'N'' are active low. All other signals are active high.

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
WRDAT	2	I	**Write Data Pattern:** Active high. WRDAT is a non-return to zero (NRZ) input which contains the data information; i.e., it is the polarity of the WRITE DATA to be output in the second half of the bit cell time.
WRCLK	19	I	**Write Clock Pattern:** Active high. WRCLCK is an NRZ input which contains the clock information; i.e., it is the polarity of the WRITE DATA to be output during the first half of the bit cell time.
PRE-SCALEN	8	I	**Pre-Scale:** Active low. When PRE-SCALEN is low, the crystal frequency is divided by 4 (the normal divide by two plus an additional divide by 2) before being used as WRCLOCK. When PRE-SCALEN is high, the crystal frequency is not pre-scaled. Typically used to encode/decode both FM and MFM with the same crystal frequency.
TICKLER	1	I	**Tickler:** Active high. TICKLER controls the operation of the RCLOCK/WCLOCK output signal. If TICKLER is high, internal logic routes the crystal controlled clock to the output. If TICKLER is low, the PLL synchronized to the incoming READ DATA will generate RCLOCK. This clock is then routed to the output signal RCLOCK/WCLOCK.
RCLOCK/ WCLOCK	7	O	**Read Clock/Write Clock:** RCLOCK is the output when the TICKLER signal is low. As an output, it is the clock phase locked onto the incoming READ DATA signal. RCLOCK is twice the data frequency. WCLOCK is the output of the crystal oscillator, divided internally by 2, and will output continuously while the DPLL is enabled and the TICKLER signal is high. WCLOCK defines the bit cell frequency.
PRE-COMP ENN	13	I	**Pre-Compensation Enable:** Active low. When PRE-COMP ENN is high, no write compensation is applied to the outgoing data. When it is low, the data stream is write pre-compensated.
FREQ PUMP	14	O	**Frequency Pump Error Signal:** Current output which is summed through a resistor along with P UP and P DOWN. The three outputs are then used to charge or discharge a capacitor which generates an error voltage for the VCO.
P DOWN	15	O	**Pump Down Error Signal:** Current output which is summed through a resistor along with FREQ PUMP and P UP. The three outputs are then used to charge or discharge a capacitor which generates an error voltage for the VCO.
P UP	16	O	**Pump Up Error Signal:** Current output which is summed through a resistor along with FREQ PUMP and P DOWN. The three outputs are then used to charge or discharge a capacitor which generates an error voltage for the VCO.
VCO CLK	3	I	**Variable Clock:** Generated by an external VCO.
CRYSTAL X1	6	I	**Crystal 1:** Input connection provided to attach a crystal for the internal oscillator. The crystal frequency is defined to be twice the data bit rate.
CRYSTAL X2	5	I	**Crystal 2:** Input connection provided to attach a crystal for the internal oscillator. The crystal frequency is defined to be twice the data bit rate.
EARLY SIG	17	O	**Early Data Signal:** Active high. Used as the input to a user supplied delay line or delay circuit. The delays are determined by the requirements of the particular disk unit to which the DPLL is attached. If pre-compensation delays are enabled by the signal PRE-COMP ENN, this signal together with NOMINAL SIG and LATE SIG are used to generate the WRITE DATA signal.
NOMINAL SIG	11	I	**Nominal Data Signal:** Active high. This input is used by the DPLL together with the EARLY SIG output and the LATE SIG to generate the write pre-compensation delays. The user should insert a delay line between the EARLY SIG output and the NOM SIG input and between the output of the first delay line and the LATE SIG input (these delays being the delay between an early and nominal signal or a nominal and late signal, respectively). The values of these depend on the disk drive to be used. If pre-compensation is not enabled, NOMINAL SIG is used to generate the WRITE DATA signal. In systems where pre-compensation is never used, the EARLY SIG output should be connected to the NOMINAL SIG input.
LATE SIG	9	I	**Late Signal:** Active high. This input is used by the DPLL together with the NOMINAL SIG to generate the write pre-compensation delays. The user should insert a delay line between the NOM SIG input and the LATE SIG input. The value depends on the disk drive to be used.
READ DATA	12	I	**Read Data:** Active high. This is the composite clock and data read from the disk unit. It is used as the incoming signal for the phase lock loop.
WRITE DATA	18	O	**Write Data:** Active high. This is the output signal, write pre-compensated if so enabled, formed from the WRDAT and WRCLK inputs.
REDAT	4	O	**Composite Read Data:** Active High. REDAT is the output signal which reflects the READ DATA input signal but synchronized to the RCLOCK generated by the internal PLL. The REDAT signal will be asserted for one period of the RCLOCK signal following the assertion of READ DATA.
V_{CC}	20	I	**+5V ±5% power input.**
GND	10	I	**Signal and power ground input.**

Disk Phase Locked Loop (DPLL) SCB68459

DPLL FUNCTIONS

Read/Write Clock

The DPLL provides a read/write clock for the IMDC. This clock is either derived from a crystal or from a PLL locking onto the combined data and clock stream from the disk.

Generation of the Output Data

The DPLL inputs two pieces of information, the NRZ data and clock information bits in parallel, from the IMDC. It combines them into the same cell by issuing the clock during the first half of the first half of the bit cell time, and the data information during the first half of the second half of the bit cell time. This information then becomes the WRITE DATA signal which goes to the disk drive.

Pre-Compensation

In addition, the DPLL provides facilities for pre-compensating the data during the writing of the data to disk. The write pre-compensating algorithm is built into the chip. Note that this feature can be disabled where necessary.

The logic in the DPLL does the pre-compensation algorithm in a slightly different manner than with a ROM lookup table. If the different data and clock patterns which require compensation are drawn out, only two patterns are found. The two patterns are a series of ones followed by a zero, or a series of zeros followed by a one. By using a shift register and simple logic, the signals necessary for gating early, late, and nominal can be determined (the last being the absence of the previous two). This is the method which is implemented in the DPLL.

PLL Function

With a phase locked loop (PLL) circuit in the read chain, the data can be recovered more accurately. The PLL provides a clock signal which is derived from the read data stream and tracks it through a reasonable variation. The clock signal is used by the decoding hardware to sample the read data stream.

Delay Lines

The delay lines referred to in this document can be purchased as a single unit or constructed with logic and passive components. The delay line, typically, has a single input with several outputs. The outputs reflect the input but delayed by some period of time. A delay line for an eight inch MFM floppy disk drive, for example, would have each output delayed 175nsec from the other. The first output is delayed 175nsec from the input and the second output is delayed 175nsec from the first output. The delay time actually used in each application would be the function of the disk drive and specified by the manufacturer.

OPERATION

The PLL in the DPLL is intended to be used with an external VCO. As shown in figure 1, the DPLL interfaces between the IMDC and the disk unit. The only other attachments are the resistors and capacitor for the external VCO. These values should be chosen for the operating characteristic required by the system. For IBM System 34 interfaces, the decode of side zero, track zero, would be used for the PRE-SCALEN input. This input would then automatically divide the crystal controlled clock output for RCLOCK/WCLOCK. The crystal used with the DPLL should be twice the data rate for the interfaced disk unit.

DESIGN NOTE

Refer to figure 1 (R1 > R2 > R3). The recommended values are:

R1 = 100K ohms
R2 = 10K ohms
R3 = 1K ohms

R1 X C1 = 5 X period
R2 X C1 = 50 X period
R3 X C1 = 500 X period

Clock Considerations

The on-chip oscillator can be controlled by either one of the following methods:

1. Crystal – if precise timing is required.
2. External drive – If application requires that the DPLL be driven from a system clock.

Crystal Timing

When a crystal is used, the on-chip oscillator operates at the resonant frequency (f_0) of the crystal. The series-resonant quartz crystal connects to the DPLL via pins 6 (CRYSTAL X1) and 5 (CRYSTAL X2). The lead lengths of the crystal should be approximately equal, and as short as possible. Also, the timing circuits should not be in close proximity to external sources of noise. The crystal should be hermetically sealed (HC type can) and have the following electrical characteristics:

Type – Fundamental mode, series resonant
Impedance at fundamental – 35ohms maximum
Impedance at harmonics and spurs – 50ohms minimum

The resonant frequency (f_0) of the crystal is related to the desired cycle time (T) by the equation: $f_0 = 2/T$, thus for a cycle time of 200 nanoseconds, f_0 = 10MHz. The DPLL will operate at crystal controlled frequencies from 4MHz to 20MHz.

Using an External Clock

The DPLL can be synchronized with an external clock by simply connecting the appropriate drive circuits to the CRYSTAL X1 and CRYSTAL X2 inputs (see figure 2.)

Disk Phase Locked Loop (DPLL) SCB68459

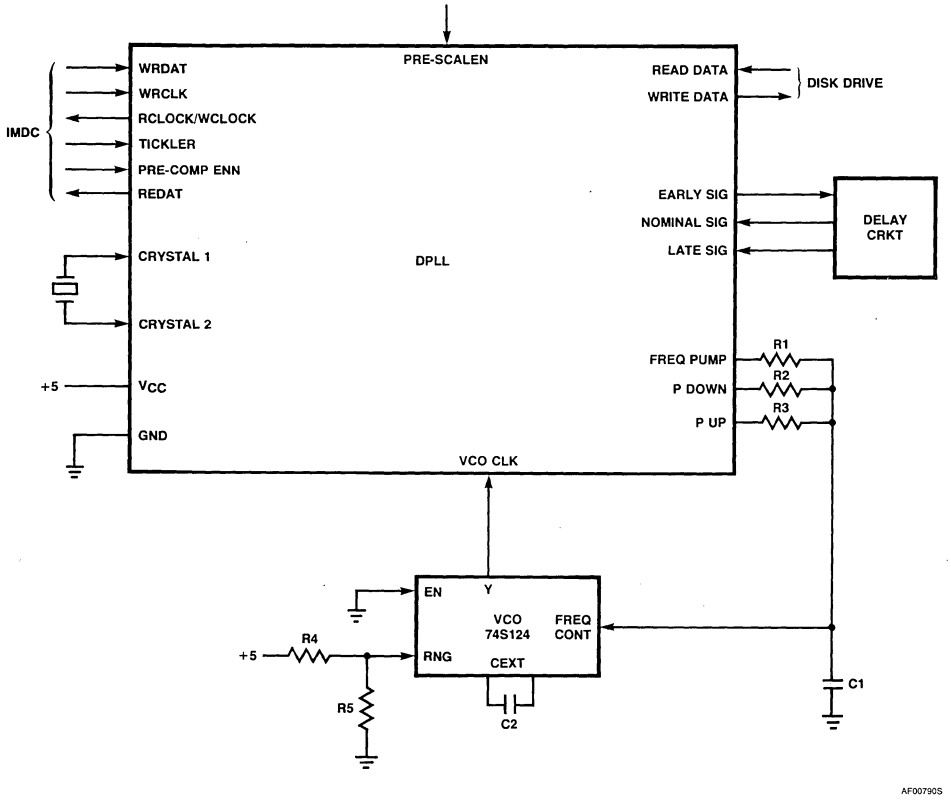


Figure 1. Typical Application

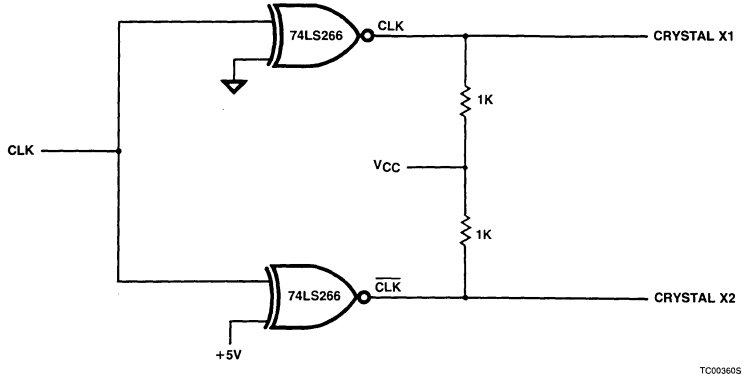


Figure 2. Recommended Clock Circuitry

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltage	−0.5 to +7.0	V
Input voltage	−0.5 to +5.5	V
Operating temperature range[2]	0 to +70	°C
Storage temperature	−65 to +150	°C

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5V ±5%, T_A = 0°C to +70°C [3,4,7]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
V_{OL}	Output low voltage	I_{OUT} = 5.3mA		0.4	V
V_{OH}	Output high voltage	I_{OUT} = −400μA	2.4		V
	All outputs except open collector[5]				
I_{IL}	Input low current	V_{IN} = 0.4V		−400	μA
I_{IH}	Input high current	V_{IN} = 2.4V		20	μA
I_{OC}	Open collector off state current[5]	V_{OUT} = 2.4V		20	μA
I_{SC}	Output short circuit current[6]	V_{CC} = max	−40	−100	mA
I_{CC}	V_{CC} supply current	V_{CC} = max		130	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other conditions other than those indicated in the Electrical Characteristics section of this data sheet is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. Parameters are valid over specified temperature range.
4. All voltage measurements are referenced to ground . For testing, all signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
5. P DOWN is an open collector output.
6. No more than one output should be connected to ground at one time.
7. Capacitive test load is 100pF for all pins.

AC ELECTRICAL CHARACTERISTICS $V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to $+70°C$[3, 4] (see figures 3 – 10)

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS		UNITS
			Min	Max	
1	3	CLK low pulse width	25		ns
2	3	CLK high pulse width	25		ns
3	3	CLK period	50		ns
4	4	READ DATA low to RCLOCK/WCLOCK high		5	ns
5	4	RCLOCK/WCLOCK low to REDAT high		10	ns
6	4	READ DATA high pulse width	10		ns
7	4	READ DATA high to RCLOCK/WCLOCK low	20		ns
8	5	CRYSTAL X1 low to RCLOCK/WCLOCK low		65	ns
9	5	CRYSTAL X1 low to RCLOCK/WCLOCK high		65	ns
10	5	VCO CLK low to RCLOCK/WCLOCK high		30	ns
11	5	VCO CLK high to RCLOCK/WCLOCK low		30	ns
12	6	VCO CLK high to P DOWN low		20	ns
13	6	VCO CLK low to P DOWN three-state		20	ns
14	7	VCO CLK low to P UP three-state		20	ns
15	7	READ DATA high to P UP high		20	ns
16	8	VCO CLK high to FREQ PUMP high		40	ns
17	8	VCO CLK high to FREQ PUMP three-state		40	ns
18	8	CRYSTAL X1 high to FREQ PUMP low		40	ns
19	8	CRYSTAL X1 high to FREQ PUMP three-state		40	ns
20	9	WRCLK high to RCLOCK/WCLOCK low	20		ns
21	9	RCLOCK/WCLOCK low to WRCLK low	20		ns
22	9	WRDAT high to RCLOCK/WCLOCK low	20		ns
23	9	RCLOCK/WCLOCK low to WRDAT low	20		ns
24	9	EARLY SIG high pulse width	20	40	ns
25	9, 10	NOMINAL SIG high to WRITE DATA high		20	ns
26	9	CRYSTAL X1 high to EARLY SIG high		40	ns
27	10	EARLY SIG high to NOMINAL SIG high		30	ns
28	10	EARLY SIG high to LATE SIG high		60	ns
29	10	EARLY SIG high to WRITE DATA high		20	ns
30	10	LATE SIG high to WRITE DATA high		20	ns

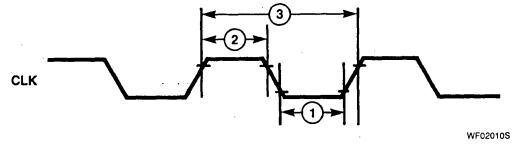


Figure 3. CLK Timing

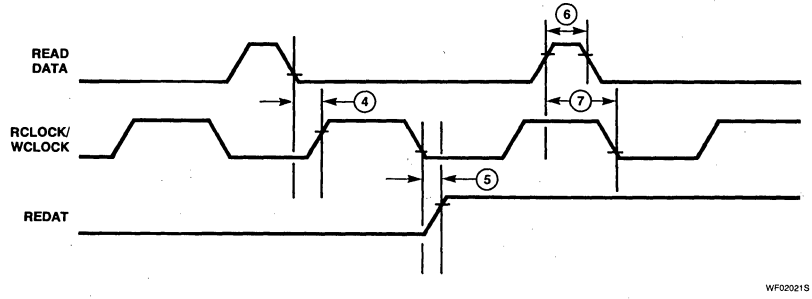


Figure 4. READ DATA/REDAT Timing

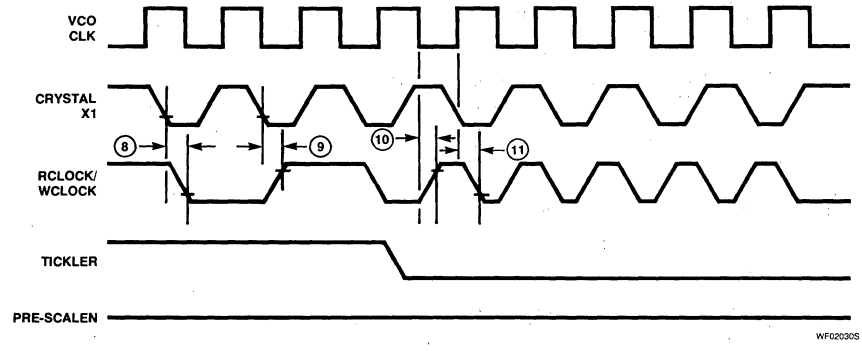


Figure 5. RCLOCK/WCLOCK Timing

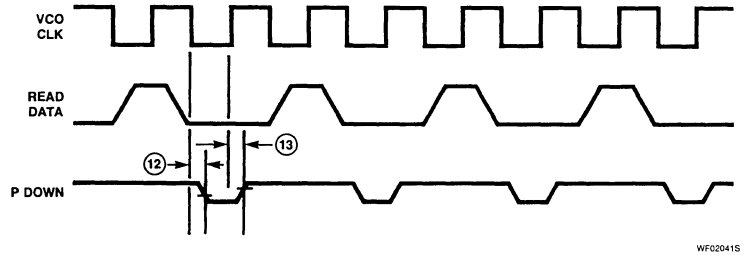


Figure 6. PUMP DOWN Timing

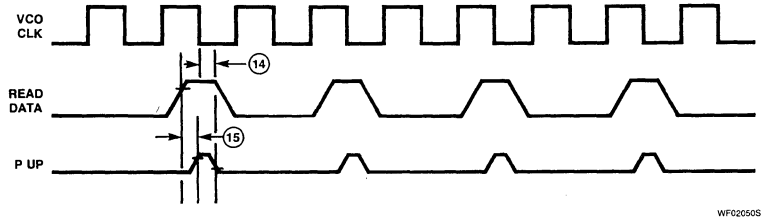


Figure 7. PUMP UP Timing

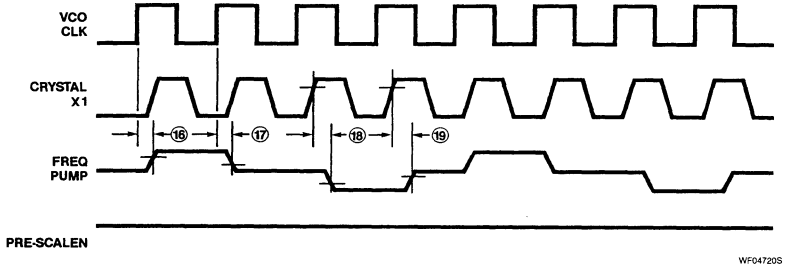


Figure 8. FREQ Pump Timing

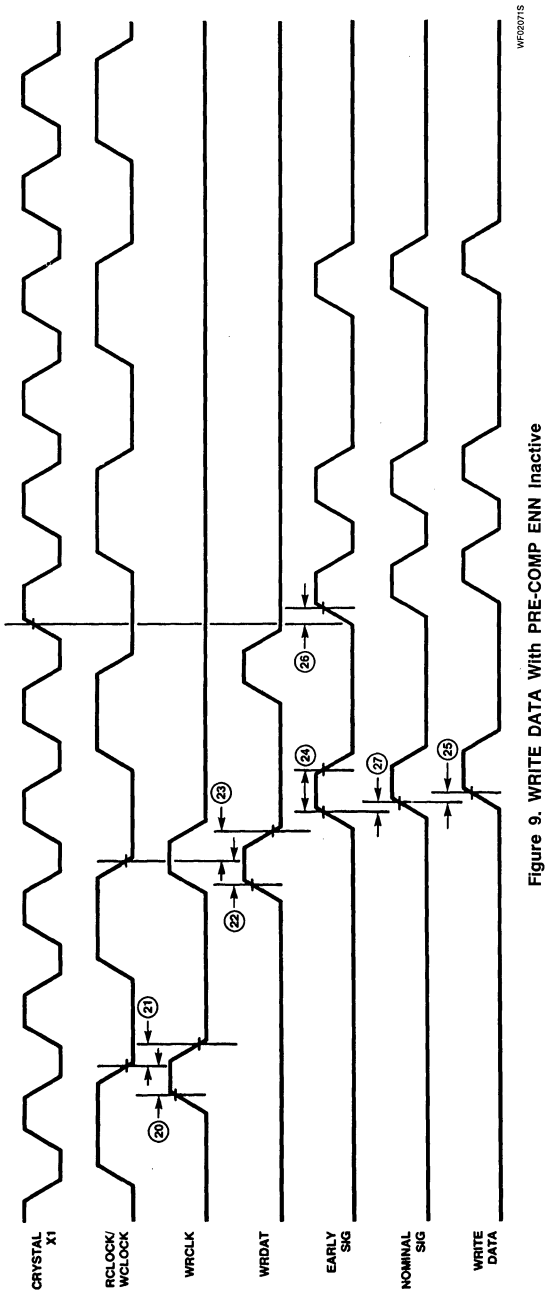


Figure 9. WRITE DATA With PRE-COMP ENN Inactive

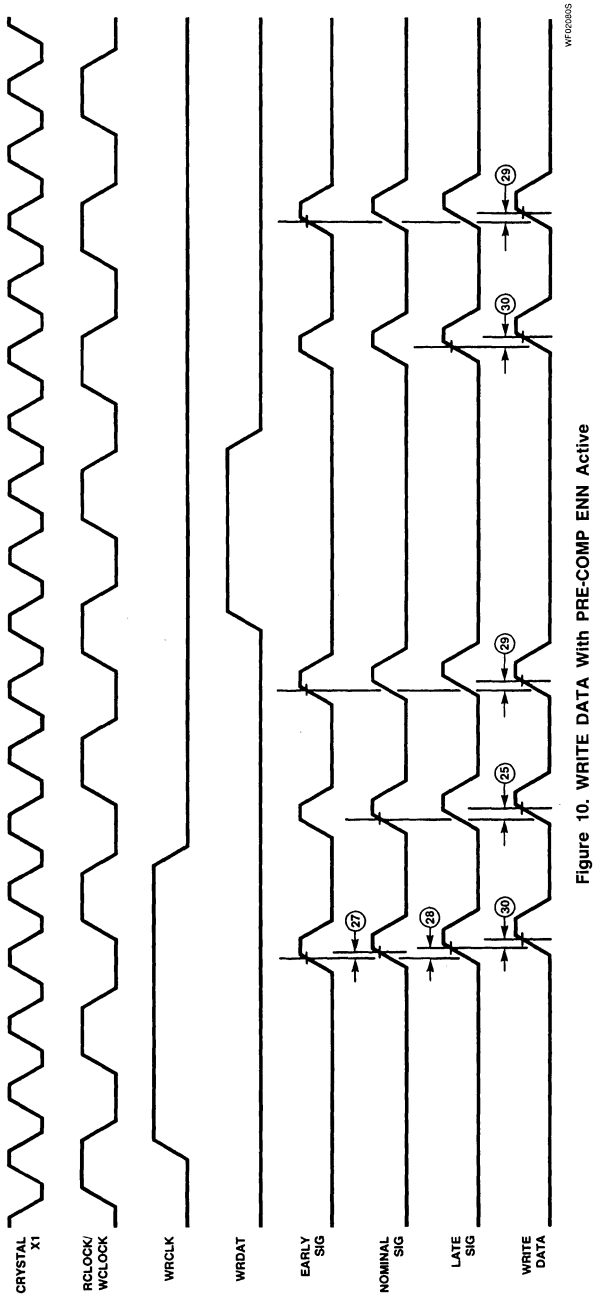


Figure 10. WRITE DATA With PRE-COMP ENN Active

Signetics

SCN68562

Dual Universal Serial Communications Controller (DUSCC)

Microprocessor Products

Preliminary Specification

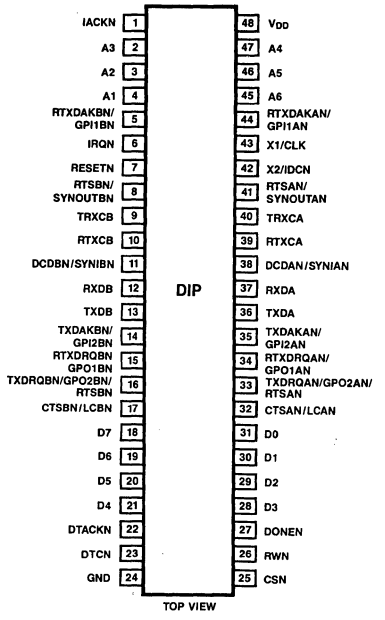
DESCRIPTION

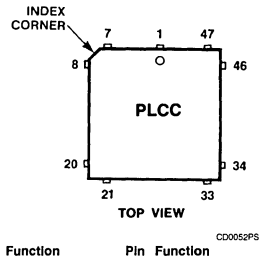
The Signetics SCN68562 Dual Universal Serial Communications Controller (DUSCC) is a single chip MOS-LSI communications device that provides two independent, multi-protocol, full duplex receiver/transmitter channels in a single package. It supports bit oriented and character oriented (byte count and byte control) synchronous data link controls as well as asynchronous protocols. The SCN68562 interfaces to the 68000 MPU via asynchronous bus control signals and is capable of program polled, interrupt driven, block move or DMA data transfers.

The operating mode and data format of each channel can be programmed independently. Each channel consists of a receiver, a transmitter, a 16-bit multifunction counter/timer, a digital phase locked loop (DPLL), a parity/CRC generator and checker, and associated control circuits. The two channels share a common bit rate generator (BRG), operating directly from a crystal or an external clock, which provides sixteen common bit rates simultaneously. The operating rate for the receiver and transmitter of each channel can be independently selected from the BRG, the DPLL, the counter/timer, or from an external 1x or 16x clock, making the DUSCC well suited for dual speed channel applications. Data rates up to 4Mbits are supported.

The transmitter and receiver each contain a four-deep FIFO with appended transmitter command and receiver status bits and a shift register. This permits reading and writing of up to four characters at a time, minimizing the potential of receiver overrun or transmitter underrun, and reducing interrupt or DMA overhead. In addition, a flow control capability is provided to disable a remote transmitter when the FIFO of the local receiving device is full.

PIN CONFIGURATION





Pin	Function	Pin	Function
1	NC	27	NC
2	IACKN	28	CSN
3	A3	29	R/WN
4	A2	30	DONEN
5	A1	31	D3
6	RTXDAKBN/ GPI1BN	32	D2
		33	D1
7	IRQN	34	D0
8	RESETN	35	CTSAN/LCAN
9	RSTBN/ SYNOUTBN	36	TXDRQAN/ GPO2AN/RTSAN
10	TRXCB	37	RTXDRQAN/ GPO1AN
11	RTXCB		
12	DCDBN/SYNIBN	38	TXDAKAN/GPI2AN
13	RXDB	39	TXDA
14	NC	40	NC
15	TXDB	41	RXDA
16	TXDAKBN/ GPI2BN	42	DCDAN/SYNIAN
		43	RTXCA
17	RTXDRQBN/ GPO1BN	44	TRXCA
		45	RSTAN/ SYNOUTAN
18	TXDRQBN/ GPO2BN/RTSBN	46	X2/IDCN
19	CTSBN/LCBN	47	X1/CLK
20	D7	48	RTXDAKAN/ GPI1AN
21	D6		
22	D5	49	A6
23	D4	50	A5
24	DTACKN	51	A4
25	DTCN	52	V_{DD}
26	GND		

Two modem control inputs (DCD and CTS) and three modem control outputs (RTS and two general purpose) are provided. Because the modem control inputs and outputs are general purpose in nature, they can be optionally programmed for other functions.

Dual Universal Serial Communications Controller (DUSCC) SCN68562

FEATURES

General Features

- Dual full-duplex synchronous/asynchronous receiver and transmitter
- Multi-protocol operation
  - BOP: HDLC/ADCCP, SDLC, SDLC loop, X.25 or X.75 link level, etc.
  - COP: BISYNC, DDCMP, X.21
  - ASYNC: 5-8 bits plus optional parity
- Four character receiver and transmitter FIFOs
- 0 to 4MHz data rate
- Programmable bit rate for each receiver and transmitter selectable from:
  - 16 fixed rates: 50 to 38.4K baud
  - One user defined rate derived from programmable counter/timer
  - External 1x or 16x clock
  - Digital phase locked loop
- Parity and FCS (frame check sequence LRC or CRC) generation and checking
- Programmable data encoding/decoding: NRZ, NRZI, FM0, FM1, Manchester
- Programmable channel mode: full/half duplex, auto-echo, or local loopback
- Programmable data transfer mode: polled, interrupt, DMA, wait
- DMA interface
  - Compatible with Signetics SCB68430 Direct Memory Access Interface (DMAI) and other DMA controllers
  - Half or full duplex operation
  - Single or dual address data transfers
  - Automatic frame termination on counter/timer terminal count or DMA DONE
- Interrupt capabilities
  - Daisy chain option
  - Vector output (fixed or modified by status)
  - Programmable internal priorities
  - Maskable interrupt conditions
  - 68K compatible
- Multi-function programmable 16-bit counter/timer
  - Bit rate generator
  - Event counter
  - Count received or transmitted characters
  - Delay generator
  - Automatic bit length measurement
- Modem controls
  - RTS, CTS, DCD, and up to four general purpose I/O pins per channel
  - CTS and DCD programmable auto-enables for TX and RX
  - Programmable interrupt on change of CTS or DCD
- On-chip oscillator for crystal
- TTL compatible
- Single +5V power supply

Asynchronous Mode Features

- Character length: 5 to 8 bits
- Odd or even parity, no parity, or force parity
- Up to two stop bits programmable in 1/16 bit increments
- 1x or 16x RX and TX clock factors
- Parity, overrun, and framing error detection
- False start bit detection
- Start bit search 1/2 bit time after framing error detection
- Break generation with handshake for counting break characters
- Detection of start and end of received break
- Character compare with optional interrupt on match

Character Oriented Protocol Features

- Character length: 5 to 8 bits
- Odd or even parity, no parity, or force parity
- LRC or CRC generation and checking
- Optional opening PAD transmission
- One or two SYN characters
- External sync capability
- SYN detection and optional stripping
- SYN or MARK linefill on underrun
- Idle in MARK or SYNs
- Parity, FCS, overrun, and underrun error detection
- BISYNC Features
  - EBCDIC or ASCII header, text and control messages
  - SYN, DLE stripping
  - EOM (end of message) detection and transmission
  - Auto transparency mode switching
  - Auto hunt after receipt of EOM sequence (with closing PAD check after EOT or NAK)
  - Control character sequence detection for both transparent and normal text

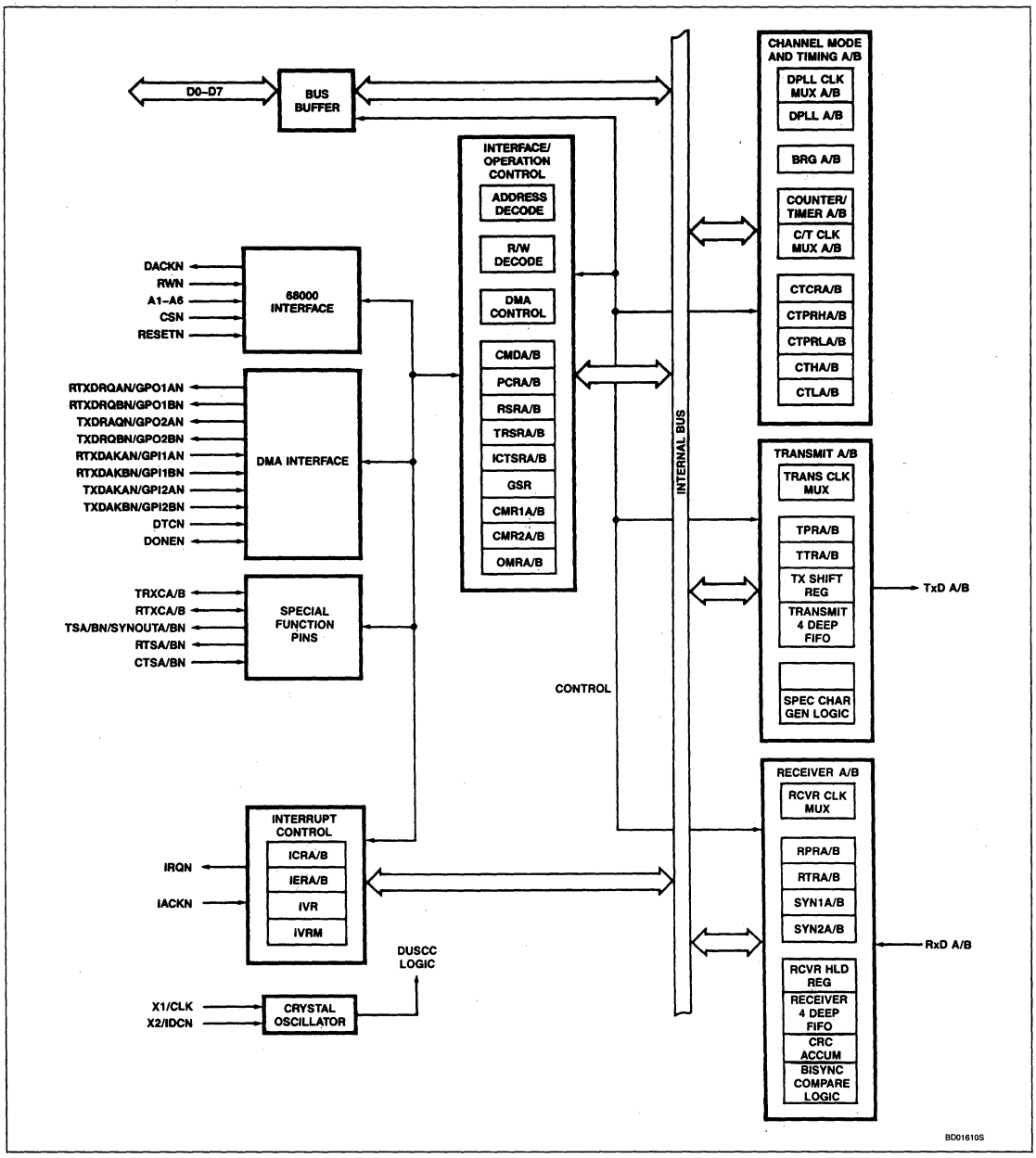
Bit Oriented Protocol Features

- Character length: 5 to 8 bits
- Detection and transmission of residual character: 0-7 bits
- Automatic switch to programmed character length for I field
- Zero insertion and deletion
- Optional opening PAD transmission
- Detection and generation of FLAG, ABORT, and IDLE bit patterns
- Detection and generation of shared (single) FLAG between frames
- Detection of overlapping (shared zero) FLAGs
- ABORT, ABORT-FLAGs, or FCS-FLAGs line fill on underrun
- Idle in MARK or FLAGs
- Secondary address recognition including group and global address
- Single or dual octet secondary address
- Extended address and control fields
- Short frame rejection for receiver
- Detection and notification of received end of message
- CRC generation and checking
- SDLC loop mode capability

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 10%, T_A = 0 to 70°C
Ceramic DIP	SCN68562C4I48
Plastic DIP	SCN68562C4N48
Plastic LCC	SCN68562C4A52

BLOCK DIAGRAM



PIN DESCRIPTION

In this data sheet, signals are discussed using the terms 'active' and 'inactive' or 'asserted' and 'negated' independent of whether the signal is active in the high (logic 1) or low (logic 0) state. N at the end of a pin name signifies the signal associated with the pin is active low (see individual pin description for the definition of the active level of each signal.) Pins which are provided for both channels are designated by either an underline (_) or by A/B after the name of the pin and the active low state indicator, N, if applicable. A similar method is used for registers provided for both channels; these are designated by either an underline or by A/B after the name.

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
A1 – A6	4 – 2, 45 – 47	5 – 3, 51 – 49	I	**Address Lines:** Active high. Address inputs which specify which of the internal registers is accessed for read/write operations.
D0 – D7	31 – 28, 21 – 18	34 – 31, 23 – 20	I/O	**Bidirectional Data Bus:** Active high, three state. Bit 0 is the LSB and bit 7 is the MSB. All data, command, and status transfers between the CPU and the DUSCC take place over this bus. The data bus is enabled when CSN is low, during interrupt acknowledge cycles and single-address DMA acknowledge cycles.
R/WN	26	29	I	**Read/Write:** A high input indicates a read cycle and a low input indicates a write cycle when a cycle is initiated by assertion of the CSN input.
CSN	25	28	I	**Chip Select:** Active low input. When low, data transfers between the CPU and the DUSCC are enabled on D0 – D7 as controlled by the R/WN and A1 – A6 inputs. When CSN is high, the DUSCC is isolated from the data bus (except during interrupt acknowledge cycles and single-address DMA transfers) and D0 – D7 are placed in the three state condition.
DTACKN	22	24	O	**Data Transfer Acknowledge:** Active low, three state. DTACKN is asserted on a write cycle to indicate that the data on the bus has been latched, and on a read cycle or interrupt acknowledge cycle to indicate valid data is on the bus. The signal is negated when completion of the cycle is indicated by negation of the CSN or IACKN input, and returns to the inactive state (three-state) a short period after it is negated. In single address DMA mode, the operation of this pin is similar to the description above. The exception is that it is negated when completion of the cycle is indicated by the assertion of DTCN or negation of DMA acknowledge inputs (whichever occurs first), and returns to the inactive state (three-state) a short period after it is negated. When negated, DTACKN becomes an open drain output and requires an external pull up resistor.
IRQN	6	7	O	**Interrupt Request:** Active low, open drain. This output is asserted upon occurrence of any enabled interrupting condition. The CPU can read the general status register to determine the interrupting condition(s), or can respond with an interrupt acknowledge cycle to cause the DUSCC to output an interrupt vector on the data bus.
IACKN	1	2	I	**Interrupt Acknowledge:** Active low. When IACKN is asserted, the DUSCC responds by placing the contents of the interrupt vector register (modified or unmodified by status) on the data bus and asserting DTACKN. If no active interrupt is pending, DTACKN is not asserted.
X1/CLK	43	47	I	**Crystal or External Clock:** When using the crystal oscillator, the crystal is connected between pins X1 and X2. If a crystal is not used, an external clock is supplied at this input. This clock is used to drive the internal bit rate generator, as an optional input to the counter/timer or DPLL, and to provide other required clocking signals.
X2/IDCN	42	46	O	**Crystal or Interrupt Daisy Chain:** Active low. When a crystal is used as the timing source, the crystal is connected between pins X1 and X2. This pin can be programmed to provide an interrupt daisy chain output which propagates the IACKN signal to lower priority devices, if no active interrupt is pending.
RESETN	7	8	I	**Master Reset:** Active low. A low on this pin resets the transmitters and receivers and resets the registers shown in table 1. Reset is asynchronous, i.e., no clock is required.
RXDA,RXDB	37, 12	41, 13	I	**Channel A (B) Receiver Serial Data Input:** The least significant bit is received first. If external receiver clock is specified for the channel, the input is sampled on the rising edge of the clock.
TXDA,TXDB	36, 13	39, 15	O	**Channel A (B) Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the marking (high) condition when the transmitter is disabled or when the channel is operating in local loopback mode. If external transmitter clock is specified for the channel, the data is shifted on the falling edge of the clock.
RTXCA,RTXCB	39, 10	43, 11	I/O	**Channel A (B) Receiver/Transmitter Clock:** As an input, it can be programmed to supply the receiver, transmitter, counter/timer, or DPLL clock. As an output, can supply the counter/timer output, the transmitter shift clock (1X), or the receiver sampling clock (1X). The maximum external receiver/transmitter clock frequency is 4MHz.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
TRXCA,TRXCB	40, 9	44, 10	I/O	**Channel A (B) Transmitter/Receiver Clock:** As an input, it can supply the receiver, transmitter, counter/timer, or DPLL clock. As an output, it can supply the counter/timer output, the DPLL output, the transmitter shift clock (1X), the receiver sampling clock (1X), the transmitter BRG clock (16X), the receiver BRG clock (16X), or the internal system clock (X1/2). The maximum external receiver/transmitter clock frequency is 4MHz.
CTSA/BN, LCA/BN	32, 17	35, 19	I/O	**Channel A (B) Clear-To-Send Input or Loop Control Output:** Active low. The signal can be programmed to act as an enable for the transmitter when not in loop mode. The DUSCC detects logic level transitions on this input and can be programmed to generate an interrupt when a transition occurs. When operating in the BOP loop mode, this pin becomes a loop control output which is asserted and negated by DUSCC commands. This output provides the means of controlling external loop interface hardware to go on-line and off-line without disturbing operation of the loop.
DCDA/BN, SYNIA/BN	38, 11	42, 12	I	**Channel A (B) Data-Carrier-Detected or External Sync Input:** The function of this pin is programmable. As a DCD active low input, it acts as an enable for the receiver or can be used as a general purpose input. For the DCD function, the DUSCC detects logic level transitions on this input and can be programmed to generate an interrupt when a transition occurs. As an active low external sync input, it is used in COP modes to obtain character synchronization without receipt of a SYN or FLAG character. This mode can be used in disc or tape controller applications or for the optional byte timing lead in X.21.
RTXDRQA/BN, GPO1A/BN	34, 15	37, 17	O	**Channel A (B) Receiver/Transmitter DMA Service Request or General Purpose Output:** Active low. For half-duplex DMA operation, this output indicates to the DMA controller that one or more characters are available in the receiver FIFO (when the receiver is enabled) or that the transmit FIFO is not full (when the transmitter is enabled). For full-duplex DMA operation, this output indicates to the DMA controller that data is available in the receiver FIFO. In non-DMA mode, this pin is a general purpose output that can be asserted and negated under program control.
TXDRQA/BN, GPO2A/BN, RTSA/BN	33, 16	36, 18	O	**Channel A (B) Transmitter DMA Service Request, General Purpose Output, or Request-to-Send:** Active low. For full-duplex DMA operation, this output indicates to the DMA controller that the transmit FIFO is not full and can accept more data. When not in full-duplex DMA mode, this pin can be programmed as a general purpose or a Request-to-Send output, which can be asserted and negated under program control (see Detailed Operation).
RTXDAKA/BN, GPI1A/BN	44, 5	48, 6	I	**Channel A (B) Receiver/Transmitter DMA Acknowledge or General Purpose Input:** Active low. For half-duplex single address DMA operation, this input indicates to the DUSCC that the DMA controller has acquired the bus and that the requested bus cycle (read receiver FIFO or load transmitter FIFO) is beginning. For full-duplex single address DMA operation, this input indicates to the DUSCC that the DMA controller has acquired the bus and that the requested read receiver FIFO bus cycle is beginning. Because the state of this input can be read under program control, it can be used as a general purpose input when not in single address DMA mode.
TXDAKA/BN, GPI2A/BN	35, 14	38, 16	I	**Channel A (B) Transmitter DMA Acknowledge or General Purpose Input:** Active low. When the channel is programmed for full-duplex single address DMA operation, this input is asserted to indicate to the DUSCC that the DMA controller has acquired the bus and that the requested load transmitter FIFO bus cycle is beginning. Because the state of this input can be read under program control, it can be used as a general purpose input when not in full duplex single address DMA mode.
DTCN	23	25	I	**Device Transfer Complete:** Active low. DTCN is asserted by the DMA controller to indicate that the requested data transfer is complete.
DONEN	27	30	I/O	**Done:** Active low, open drain. See Detailed Operation for a description of the function of this pin.
RTSA/BN, SYNOUTA/BN	41, 8	45, 9	O	**Channel A (B) Sync Detect or Request-to-Send:** Active low. If programmed as a sync output, it is asserted one bit time after the specified sync character (COP or BISYNC modes) or a FLAG (BOP modes) is detected by the receiver. As a Request-to-Send modem control signal, it functions as described previously for the TXDRQ_N/RTS_N pin and, alternatively, it can be programmed to perform the RTS function in full duplex single address DMA mode.
V_{DD}	48	52	I	+5V ±10% power input.
GND	24	26	I	Signal and power ground input.

REGISTERS

The addressable registers of the DUSCC are shown in table 1. The following rules apply to all registers:

1. A read from a reserved location in the map results in a read from the 'null register'. The null register returns all ones for data and results in a normal bus cycle. A write to one of these locations results in a normal bus cycle without a write being performed.
2. Unused bits of a defined register are read as zeros, unless ones have been loaded after master reset.
3. Bits that are unused in the chosen mode but are used in others are readable and writable but their contents are ignored in the chosen mode.
4. All registers are addressable as 8-bit quantities. To facilitate operation with the 68K MOVEP instruction, addresses are ordered such that certain sets of registers may also be accessed as words or long words.

The operation of the DUSCC is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The contents of certain control registers are initialized on RESET (set to zero). Care should be exercised if the contents of a register are changed during operation, since certain changes may cause operational problems, e.g., changing the channel mode at an inappropriate time may cause the reception or transmission of an incorrect character. In general, the contents of registers which control transmitter or receiver operation, or the counter/timer, should be changed only when they are not enabled.

The DUSCC registers can be separated into five groups to facilitate their usage:

1. Channel mode configuration and pin description registers
2. Transmitter and receiver parameter and timing registers
3. Counter/timer control and value registers
4. Interrupt control and status registers
5. Command register

This arrangement is used in the following description of the DUSCC registers.

Channel Mode Configuration and Pin Description Registers

There are five registers in this group for each channel. The bit format for each of these registers is contained in table 2. The primary function of these registers is to define configuration of the channels and the function of the programmable pins.

Channel Mode Register 1 (CMR1A, CMR1B)

[7:6] Data Encoding — These bits select the data encoding for the received and transmitted data:

00 If the DPLL is set to NRZI mode (see DPLL commands), it selects positive logic (1 = high, 0 = low). If the DPLL is set to FM mode (see DPLL commands), Manchester (bi-phase level) encoding is selected.

01 NRZI. Non-return-to-zero inverted.

10 FM0. Bi-phase space.

11 FM1. Bi-phase mark.

[5] Extended Control (BOP) —

0 No. A one-octet control field follows the address field.

1 Yes. A two-octet control field follows the address field.

[5] Parity (COP/ASYNC), Code Select (BISYNC) —

0 Even parity if with parity is selected by [4:3] or a 0 in the parity bit position if force parity is selected by [4:3]. In BISYNC protocol mode, internal character comparisons are made using EBCDIC coding.

1 Odd parity if with parity is selected by [4:3] or a 1 in the parity bit position if force parity is selected by [4:3]. In BISYNC protocol mode, internal character comparisons are made using 8-bit ASCII coding.

[4:3] Address Mode (BOP) — This field controls whether a single octet or multiple octets follow the opening FLAG(s) for both the receiver and the transmitter. This field is activated by selection of BOP secondary mode through the channel protocol mode bits CMR1_[2:0] (see Detailed Operation).

00 Single octet address.
01 Extended address.
10 Dual octet address.
11 Dual octet address with group.

[4:3] Parity Mode (COP/ASYNC) — This field selects the parity mode for both the receiver and the transmitter. A parity bit is added to the programmed character length if with parity or force parity is selected:

00 No parity. Required when BISYNC protocol mode is programmed.
01 Reserved.
10 With parity. Odd or even parity is selected by [5].
11 Force parity. The parity bit is forced to the state selected by [5].

[2:0] Channel Protocol Mode — This field selects the operational protocol and submode for both the receiver and transmitter:

000 BOP Primary. No address comparison is performed. For receive, all characters received after the opening FLAG(s) are transferred to the FIFO.

001 BOP Secondary. This mode activates the address modes selected by [4:3]. Except in the case of extended address ([4:3]=01), an address comparison is performed to determine if a frame should be received. Refer to Detailed Operation for details of the various addressing modes. If a valid comparison occurs, the receiver is activated and the address octets and all subsequent received characters of the frame are transferred to the receive FIFO.

010 BOP Loop. The DUSCC acts as a secondary station in a loop. The GO-ON-LOOP and GO-OFF-LOOP commands are used to cause the DUSCC to go on and off the loop. Normally, the TXD output echoes the RXD input with a three bit time delay. If the transmitter is enabled and the 'go active on poll' command has been asserted, the transmitter will begin sending when an EOP sequence consisting of a zero followed by seven ones is detected. The DUSCC changes the last one of the EOP to zero, making it another FLAG, and then operates as described in the detailed operation section. The loop sending status bit (TRSR[6]) is asserted concurrent with the beginning of transmission. The frame should normally be terminated with an EOM followed by an echo of the marking RXD line so that secondary stations further down the loop can append their messages to the messages from up-loop stations by the same process. If the 'go active on poll' command is not asserted, the transmitter remains inactive (other than echoing the received data) even when the EOP sequence is received.

011 BOP Loop without address comparison. Same as normal loop mode except that address field comparisons are disabled. All received frames are transmitted to the CPU.

100 COP Dual SYN. Character sync is achieved upon receipt of a bit sequence matching the contents of the appropriate bits of S1R and S2R (SYN1-SYN2), including parity bits if any.

101 COP Dual SYN (BISYNC). Character sync is achieved upon receipt of a bit sequence matching the contents of the appropriate bits of S1R and S2R

Table 1. DUSCC REGISTER ADDRESS MAP

ADDRESS BITS[1] 6 5 4 3 2 1	ACRONYM	REGISTER NAME	MODE	AFFECTED BY RESET
c 0 0 0 0 0	CMR1	Channel Mode Register 1	R/W	Yes – 00
c 0 0 0 0 1	CMR2	Channel Mode Register 2	R/W	Yes – 00
c 0 0 0 1 0	S1R	SYN 1/Secondary Address 1 Register	R/W	No
c 0 0 0 1 1	S2R	SYN 2/Secondary Address 2 Register	R/W	No
c 0 0 1 0 0	TPR	Transmitter Parameter Register	R/W	Yes – 00
c 0 0 1 0 1	TTR	Transmitter Timing Register	R/W	No
c 0 0 1 1 0	RPR	Receiver Parameter Register	R/W	Yes – 00
c 0 0 1 1 1	RTR	Receiver Timing Register	R/W	No
c 0 1 0 0 0	CTPRH	Counter/Timer Preset Register High	R/W	No
c 0 1 0 0 1	CTPRL	Counter/Timer Preset Register Low	R/W	No
c 0 1 0 1 0	CTCR	Counter/Timer Control Register	R/W	No
c 0 1 0 1 1	OMR	Output and Miscellaneous Register	R/W	Yes – 00
c 0 1 1 0 0	CTH	Counter/Timer High	R	No
c 0 1 1 0 1	CTL	Counter/Timer Low	R	No
c 0 1 1 1 0	PCR	Pin Configuration Register	R/W	Yes – 00
c 0 1 1 1 1	CCR	Channel Command Register	R/W	No
c 1 0 0 X X	TXFIFO	Transmitter FIFO	W	No
c 1 0 1 X X	RXFIFO	Receiver FIFO	R	No
c 1 1 0 0 0	RSR	Receiver Status Register	R/W[2]	Yes – 00
c 1 1 0 0 1	TRSR	Transmitter and Receiver Status Register	R/W[2]	Yes – 00
c 1 1 0 1 0	ICTSR	Input and Counter/Timer Status Register	R/W[2]	Yes
d 1 1 0 1 1	GSR	General Status Register	R/W[2]	Yes – 00
c 1 1 1 0 0	IER	Interrupt Enable Register	R/W	Yes – 00
c 1 1 1 0 1		Not used		
0 1 1 1 1 0	IVR	Interrupt Vector Register – Unmodified	R/W	Yes – 0F
1 1 1 1 1 0	IVRM	Interrupt Vector Register – Modified	R	Yes – 0F
0 1 1 1 1 1	ICR	Interrupt Control Register	R/W	Yes – 00
1 1 1 1 1 1		Not used		

NOTES:

1. c = 0 for channel A, c = 1 for channel B.
   d = don't care – register may be accessed as either channel.
   x = don't care – FIFOs are addressable at any of four adjacent addresses to allow them to be addressed as byte/word/long word with the 68K MOVEP instruction.
2. A write to this register may perform a status resetting operation.

(SYN1-SYN2). In this mode, special transmitter and receive logic is activated. Transmitter and receiver character length must be programmed to 8 bits and no parity (see Detailed Operation).

110 COP Single SYN. Character sync is achieved upon receipt of a bit sequence matching the contents of the appropriate bits of S1R (SYN1), including parity bit if any. This mode is required when the external sync mode is selected (see description of RPR[4], BOP/COP).

111 Asynchronous. Start/stop format.

Channel Mode Register 2 (CMR2A, CMR2B)

[7:6] Channel Connection — This field selects the mode of operation of the channel. The user must exercise care when switching into and out of the various modes. The selected mode will be activated immediately upon mode selection, even if this occurs in the middle of a received or transmitted character.

00 Normal mode. The transmitter and receiver operate independently in either half or full-duplex, controlled by the respective enable commands.

01 Automatic echo mode. Automatically retransmits the received data with a half-bit time delay (ASYNC, 16X clock mode) or a one-bit time delay (all other modes). The following conditions are true while in automatic echo mode:

1. Received data is reclocked and retransmitted on the TXD output.
2. The receiver clock is used for the transmitter.
3. The receiver must be enabled, but the transmitter need not be enabled.
4. The TXRDY and underrun status bits are inactive.
5. The received parity and/or FCS are checked if required, but are not regenerated for transmission, i.e., transmitted parity and/or FCS are as received.
6. In ASYNC mode, character framing is checked, but the stop bits are retransmitted as received. A received break is echoed as received.
7. CPU to receiver communication continues normally, but the CPU to transmitter link is disabled.

10 Local loopback mode. In this mode:

1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver if NRZI or NRZ encoding is used. For FM or Manchester encoding because the receiver clock is derived from the DPLL,

the DPLL source clock must be maintained.

3. The TXD output is held high.
4. The RXD input is ignored.
5. The receiver and transmitter must be enabled.
6. CPU to transmitter and receiver communications continue normally.

11 Reserved.

[5:3] Data Transfer Interface — This field specifies the type of data transfer between the DUSCC's RX and TX FIFOs and the CPU. All interrupt and status functions operate normally regardless of the data transfer interface programmed. Refer to Detailed Operation for details of the various DMA transfer interfaces.

000 Half duplex single address DMA.

001 Half duplex dual address DMA.

010 Full duplex single address DMA.

011 Full duplex dual address DMA.

100 Wait on receive only. In this mode a read of a non-empty receive FIFO results in a normal bus cycle. However, if the receive FIFO of the channel is empty when a read RX FIFO cycle is initiated, the DTACKN output remains negated until a character is received and loaded into the FIFO. DTACKN is then asserted and the cycle is completed normally.

101 Wait on transmit only. In this mode a write to a non-full transmit FIFO results in a normal bus cycle. However, if the transmit FIFO of the channel is full when a write TX FIFO cycle is initiated, the DTACKN output remains negated until a FIFO position becomes available for the new character. DTACKN is then asserted and the cycle is completed normally.

110 Wait on transmit and receive. As above for both wait on receive and transmit operations.

111 Polled or interrupt. DMA and wait functions of the channel are not activated. Data transfers to the RX and TX FIFOs are via normal bus read and write cycles in response to polling of the status registers and/or interrupts.

[2:0] Frame Check Sequence Select — This field selects the optional frame check sequence (FCS) to be appended at the end of a transmitted frame. When CRC is selected in COP, then no parity and 8-bit character length must be used. The selected FCS is transmitted as follows:

1. Following the transmission of a FIFO'ed character tagged with the 'send EOM' command.
2. If underrun control (TPR[7:6]) is programmed for TEOM, upon occurrence of an underrun.
3. If TEOM on zero count or done (TPR[4]) is asserted and the counter/timer is counting transmitted characters, after transmission of the character which causes the counter to reach zero count.
4. In DMA mode with TEOM on zero count or done (TPR[4]) set, after transmission of a character if DONEN is asserted when that character was loaded into the TX FIFO by the DMA controller.

000 No frame check sequence.

001 Reserved

010 LRC8: Divisor = x^8+1, dividend preset to zeros. The TX sends the calculated LRC non-inverted. The RX indicates an error if the computed LRC is not equal to 0. Valid for COP modes only.

011 LRC8: Divisor = x^8+1, dividend preset to ones. The TX sends the calculated LRC non-inverted. The RX indicates an error if the computed LRC is not equal to 0. Valid for COP modes only.

100 CRC16: Divisor = $x^{16}+x^{15}+x^2+1$, dividend preset to zeros. The TX sends the calculated CRC non-inverted. The RX indicates an error if the computed CRC is not equal to 0. Not valid for ASYNC mode.

101 CRC16: Divisor = $x^{16}+x^{15}+x^2+1$, dividend preset to ones. The TX sends the calculated CRC non-inverted. The RX indicates an error if the computed CRC is not equal to 0. Not valid for ASYNC mode.

110 CRC-CCITT: Divisor = $x^{16}+x^{12}+x^5+1$, dividend preset to zeros. The TX sends the calculated CRC non-inverted. The RX indicates an error if the computed CRC is not equal to 0. Not valid for ASYNC mode.

111 CRC-CCITT: Divisor = $x^{16}+x^{12}+x^5+1$, dividend preset to ones. The TX sends the calculated CRC inverted. The RX indicates an error if the computed CRC is not equal to H'F0B8'. Not valid for ASYNC mode.

SYN1/Secondary Address 1 Register (S1RA, S1RB)

[7:0] Character Compare — In ASYNC mode this register holds a 5 – 8 bit long bit pattern which is compared with received characters. If a match occurs, the character compare status bit (RSR[7]) is set. This field is ignored if the receiver is in a break condition.

In COP modes, this register contains the 5 to 8 bit SYN1 bit pattern, right justified. Parity bit need not be included in the value placed in the register even if parity is specified in CMR1[4:3]. However, a character received with parity error, when parity is specified, will not match. In ASYNC or COP modes, if parity is specified, then any unused bits in this register must be programmed to zeros. In BOP secondary mode it contains the address used to compare the first received address octet. The register is not used in BOP primary mode or secondary modes where address comparisons are not made, such as when extended addressing is specified.

SYN2/Secondary Address 2 Register (S2RA, S2RB)

[7:0] — This register is not used in ASYNC, COP single SYN, BOP primary modes, BOP secondary modes with single address field, and BOP secondary modes where address comparisons are not made, such as when extended addressing is specified.

In COP dual SYN modes, it contains the 5 to 8 bit SYN2 bit pattern, right justified. Parity bit need not be included in the value placed in the register even if parity is specified in CMR1[4:3]. However, a character received with parity error, when parity is specified, will not match. If parity is specified, then any unused bits in this register must be programmed to zeros. In BOP secondary mode using two address octets, it contains the partial address used to compare the second received address octet.

Pin Configuration Register (PCRA, PCRB)

This register selects the functions for multipurpose I/O pins.

[7] X2/IDC — This bit is defined only for PCRA. It is not used in PCRB.

0 The X2/IDCN pin is used as a crystal connection.

1 The X2/IDCN pin is the interrupt daisy chain output.

[6] GPO2/RTS — The function of this pin is programmable only when not operating in full duplex DMA mode.

0 The TXDRQ_N/GPO2_N/RTS_N pin is a general purpose output. It is low when OMR[2] is a 1 and high when OMR[2] is 0.

1 The pin is a request-to-send output (see Detailed Operation).

Table 2. CHANNEL CONFIGURATION/PIN DEFINITION REGISTERS BIT FORMATS

CHANNEL MODE REG 1

(CMR1A, CMR1B)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Data Encoding		**Extended Control**	**Address Mode (BOP)**		**Channel Protocol Mode**		
00 – NRZ/Manchester 01 – NRZI 10 – FM0 11 – FM1		BOP only 0 – no 1 – yes	00 – 8 bit 01 – extended address 10 – 16 bit 11 – 16 bit w/group		000 – BOP primary 001 – BOP secondary 010 – BOP loop 011 – BOP loop – no adr. comp.		
		#Parity	**Parity Mode (COP/ASYNC)**		100 – COP dual SYN 101 – COP dual SYN (BISYNC) 110 – COP single SYN 111 – asynchronous		
		0 – even 1 – odd	00 – no parity 01 – reserved 10 – with parity 11 – force parity				

#In BISYNC protocol mode, 0 = EBCDIC, 1 = 8-bit ASCII coding.

CHANNEL MODE REG 2

(CMR2A, CMR2B)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Channel Connection		**Data Transfer Interface**			**Frame Check Sequence Select**		
00 – normal 01 – auto echo 10 – local loop 11 – reserved		000 – half duplex single address DMA 001 – half duplex dual address DMA 010 – full duplex single address DMA 011 – full duplex dual address DMA 100 – wait on RX only 101 – wait on TX only 110 – wait on RX or TX 111 – polled or interrupt			000 – none 001 – reserved 010 – LRC8 preset 0s 011 – LRC8 preset 1s 100 – CRC 16 preset 0s 101 – CRC 16 preset 1s 110 – CRC CCITT preset 0s 111 – CRC CCITT preset 1s		

SYN1/SECONDARY ADDRESS REG 1

(S1RA, S1RB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ASYNC – Character compare (5 – 8 bits) COP – SYN1 (5 – 8 bits) BOP – First address octet							

SYN2/SECONDARY ADDRESS REG 2

(S2RA, S2RB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ASYNC – not used COP – SYN2 (5 – 8 bits) BOP – Second address octet							

PIN CONFIGURATION REG

(PCRA, PCRB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
X2/IDC	**GPO2/RTS**	**SYNOUT/RTS**	**RTXC Pin**		**TRXC Pin**		
* 0 – X2 1 – IDC	0 – GPO2 1 – RTS	0 – SYNOUT 1 – RTS	00 – input 01 – C/T 10 – TXCLK 1x 11 – RXCLK 1x		000 – input 001 – XTAL/2 010 – DPLL 011 – C/T	100 – TXCLK 16x 101 – RXCLK 16x 110 – TXCLK 1x 111 – RXCLK 1x	

*PCRA only. Not used in PCRB.

[5] SYNOUT/RTS —

0 The SYNOUT_N/RTS_N pin is an active low output which is asserted one bit time after a SYN pattern (COP modes) in HSRH/HSRL or FLAG (BOP modes) is detected in CCSR. The output remains asserted for one receiver clock period. See figure 1 for receiver data path.

1 The pin is a request-to-send output (see Detailed Operation). The logical state of the pin is controlled by OMR[0], when set the output is zero.

[4:3] RTXC —

00 The pin is an input. It must be programmed for input when used as the input for the receiver or transmitter clock, the DPLL, or the C/T.

01 The pin is an output from the counter/timer. Refer to CTCRA/B description.

10 The pin is an output from the transmitter shift register clock.

11 The pin is an output from the receiver shift register clock.

2

Table 3. TRANSMITTER AND RECEIVER PARAMETER AND TIMING REGISTER BIT FORMAT

TRANSMITTER PARAMETER REG

(TPRA, TPRB)	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	Underrun Control		**Idle**	**TEOM On Zero Cnt Or Done**	**TX RTS Control**	**CTS Enable TX**	**TX Character Length**	
COP	00 – FCS-idle 01 – reserved 10 – MARKs 11 – SYNs		0 – MARKs 1 – SYNs	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes	00 – 5 bits 01 – 6 bits 10 – 7 bits 11 – 8 bits	
	Underrun Control		**Idle**	**TEOM On Zero Cnt Or Done**				
BOP	00 – FCS-FLAG-idle 01 – reserved 10 – ABORT-MARKs 11 – ABORT-FLAGs		0 – MARKs 1 – FLAGs	0 – no 1 – yes				
	Stop Bits Per Character							
ASYNC	9/16 to 1, 17/16 to 1.5, 25/16 to 2 programmable in 1/16 bit increments							

TRANSMITTER TIMING REG

(TTRA, TTRB)	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	External Source	**Transmitter Clock Select**			**Bit Rate Select**			
	0 – RTXC 1 – TRXC	000 – 1x external 001 – 16x external 010 – DPLL 011 – BRG 100 – 2x other channel C/T 101 – 32x other channel C/T 110 – 2x own channel C/T 111 – 32x own channel C/T			one of sixteen rates from BRG			

RECEIVER PARAMETER REG

(RPRA, RPRB)	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	not used	not used	not used	**RX RTS Control**	**Strip Parity**	**DCD Enable RX**	**RX Character Length**	
ASYNC				0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes	00 – 5 bits 01 – 6 bits 10 – 7 bits 11 – 8 bits	
	SYN Strip	**FCS to FIFO**	**Auto Hunt & Pad Chk**	**Ext Sync**	**Strip Parity**			
COP	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes			
	not used	**FCS to FIFO**	**Overrun Mode**	not used	**All Party Address**			
BOP		0 – no 1 – yes	0 – hunt 1 – cont		0 – no 1 – yes			

Table 3. TRANSMITTER AND RECEIVER PARAMETER AND TIMING REGISTER BIT FORMAT (Continued)

RECEIVER TIMING REG

(RTRA, RTRB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
External Source	Receiver Clock Select			Bit Rate Select			
0 – RTXC 1 – TRXC	000 – 1x external 001 – 16x external 010 – BRG 011 – C/T of channel 100 – DPLL, source = 64x X1/CLK 101 – DPLL, source = 32x External 110 – DPLL, source = 32x BRG 111 – DPLL, source = 32x C/T		(ASYNC protocol mode only)	one of sixteen rates from BRG			

OUTPUT AND MISC REG

(OMRA, OMRB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
TX Residual Character Length			TXRDY Activate	RXRDY Activate	OUT 2	OUT 1	RTS
000 – 1 bit 001 – 2 bits 010 – 3 bits 011 – 4 bits 100 – 5 bits 101 – 6 bits 110 – 7 bits 111 – same as TPR[1:0]			0 – FIFO not full 1 – FIFO empty	0 – FIFO not empty 1 – FIFO full	0 – 0 1 – 1	0 – 0 1 – 1	0 – 0 1 – 1

[2:0] TRXC —

000 The pin is an input. It must be programmed for input when used as the input for the receiver or transmitter clock, the DPLL, or the C/T.

001 The pin is an output from the crystal oscillator. (XTAL/2)

010 The pin is an output from the DPLL output clock.

011 The pin is an output from the counter/timer. Refer to CTCRA/B description.

100 The pin is an output from the selected transmitter BRG frequency divided by two.

101 The pin is an output from the selected receiver BRG frequency divided by two.

110 The pin is an output from the transmitter shift register clock.

111 The pin is an output from the receiver shift register clock.

Transmitter and Receiver Parameter and Timing Registers

This set of five registers contains the information which controls the operation of the transmitter and receiver for each channel. Table 3 shows the bit map format for each of these registers. The registers of this group are:

1. Transmitter parameter and timing registers (TPRA/B and TTRA/B)
2. Receiver parameter and timing registers (RPRA/B and RTRA/B)
3. Output and miscellaneous register (OMRA/B)

The first and second group of registers define the transmitter and receiver parameters and timing. Included in the receiver timing registers are the programming parameters for the DPLL. The last register of the group, OMR contains additional transmitter and receiver information and controls the logical state of the output pins when they are not used as a part of the channel configuration.

Transmitter Parameter Register (TPRA, TPRB)

[7:6] Underrun Control — In BOP and COP modes, this field selects the transmitter response in the event of an underrun (i.e., the TX FIFO is empty).

00 Normal end of message termination. In BOP, the transmitter sends the FCS (if selected by CMR2[2:0]) followed by a FLAG and then either MARKs or FLAGs, as specified by [5]. In COP, the transmitter sends the FCS (if selected by CMR2[2:0]) and then either MARKs or SYNs, as specified by [5].

01 Reserved.

10 In BOP, the transmitter sends an ABORT (11111111) and then places the TXD output in a marking condition until receipt of further instructions. In COP, the transmitter places the TXD output in a marking condition until receipt of further instructions.

11 In BOP, the transmitter sends an ABORT (11111111) and then sends FLAGs until receipt of further instructions. In COP, the transmitter sends SYNs until receipt of further instructions.

[5] Idle — In BOP and COP modes, this bit selects the transmitter output during idle. Idle is defined as the state following a normal end of message until receipt of the next transmitter command.

0 Idle in marking condition.

1 Idle sending SYNs (COP) or FLAGs (BOP).

[4] Transmit EOM on Zero Count or Done — In BOP and COP modes, the assertion of this bit causes the end of message (FCS in COP, FCS-FLAG in BOP) to be transmitted upon the following events:

1. If the counter/timer is counting transmitted characters, after transmission of the character which causes the counter to reach zero count. (DONEN is also asserted as an output if the channel is in a DMA operation.)
2. If the channel is operating in DMA mode, after transmission of a character if DONEN was asserted when that character was loaded into the TX FIFO by the DMA controller.

[7:4] Stop Bits per Character — In ASYNC mode, this field programs the length of the stop bit appended to the transmitted character as shown in table 4.

Stop bit lengths of 9/16 to 1 and 1-9/16 to 2 bits, in increments of 1/16 bit, can be programmed for character lengths of 6, 7, and 8 bits. For a character length of 5 bits, 1-1/16 to 2 stop bits can be programmed in increments of 1/16 bit. The receiver only checks

Table 4. STOP BITS – TRANSMITTED CHARACTER

[7:4]	5 BITS/ CHAR	6, 7, or 8 BITS/CHAR
0000	1.063	0.563
0001	1.125	0.625
0010	1.188	0.688
0011	1.250	0.750
0100	1.313	0.813
0101	1.375	0.875
0110	1.438	0.938
0111	1.500	1.000
1000	1.563	1.563
1001	1.625	1.625
1010	1.688	1.688
1011	1.750	1.750
1100	1.813	1.813
1101	1.875	1.875
1110	1.938	1.938
1111	2.000	2.000

for a 'mark' condition at the center of the first stop bit position (one bit time after the last data bit, or after the parity bit if parity is enabled) in all cases.

If an external 1X clock is used for the transmitter, [7] = 0 selects one stop bit and [7] = 1 selects two stop bits to be transmitted. If Manchester, NRZI, or FM data encoding is selected, only integral stop bit lengths should be used.

[3] Transmitter Request-to-Send Control — This bit controls the deactivation of the RTS_N output by the transmitter (see Detailed Operation).

0 RTS_N is not affected by status of transmitter.

1 RTS_N changes state as a function of transmitter status.

[2] Clear-to-Send Enable Transmitter — The state of this bit determines if the CTS_N input controls the operation of the channel's transmitter (see Detailed Operation). The duration of CTS level change is described in the discussion of ICTSR[4].

0 CTS_N has no affect on the transmitter.

1 CTS_N affects the state of the transmitter.

[1:0] Transmitted Bits per Character — This field selects the number of data bits per character to be transmitted. The character length does not include the start, parity, and stop bits in ASYNC or the parity bit in COP. In BOP modes the character length for the address and control fields is always 8 bits, and the value of this field only applies to the information (I) field, except for the last character of the I field, whose length is specified by OMR[7:5].

Transmitter Timing Register (TTRA, TTRB)

[7] External Source — This bit selects the RTXC pin or the TRXC pin of the channel as the transmitter clock input when [6:4] specifies external. When used for input, the selected pin must be programmed as an input in the PCR [4:3] or [2:0].

0 External input from RTXC pin.

1 External input from TRXC pin.

[6:4] Transmitter Clock Select — This field selects the clock for the transmitter.

000 External clock from TRXC or RTXC at 1x the shift (baud) rate.

001 External clock from TRXC or RTXC at 16x the shift rate.

010 Internal clock from the phase locked loop at 1x the bit rate. It should be used only in half-duplex operation since the DPLL will periodically resync itself to the received data if in full duplex operation.

011 Internal clock from the bit rate generator at 32x the shift rate. The clock signal is divided by two before use in the transmitter which operates at 16x the baud rate. Rate selected by [3:0].

100 Internal clock from counter/timer of other channel. The C/T should be programmed to produce a clock at 2X the shift rate.

101 Internal clock from counter/timer of other channel. The C/T should be programmed to produce a clock at 32X the shift rate.

110 Internal clock from the counter/timer of own channel. The C/T should be programmed to produce a clock at 2X the shift rate.

111 Internal clock from the counter/timer of own channel. The C/T should be programmed to produce a clock at 32X the shift rate.

[3:0] Bit Rate Select — This field selects an output from the bit rate generator to be used by the transmitter circuits. The actual frequency output from the BRG is 32X the bit rate shown in table 5. With a crystal or external clock of 14.7456MHz the bit rates are as given in table 5 (this input is divided by two before being applied to the oscillator circuit).

Table 5. RECEIVER/TRANSMITTER BAUD RATES

[3:0]	BIT RATE	[3:0]	BIT RATE
0000	50	1000	1050
0001	75	1001	1200
0010	110	1010	2000
0011	134.5	1011	2400
0100	150	1100	4800
0101	200	1101	9600
0110	300	1110	19.2K
0111	600	1111	38.4K

Receiver Parameter Register (RPRA, RPRB)

[7] SYN Stripping — This bit controls the DUSCC processing in COP modes of SYN 'character patterns' that occur after the initial character synchronization. Refer to Detailed Operation of the receiver for details and definition of SYN 'patterns', and their accumulation of FCS.

0 Strip only leading SYN 'patterns' (i.e. before a message).

1 Strip all SYN 'patterns' (including all odd DLE's in BISYNC transparent mode).

[6] Transfer Received FCS to FIFO — In BISYNC and BOP modes, the assertion of this bit causes the received FCS to be loaded into the RXFIFO. BOP mode operates correctly only if a minimum of two extra FLAGs (without shared zeros) are appended to the frame. If the FCS is specified to be transferred to the FIFO, the EOM status bit will be tagged onto the last byte of the FCS instead of to the last character of the message.

0 Do not transfer FCS to RXFIFO.

1 Transfer FCS to RXFIFO.

[5] Auto-Hunt and Pad Check (BISYNC) — In BISYNC mode, the assertion of this bit causes the receiver to go into hunt for character sync mode after detecting certain end-of-message (EOM) characters. These are defined in the Detailed Operations section for COP receiver operation. After the EOT and NAK sequences, the receiver also does a check for a closing PAD of four 1's.

0 Disable auto-hunt and PAD check.

1 Enable auto-hunt and PAD check.

[5] Overrun Mode (BOP) — The state of this control bit determines the operation of the receiver in the event of a data overrun, i.e., when a character is received while the RXFIFO and the RX shift register are both full.

0 The receiver terminates receiving the current frame and goes into hunt phase, looking for a FLAG to be received.

1 The receiver continues receiving the current frame. The overrunning character is lost. (The five characters already assembled in the RXFIFO and RX shift register are protected).

[4] Receiver Request-to-Send Control (ASYNC) — See Detailed Operation.

0 Receiver does not control RTS_N output.

1 Receiver can negate RTS_N output.

[4] External Sync (COP) — In COP single SYN mode, the assertion of this bit enables external character synchronization and receipt of SYN patterns is not required. In order to use this feature, the DUSCC must be programmed to COP single SYN mode, CMR1[2:0] = 110, which is used to set up the internal data paths. In all other respects, however, the external sync mode operation is protocol transparent. A negative signal on the DCD_N/SYNI_N pin will cause the receiver to establish synchronization on the next rising edge of the receiver clock. Character assembly will start at this edge with the RXD input pin considered to have the second bit of data. The sync signal can then be negated. Receipt of the active high external sync input causes the SYN detect status bit (RSR[2]) to be set and the SYNOUT_N pin to be asserted for one bit time. When this mode is enabled, the internal SYN (COP mode) detection and special character recognition (e.g., IDLE, STX, ETX, etc.) circuits are disabled. Character assembly begins as if in the I-field with character length as programmed in RPR[1:0]. Incoming COP frames with parity specified optionally can have it stripped by programming RPR[3]. The user must wait at least eight bit times after RX is enabled before applying the SYNI_N signal. This time is required to flush the internal data paths. The receiver remains in this mode and further external sync pulses are ignored until the receiver is disabled and then reenabled to resynchronize or to return to normal mode. See figure 2.

0 External sync not enabled.

1 External sync enabled.

Note that EXT SYNC and DCD ENABLE RX cannot be asserted simultaneously since they use the same pin.

[3] Strip Parity — In COP and ASYNC modes with parity enabled, this bit controls whether the received parity bit is stripped from the data placed in the receiver FIFO. It is valid only for programmed character lengths of 5, 6, and 7 bits. If the bit is stripped, the corresponding bit in the received data is set to zero.

0 Transfer parity bit as received.

1 Strip parity bit from data.

[3] All Parties Address — In BOP secondary modes, the assertion of this bit causes the receiver to 'wake up' upon receipt of the address H'FF' or H'FF, FF', for single and dual octet address modes respectively, in addition to its normal station address. This feature allows all stations to receive a message.

0 Don't recognize all parties address.

1 Recognize all parties address.

[2] DCD Enable Receiver — If this bit is asserted, the DCD_N/SYNI_N input must be low in order for the receiver to operate. If the input is negated (goes high) while a character is being received, the receiver terminates receipt of the current message (this action in effect disables the receiver). If DCD is subsequently asserted, the receiver will search for the start bit, SYN pattern, or FLAG, depending on the channel protocol. (Note that the change of input can be programmed to generate an interrupt; the duration of the DCD level change is described in the discussion of the input and counter/timer status register ICTSR[5]).

0 DCD not used to enable receiver

1 DCD used to enable receiver

EXT SYNC and DCD ENABLE RX cannot be asserted simultaneously since they use the same pin.

[1:0] Received Bits per Character — This field selects the number of data bits per character to be assembled by the receiver. The character length does not include the start, parity, and stop bits in ASYNC or the parity bit in COP. In BOP modes, the character length for the address and control fields is always 8 bits, and the value of this field only applies to the information field. If the number of bits assembled for the last character of the I field is less than the value programmed in this field, RCL not zero (RSR[0]) is asserted and the actual number of bits received is given in TRSR[2:0].

Receiver Timing Register (RTRA, RTRB)

[7] External Source — This bit selects the RTXC pin or the TRXC pin of the channel as the receiver or DPLL clock input, when [6:4] specifies external. When used for input, the selected pin must be programmed as an input in the PCR [4:3] or [2:0].

0 External input from RTXC pin.

1 External input from TRXC pin.

[6:4] Receiver Clock Select — This field selects the clock for the receiver.

000 External clock from TRXC or RTXC at 1X the shift (baud) rate.

001 External clock from TRXC or RTXC at 16X the shift rate. Used for ASYNC mode only.

010 Internal clock from the bit rate generator at 32X the shift rate. Clock is divided by two before use by the receiver logic, which operates at 16X the baud rate. Rate selected by [3:0]. Used for ASYNC mode only.

011 Internal clock from counter/timer of own channel. The C/T should be programmed to produce a clock at 32X the shift rate. Clock is divided by two before use in the receiver logic. Used for ASYNC mode only.

100 Internal clock from the digital phase locked loop. The clock for the DPLL is a 64X clock from the crystal oscillator or system clock input. (The input to the oscillator is divided by two).

101 Internal clock from the digital phase locked loop. The clock for the DPLL is an external 32X clock from the RTXC or TRXC pin, as selected by [7].

110 Internal clock from the digital phase locked loop. The clock for the DPLL is a 32X clock from the BRG. The frequency is programmed by [3:0].

111 Internal clock from the digital phase locked loop. The clock for the DPLL is a 32X clock from the counter/timer of the channel.

[3:0] Bit Rate Select — This field selects an output from the bit rate generator to be used by the receiver circuits. The actual frequency output from the BRG is 32X the bit rate shown in table 5.

Output and Miscellaneous Register (OMRA, OMRB)

[7:5] Transmitted Residual Character Length — In BOP modes, this field determines the number of bits transmitted for the last character in the information field. This length applies to:

- the character in the transmit FIFO accompanied by the FIFO'ed TEOM command.
- the character loaded into the FIFO by the DMA controller if DONEN is simultaneously asserted and TPR[4] is asserted.
- the character loaded into the FIFO which causes the counter to reach zero count when TPR[4] is asserted.

The length of all other characters in the frame's information field is selected by TPR[1:0]. If this field is 111, the number of bits in the last character is the same as programmed in TPR[1:0].

[4] TXRDY Activate Mode —

0 FIFO not full. The channel's TXRDY status bit is asserted each time a character is transferred from the transmit FIFO to the transmit shift

register. If not reset by the CPU, TXRDY remains asserted until the FIFO is full, at which time it is automatically negated.

1 FIFO empty. The channel's TXRDY status bit is asserted when a character transfer from the transmit FIFO to the transmit shift register causes the FIFO to become empty. If not reset by the CPU, TXRDY remains asserted until the FIFO is full, at which time it is negated.

If the TXRDY status bit is reset by the CPU, it will remain negated regardless of the current state of the transmit FIFO, until it is asserted again due to the occurrence of one of the above conditions.

[3] RXRDY Activate Mode —

0 FIFO not empty. The channel's RXRDY status bit is asserted each time a character is transferred from the receive shift register to the receive FIFO. If not reset by the CPU, RXRDY remains asserted until the receive FIFO is empty, at which time it is automatically negated.

1 FIFO full. The channel's RXRDY status bit is asserted when a character transfer from the receive shift register to the receive FIFO causes the FIFO to become full. If not reset by the CPU, RXRDY remains asserted until the FIFO is empty, at which time it is negated.

The RXRDY status bit will also be asserted, regardless of the receiver FIFO full condition, when an end-of-message character is loaded in the RXFIFO (BOP/BISYNC), when a BREAK condition (ASYNC mode) is detected in RSR[2], or when the counter/timer is programmed to count received characters and the character which causes it to reach zero is loaded in the FIFO (all modes). (Refer to the detailed operation of the receiver.)

If reset by the CPU, the RXRDY status bit will remain negated, regardless of the current state of the receiver FIFO, until it is asserted again due to one of the above conditions.

[2] General Purpose Output 2 — This general purpose bit is used to control the TXDRQ_N/GP02_/RTS_N pin, when it is used as an output. The output is high when the bit is a 0 and is low when the bit is a 1.

[1] General Purpose Output 1 — This bit is used to control the RTXDRQ_N/GPO1_N output, which is a general purpose output when the channel is not in DMA mode. The output is high when the bit is a 0 and is low when the bit is a 1.

[0] Request-to-Send Output — This bit controls the TXDRQ_N/GPO2_N/RTS_N and SYNOUT_N/RTS_N pin, when either is used as a RTS output. The output is high when the bit is a 0 and is low when the bit is a 1.

Counter/Timer Control and Value Registers

There are five registers in this set consisting of the following:

1. Counter/timer control register (CTCRA/B)
2. Counter/timer preset high and low registers (CTPRHA/B, CTPRLA/B)
3. Counter/timer (current value) high and low registers (CTHA/B, CTLA/B)

The format of each of the registers of this set is contained in table 6. The control register contains the operational information for the counter/timer. The preset registers contain the count which is loaded into the counter/timer circuits. The third group contains the current value of the counter/timer as it operates.

Counter/Timer Control Register (CTCRA, CTCRB)

[7] Zero Detect Interrupt — This bit determines whether the assertion of the C/T ZERO COUNT status bit (ICTSR[6]) causes an interrupt to be generated.

0 Interrupt disabled.

1 Interrupt enabled if master interrupt enable (ICR[1] or ICR[0]) is asserted.

[6] Zero Detect Control — This bit determines the action of the counter upon reaching zero count.

0 The counter/timer is preset to the value contained in the counter/timer preset registers (CTPRL, CTPRH) at the next clock edge.

1 The counter/timer continues counting without preset. The value at the next clock edge will be H'FFFF'.

[5] Counter/Timer Output Control — This bit selects the output waveform when the counter/timer is selected to be output on TRXC or RTXC.

1 The output is a single clock positive width pulse each time the C/T reaches zero count. (The duration of this pulse is one clock period.)

0 The output toggles each time the C/T reaches zero count. The output is cleared to low by either of the preset counter/timer commands.

[4:3] Clock Select — This field selects whether the clock selected by [2:0] is prescaled prior to being applied to the input of the C/T.

00 No prescaling.

01 Divide clock by 16.

10 Divide clock by 32.

11 Divide clock by 64.

[2:0] Clock Source — This field selects the clock source for the counter timer.

000 RTXC pin. Pin must be programmed as input.

001 TRXC pin. Pin must be programmed as input.

010 Source is the crystal oscillator or system clock input divided by four.

011 This selects a special mode of operation. In this mode the counter, after receiving the 'start C/T' command, delays the start of counting until the RXD input goes low. It continues counting until the RXD input goes high, then stops and sets the C/T zero count status bit. The CPU can use the value in the C/T to determine the bit rate of the incoming data. The clock is the crystal oscillator or system clock input divided by four.

100 Source is the 32x BRG output selected by RTR[3:0] of own channel.

101 Source is the 32x BRG output selected by TTR[3:0] of own channel.

110 Source is the internal signal which loads received characters from the receive shift register into the receiver FIFO. When operating in this mode, the FIFO'ed EOM status bit (RSR[7]) shall be set when the character which causes the count to go to zero is loaded into the receive FIFO.

111 Source is the internal signal which transfers characters from the data bus into the transmit FIFO. When operating in this mode, and if the TEOM on zero count or done control bit (TPR[4]) is asserted, the FIFO'ed Send EOM command will be automatically asserted when the character which causes the count to go to zero is loaded into the transmit FIFO.

Counter/Timer Preset High Register (CTPRHA, CTPRHB)

[7:0] MSB — This register contains the eight most significant bits of the value loaded into the counter/timer upon receipt of the load C/T from preset register command or when the counter/timer reaches zero count and the zero detect control bit (CTCR[6]) is negated.

Table 6. COUNTER/TIMER CONTROL AND VALUE REGISTER BIT FORMATS

COUNTER/TIMER CONTROL REG

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(CTCRA, CTCRB)	Zero Detect Interrupt	Zero Detect Control	Output Control	Prescaler		Clock Source		
	0 – disable 1 – enabled	0 – preset 1 – continue	0 – square 1 – pulse	00 – 1 01 – 16 10 – 32 11 – 64		000 – RTXC pin 001 – TRXC pin 010 – X1/CLK divided by 4 011 – X1/CLK divided by 4 gated by RXD 100 – RX BRG 101 – TX BRG 110 – RX characters 111 – TX characters		

COUNTER/TIMER PRESET HIGH REG

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(CTPRHA, CTPRHB)	Most significant bits of counter/timer preset value							

COUNTER/TIMER PRESET REGISTER LOW

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(CTPRLA, CTPRLB)	Least significant bits of counter/timer preset value							

COUNTER/TIMER HIGH

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(CTHA, CTHB)	Most significant bits of counter/timer							

COUNTER/TIMER LOW

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(CTLA, CTLB)	Least significant bits of counter/timer							

The minimum 16-bit counter/timer preset value is H'0002'.

Counter/Timer Preset Low Register (CTPRLA, CTPRLB)

[7:0] LSB — This register contains the eight least significant bits of the value loaded into the counter/timer upon receipt of the load C/T from preset register command or when the counter/timer reaches zero count and the zero detect control bit (CTCR[6]) is negated. The minimum 16-bit counter/timer preset value is H'0002'.

Counter/Timer High Register (CTHA, CTHB)

[7:0] MSB — A read of this 'register' provides the eight most significant bits of the current value of the counter/timer. It is recommended that the C/T be stopped via a stop counter command before it is read in order to prevent errors which may occur due to the read being performed while the C/T is changing. This count is continued after the register is read.

Counter/Timer Low Register (CTLA, CTLB)

[7:0] LSB — A read of this 'register' provides the eight least significant bits of the current value of the counter/timer. It is recommended that the C/T be stopped via a stop counter command before it is read, in order to prevent errors which may occur due to the read being performed while the C/T is changing. This count is continued after the register is read.

Interrupt Control and Status Registers

This group of registers define mechanisms for communications between the DUSCC and the processor and contain the device status information. Four registers, available for each channel, and four common device registers comprise this group which consists of the following:

1. Interrupt enable register (IERA/B)
2. Receiver status register (RSRA/B)
3. Transmitter and receiver status register (TRSRA/B)
4. Input and counter/timer status register (ICTSRA/B)
5. Interrupt vector register (IVR) and modified interrupt vector register (IVRM)
6. Interrupt control register (ICR)
7. General status register (GSR)

See table 7 for bit formats and figure 3 for table relationships.

Interrupt Enable Register (IERA, IERB)
This register controls whether the assertion of bits in the channel's status registers causes an interrupt to be generated. An additional condition for an interrupt to be generated is that the channel's master interrupt enable bit, ICR[0] or ICR[1], be asserted.

[7] DCD/CTS —
0 Interrupt not enabled.
1 Interrupt generated if ICTSR[4] or ICTSR[5] are asserted.

[6] TXRDY —
0 Interrupt not enabled.
1 Interrupt generated if TXRDY (GSR[1] or GSR[5] for channels A and B respectively) is asserted.

[5] TRSR 73 —
0 Interrupt not enabled.
1 Interrupt generated if bits 7, 6, 5, 4 or 3 of the TRSR are asserted.

[4] RXRDY —
0 Interrupt not enabled.
1 Interrupt generated if RXRDY (GSR[0] or GSR[4] for channels A and B respectively) is asserted.

[3] RSR 76 —
0 Interrupt not enabled.
1 Interrupt generated if bits 7 or 6 of the RSR are asserted.

[2] RSR 54 —
0 Interrupt not enabled.
1 Interrupt generated if bits 5 or 4 of the RSR are asserted.

[1] RSR 32 —
0 Interrupt not enabled.
1 Interrupt generated if bits 3 or 2 of the RSR are asserted.

Table 7. INTERRUPT CONTROL AND STATUS REGISTER BIT FORMAT

RECEIVER STATUS REG

	*BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(RSRA, RSRB) ASYNC	# Char compare	RTS negated	Overrun error	not used	BRK end detect	BRK start detect	# Framing error	# Parity error
COP	# EOM detect +	PAD error +	Overrun error	not used	not used	Syn detect	# CRC error	# Parity error
BOP	# EOM detect	Abort detect	Overrun error	Short frame detect	Idle detect	Flag detect	# CRC error	# RCL not zero
LOOP	# EOM detect	Abort/EOP detect	Overrun error	Short frame detect	Turn-around detect	Flag detect	# CRC error	# RCL not zero

NOTES: # Status bit is FIFO'ed.
+ COP BISYNC mode only.
* All modes indicate character count complete.

TRANSMITTER AND RECEIVER STATUS REG

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(TRSRA, TRSRB) ASYNC	Transmitter empty	CTS underrun	not used	Send break ack	DPLL error	not used	not used	not used
COP	Transmitter empty	CTS underrun	Frame complete	Send SOM ack	DPLL error	not used	RX hunt mode	RX xpnt mode
BOP	Transmitter empty	CTS underrun / Loop sending*	Frame complete	Send SOM/ abort ack	DPLL error	**RX Residual Character Length**: 000 – 0 bit, 001 – 1 bits, 010 – 2 bits, 011 – 3 bits, 100 – 4 bits, 101 – 5 bits, 110 – 6 bits, 111 – 7 bits		

*Loop mode only

INPUT AND COUNTER/TIMER STATUS REGISTER

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(ICTSRA, ICTSRB)	C/T running	C/T zero count	Delta DCD	Delta CTS/LC	DCD	CTS/LC	GPI2	GPI1

Table 7. INTERRUPT CONTROL AND STATUS REGISTER BIT FORMAT (Continued)

INTERRUPT ENABLE REG

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(IERA, IERB)	DCD/CTS	TXRDY	TRSR [7:3]	RXRDY	RSR [7:6]	RSR [5:4]	RSR [3:2]	RSR [1:0]
	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes

INTERRUPT VECTOR REG AND INTERRUPT VECTOR MODIFIED REG

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(IVR, IVRM)	8-bit interrupt vector							

GENERAL STATUS REG

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(GSR)	Channel B				Channel A			
	External or C/T status	RX/TX status	TXRDY	RXRDY	External or C/T status	RX/TX status	TXRDY	RXRDY

INTERRUPT CONTROL REG

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(ICR)	Channel A/B Interrupt Priority		Vector Mode		Bits to Modify	Vector Includes Status	Channel A Master Int Enable	Channel B Master Int Enable
	00 – channel A 01 – channel B 10 – interleaved A 11 – interleaved B		00 – vectored 01 – vectored 10 – vectored 11 – non vectored		0 – 2:0 1 – 4:2	0 – no 1 – yes	0 – no 1 – yes	0 – no 1 – yes

[0] RSR 10 —

0 Interrupt not enabled.

1 Interrupt generated if bits 1 or 0 of the RSR are asserted.

Receiver Status Register (RSRA, RSRB)

This register informs the CPU of receiver status. Bits indicated as 'not used' in a particular mode will read as zero. The logical OR of these bits is presented in GSR[2] or GSR[6] (OR'ed with the bits of TRSR) for channels A and B respectively. Unless otherwise indicated, asserted status bits are reset only by performing a write operation to the status register with the bits to be reset being ones in the accompanying data word, or when the RESETN input is asserted, or when a 'reset receiver' command is issued.

Certain status bits are specified as being FIFO'ed. This means that they occupy positions in a status FIFO that correspond to the data FIFO. As the data is brought to the top of the FIFO (the position read when the RXFIFO is read), the FIFO'ed status bits are logically OR'ed with the previous contents of the corresponding bits in the status register. This permits the user to obtain status either character by character or on a block basis. For character by character status, the SR bits should be read and then cleared before reading the character data from RXFIFO. For block status, the status register is initially cleared and then read after the message is received. Asserted status bits can be programmed to generate an interrupt (see Interrupt Enable Register).

[7] Character Count Complete (All Modes), Character Compare (ASYNC), EOM (BISYNC/BOP/LOOP) — If the counter/timer is programmed to count received characters, this bit is asserted when the character which causes the count to go to zero is loaded into the receive FIFO. It is also asserted to indicate the following conditions:

ASYNC The character currently at the top of RXFIFO matched the contents of S1R. A character will not compare if it is received with parity error even if the data portion matches.

BISYNC The character currently at the top of the FIFO was either a text message terminator or a control sequence received outside of a text or header field. See Detailed Operation of COP Receiver. If transfer FCS to FIFO (RPR[6]) is set, the EOM will instead be tagged onto the last byte of the FCS. Note that if an overrun occurs during receipt of a message, the EOM character may be lost, but this status bit will still be asserted to indicate that an EOM was received. For two byte EOM comparisons, only the second byte is tagged (assuming the CRC is not transferred to the FIFO).

BOP, LOOP The character currently at the top of the FIFO was the last character of the frame. If transfer FCS to FIFO (RPR[6]) is asserted, the EOM will be tagged instead onto the last byte of the FCS. Note that if an overrun occurs during receipt of a message, the EOM character may be lost, but this status bit will still be asserted to indicate that an EOM was received.

[6] RTS Negated (ASYNC), PAD Error (BISYNC), ABORT (BOP) —

ASYNC The RTSN output was negated due to receiving the start bit of a new character while the RXFIFO was full (see RPR[4]).

BISYNC PAD error detected (see RPR[5]).

BOP An ABORT sequence consisting of a zero followed by seven ones was received after receipt of the first address octet but before receipt of the closing FLAG. The user should read RXFIFO until it is empty and determine if any valid characters from a previous frame are in the FIFO. If no character with a tagged EOM detect ([7]) is found, all characters are from the current frame and should be discarded along with any previously read by the CPU. An ABORT detect causes the receiver to automatically go into search for FLAG state. An abort during a valid frame does not cause the CRC to reset; this will occur when the next frame begins.

LOOP Performs the ABORT detect function as described for BOP without the restriction that the pattern be detected during an active frame. A zero followed by seven ones is the end-of-poll sequence which allows the transmitter to go active if the 'go active on poll' command has been invoked.

[5] Overrun Error (All Modes) — A new character was received while the receive FIFO was full and a character was already waiting in the receive shift register to be transferred to the FIFO. The DUSCC protects the five characters previously assembled (four in RXFIFO, one in the RX shift register) and discards the overrunning character(s). After the CPU reads the FIFO, the character waiting in the RXSR will be loaded into the available FIFO position. This releases the RXSR and a new character assembly will start at the next character boundary. In this way, only valid characters will be assembled, i.e. no partial character assembly will occur regardless of when the RXSR became available during the incoming data stream.

[4] Short Frame (BOP/LOOP) —

ASYNC Not used

COP Not used

BOP, LOOP A closing flag was received with missing fields in the frame. See detailed operation for BOP receiver.

[3] BREAK End Detect (ASYNC), IDLE (BOP), Turnaround (LOOP) —

ASYNC 1X clock mode: The RXD input has returned to the marking state for at least one period of the 1X receiver clock after detecting a BREAK.
16X clock mode: The RXD input has returned to the marking (high) state for at least one-half bit time after detecting a BREAK. A half bit time is defined as eight clock cycles of the 16X receiver clock.

COP Not used

BOP An IDLE sequence consisting of a zero followed by fifteen ones was received. During a valid frame, an abort must precede an idle. However, outside of a valid frame, an idle is recognized and abort is not.

LOOP A turnaround sequence consisting of eight contiguous zeros was detected outside of an active frame. This should normally be used to terminate transmitter operation and return the system to the 'echoing RXD' mode.

[2] BREAK Start Detect (ASYNC), SYN Detect (COP), FLAG Detect (BOP/LOOP)

ASYNC An all zero character, including parity (if specified) and first stop bit, was received. The receiver shall be capable of detecting breaks which begin in the middle of a previous character. Only a single all-zero character shall be put into the FIFO when a break is detected. Additional entries to the FIFO are inhibited until the end of break has been detected (see above) and a new character is received.

COP A SYN pattern was received. Refer to Detailed Operation for definition of SYN patterns. Set one bit time after detection of SYN pattern in HSRH, HSRL. See figure 1 for receiver data path.

BOP, LOOP A FLAG sequence (01111110) was received. Set one bit time after FLAG is detected in CCSR. See figure 1 for receiver data path.

[1] Framing Error (ASYNC), CRC Error (COP/BOP/LOOP) —

ASYNC At the first stop bit position the RXD input was in the low (space) state. The receiver only checks for framing error at the nominal center of the first stop bit regardless of the number of stop bits programmed in TPR[7:4]. This bit is not set for BREAKS.

COP In BISYNC COP mode, this bit is set upon receipt of the BCC byte(s), if any, to indicate that the received BCC was in error. The bit is normally FIFO'ed with the last byte of the frame (the character preceding the first BCC byte). However, if transfer FCS to FIFO (RPR[6]) is asserted, this bit is FIFO'ed with the last BCC byte. The value of this bit should be ignored for non-text messages or if the received frame was aborted via an ENQ. In non-BISYNC COP modes, the bit is set with each received character if the current value of the CRC checker is not equal to the non-error value (see CMR2[2:0]).

BOP, LOOP This bit is set upon receipt of the FCS byte(s), if any, to indicate that the received FCS was in error. The bit is normally FIFO'ed with the last byte of the I field (the character preceding the first FCS byte). However, if transfer FCS to FIFO (RPR[6]) is asserted, this bit is FIFO'ed with the last FCS byte.

[0] Parity Error (ASYNC/COP), RCL Not Zero (BOP/LOOP) —

ASYNC The parity bit of the received character was not as expected. A parity error does not affect the parity bit put into the FIFO as part of the character when strip parity (RPR [3]) is negated.

COP The parity bit of the received character was not as expected. A parity error does not affect the parity bit put into the FIFO as part of the character when strip parity (RPR[3]) is negated. A SYN or other character received with parity error is treated as a data character. Thus, a SYN with parity error received while in SYN search state will not establish character sync. Characters received with parity error while in the SYN search state will not set the error bit.

BOP, LOOP The last character of the I field did not have the character length specified in RPR[1:0]. The actual received character length of this byte can be read in TRSR[2:0]. This bit is FIFO'ed with the EOM character but TRSR[2:0] is not. An exception occurs if the command to transfer the FCS to the FIFO is active. In this case, the bit will be FIFO'ed with the last byte of the FCS, i.e., with REOM. In the event

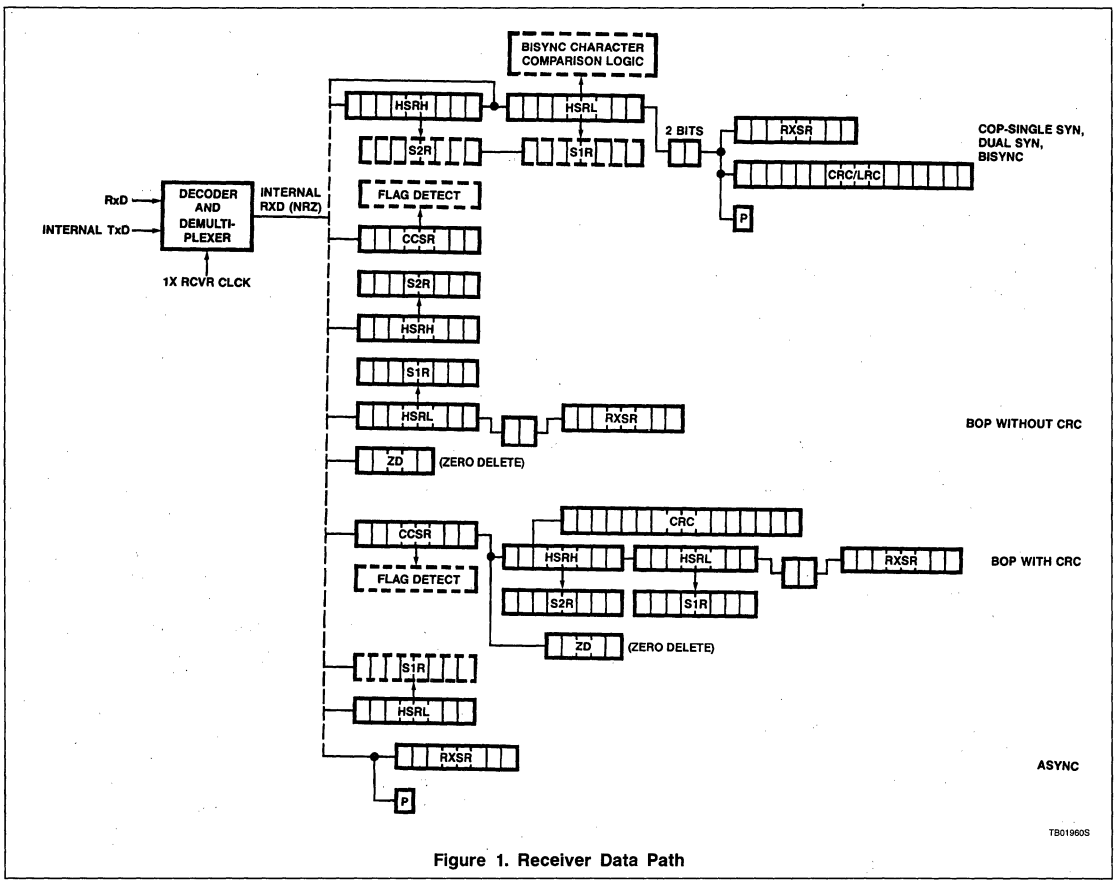


Figure 1. Receiver Data Path

that residual characters from two consective frames are received and are both in the FIFO, the length in TRSR[2:0] applies to the last received residual character.

Transmitter/Receiver Status Register (TRSRA, TRSRB)

This register informs the CPU of transmitter and receiver status. Bits indicated as not used in a particular mode will read as zero. The logical OR of bits [7:3] is presented in GSR[2] or GSR[6] (OR'ed with the bits of RSR) for channels A and B respectively. Unless otherwise indicated, asserted status bits are reset only:

1. By performing a write operation to the status register with the bits to be reset being ones in the accompanying data word [7:3].
2. When the RESETN input is asserted.
3. For [7:4], when a 'reset transmitter' command is issued.
4. For [3:0], when a 'reset receiver' command is issued.
5. For [2:0], see description in BOP mode.

Asserted status bits in [7:3] can be programmed to generate an interrupt. See IER.

[7] Transmitter Empty — Indicates that the transmit shift register has completed serializing a character and found no other character to serialize in the TX FIFO. The bit is not set until at least one character from the transmit FIFO (not including PAD characters in synchronous modes) has been serialized. The transmitter action after transmitter empty depends on operating mode:

ASYNC	The TXD output is held in the MARK state until another character is loaded into the TXFIFO. Normal operation then continues.
COP	Action is specified by TPR[7:6].
BOP, LOOP	Action is specified by TPR[7:6].

[6] CTS Underrun (ASYNC/COP/BOP), Loop sending (LOOP) —

ASYNC, COP, BOP	This bit is set only if CTS enable TX (TPR [2]) is asserted. It indicates that the transmit shift register was ready to begin serializing a character and found the CTS_N input negated. In ASYNC mode, this bit will be reasserted if cleared by the CPU while the CTS_N input is negated.
LOOP	Asserted when the go active on poll command has been invoked and an EOP sequence has been detected, causing the transmitter to go active by changing the EOP to a FLAG (see detailed operation of transmitter).

[5] Frame Complete (COP/BOP) —

ASYNC Not used.

COP Asserted at the beginning of transmission of the end of message sequence invoked by which is either a TEOM command, or when TPR[4], or TPR[7:6] = 00. The CPU can invoke the TSOM command after this bit is set to control the number of SYNs between transmitted frames.

BOP Asserted at the beginning of transmission of the end of message sequence which is invoked by either a TSOM command, or when TPR[4] = 1, or TPR[7:6] = 00. The CPU can invoke the send TSOM command after this bit is set to control the number of FLAGs between transmitted frames.

In COP/BOP modes, the frame complete status bit is set during the next-to-last bit (on TXD pin) of the last character in the data/information field. In BOP mode, if a 1-bit residual character is selected through OMR[7:5], then this bit is set during the next-to-last bit (on TXD pin) of the last full length character of the information field.

[4] Send Break Ack (ASYNC)/Send SOM ACK (COP)/Send SOM-Abort Ack (BOP) —

XASYNC Set when the transmitter begins transmission of a break in response to the send break command. If the command is reinvoked, the bit will be set again at the beginning of the next character time. The user can control the length of the break by counting character times through this mechanism.

COP Set when the transmitter begins transmission of a SYN pattern in response to the TSOM or TSOMP command. If the command is reinvoked, the bit will be set again at the beginning of the next transmitted SYN pattern. The user can control the number of SYNs which are sent through this mechanism.

BOP Set when the transmitter begins transmission of a FLAG/ABORT in response to the TSOM or TSOMP or TABRK command. If the command is reinvoked, the bit will be set again at the beginning of the next transmitted FLAG/ABORT. The user can control the number of FLAGs/ABORTs which are sent through this mechanism.

[3] DPLL Error — Set while the DPLL is operating in FM mode to indicate that a data transition was not detected within the detection window for two consecutive bits and that the DPLL was forced into search mode. This feature is disabled when the DPLL is specified as the clock source for the transmitter via TTR[6:4].

[2:0] Received Residual Character Length (BOP) —

BOP This field should be examined to determine the length of the last character of the I field (character tagged with REOM status bit) if RSR[0] is set to indicate that the length was not equal to the character length specified in RPR[1:0]. This field is negated when a reset receiver or disable receiver command is issued, or when the first control character for the next frame of data is in HSRL (see figure 1). Care must be taken to read TRSR[2:0] before these bits are cleared.

[1] Receiver in Hunt Mode (COP) —

COP This bit is asserted after the receiver is reset or disabled. It indicates that the receiver is in the hunt mode, searching the data stream for a SYN sequence to establish character synchronization. The bit is negated automatically when character sync is achieved.

[0] Receiver in Transparent Mode (BISYNC) —

COP Indicates that a DLE-STX sequence was received and the receiver is operating in BISYNC transparent mode. Set two bit times after detection of STX in HSRL. See Figure 1 for receiver data path. Transparent mode operation is terminated and the bit is negated automatically when one of the terminators for transparent text mode is received (DLE-ETX/ETB/ITB/ENQ).

Input and Counter/Timer Status Register (ICTSRA, ICTSRB)

This register informs the CPU of status of the counter/timer and inputs. The logical OR of bits [6:4] is presented in GSR[3] or GSR[7] for channels A and B respectively. Unless otherwise specified, bits of this register are reset only:

1. By performing a write operation to the status register with the bits to be reset (ones in the accompanying data word for bits [6:4] only).
2. When the RESETN input is asserted (bits [7:4]) only.

[7] Counter/Timing Running — Set when the C/T is started by start C/T command and reset when it is stopped by a stop C/T command.

[6] Counter/Timer Zero Detect — Set when the counter/timer reaches zero count. The assertion of this bit causes an interrupt to be generated if ICTCR[7] and the channel's master interrupt enable (ICR[1] or ICR[0]) are asserted.

[5] Delta DCD — The DCD input is sampled approximately every 6.8μsec using the 32X, 4800 baud output from the BRG. After synchronizing with the sampling clock, at least two consecutive samples at the same level are required to establish the level. As a consequence, a change of state at the DCD input, lasting at least 17μsec, will set this bit. The reset circuitry initializes the sampling circuits so that a change is not falsely indicated at power on time. The assertion of this bit causes an interrupt to be generated if IER[7] and the channel's master interrupt enable (ICR[1] or ICR[0]) are asserted.

[4] Delta CTS/LC — When not in loop mode, the CTS input is sampled approximately every 6.8μsec using the 32X, 4800 baud output from the BRG. After synchronizing with the sampling clock, at least two consecutive samples at the same level are required to establish the level. As a consequence, a change of state at the CTS input, lasting at least 17μs, will set this bit. The reset circuitry initi alizes the sampling circuits so that a change is not falsely indicated at power on time. The assertion of this bit causes an interrupt to be generated if IER[7] and the channel's master interrupt enable (ICR[1] or ICR[0]) are asserted.

In SDLC loop mode, this bit is set upon transitions of the LC output. LC is asserted in response to the 'go on-loop' command when the receiver detects a sequence of eight ones.

[3:0] Current State of DCD, CTS, GPI2, and GPI1 Inputs — This field provides the current state of the channel's input pins. The bit's value is latched at the beginning of the read cycle.

Interrupt Vector Register (IVR) and Modified Vector Register (IVRM)

[7:0] Register Content — If ICR[2] = 0, the content of IVR register is output on the data bus when the DUSCC has issued an interrupt request and the responding interrupt acknowledge (IACKN) is received. The value in the IVR is initialized to H'0F' on master reset. If 'vector includes status' is specified by ICR[2] = 1, bits [2:0] or [4:2] (depending on ICR[3]), of the vector are modified as shown in table 8 to indicate the highest priority interrupt currently active. The priority is programmable through the ICR. This modified vector is stored in the IVRM. When ICR[2] =

1, the content of the IVRM is output on to the data bus on the interrupt acknowledge. The vector is not modified, regardless of the value of ICR[2], if the CPU has not written an initial vector into this register.

Either the modified or unmodified vector can also be read by the CPU via a normal bus read cycle (see table 1). The vector value is locked at the beginning of the IACK or read cycle until the cycle is completed.

Interrupt Control Register (ICR)

[7:6] Channel A/B Interrupt Priority — Selects the relative priority between channels A and B. The state of this bit determines the value of the interrupt vector (see Interrupt Vector Register). The priority within each channel, from highest to lowest, is as follows:

0 Receiver ready

1 Transmitter ready

2 RX/TX status

3 External or C/T status

00 Channel A has the highest priority. The DUSCC interrupt priorities from highest to lowest are as follows: A(0), A(1), A(2), A(3), B(0), B(1), B(2), B(3)

10 Priorities are interleaved between channels, but channel A has the highest priority between events of equal channel priority. The DUSCC interrupt priorities from highest to lowest are as follows: A(0), B(0), A(1), B(1), A(2), B(2), A(3), B(3)

01 Channel B has the highest priority. The DUSCC interrupt priorities from highest to lowest are as follows: B(0), B(1), B(2), B(3), A(0), A(1), A(2), A(3)

11 Priorities are interleaved between channels, but channel B has the highest priority between events of equal channel priority. The DUSCC interrupt priorities from highest to lowest are as follows: B(0), A(0), B(1), A(1), B(2), A(2), B(3), A(3)

Table 8. INTERRUPT STATUS ENCODING

IVRM [2:0]/ [4:2]	HIGHEST PRIORITY INTERRUPT CONDITION
000	Channel A receiver ready
001	Channel A transmitter ready
010	Channel A RX/TX status
011	Channel A external or C/T status
100	Channel B receiver ready
101	Channel B transmitter ready
110	Channel B RX/TX status
111	Channel B external or C/T status

[5:4] Vector Mode — The value of this field determines the response of the DUSCC when the interrupt acknowledge (IACKN) is received from the CPU.

00 or 01 or 10 Vectored mode. Upon interrupt acknowledge, the DUSCC locks its current interrupt status until the end of the acknowledge cycle. If it has an active interrupt pending, it responds with the appropriate vector and then asserts DTACKN. If it does not have an interrupt, it propagates the acknowledge through its X2/IDCN output if this function is programmed in PCRA[7]. Otherwise, the IACKN is ignored. Locking the interrupt status at the leading edge of IACKN prevents a device at a high position in the interrupt daisy chain from responding to an IACK issued for a lower priority device while the acknowledge is being propagated to that device.

11 Non-vectored mode. The DUSCC ignores an IACK if one is received; the interrupt vector is not placed on the data bus. The internal interrupt status is locked when a read of the IVR is performed. Except for the absence of the vector on the bus, the DUSCC performs as it does in vectored mode – the vector is prioritized and modified if programmed.

[3] Vector Bits to Modify — Selects which bits of the vector stored in the IVR are to be modified to indicate the highest priority interrupt pending in the DUSCC. See Interrupt Vector Register.

0 Modify bits 2:0 of the vector.

1 Modify bits 4:2 of the vector.

[2] Vector Includes Status — Selects whether the modified (includes status) (IVRM) or unmodified vector (IVR) is output in response to an interrupt acknowledge (see Interrupt Vector Register).

0 Unmodified vector.

1 Modified vector.

[1] Channel A Master Interrupt Enable —

0 Channel A interrupts are disabled.

1 Channel A interrupts are enabled.

[0] Channel B Master Interrupt Enable —

0 Channel B interrupts are disabled.

1 Channel B interrupts are enabled.

General Status Register (GSR)

This register provides a 'quick look' at the overall status of both channels of the DUSCC. A write to this register with 1s at the corresponding bit positions causes TXRDY (bits 5 and 1) and/or RXRDY (bits 4 and 0) to be reset. The other status bits can be reset only by resetting the individual status bits that they point to.

[7] Channel B External or Counter/Timer Status — This bit indicates that one of the following status bits is asserted: ICTSRB[6:4].

[6] Channel B Receiver or Transmitter Status — This bit indicates that one of the following status bits is asserted: RSRB[7:0], TRSRB[7:3].

[5] Channel B Transmitter Ready — The assertion of this bit indicates that one or more characters may be loaded into the channel B transmitter FIFO to be serialized by the transmit shift register. See description of OMR[4]. This bit can be asserted only when the transmitter is enabled. Disabling or resetting the transmitter negates TXRDY.

[4] Channel B Receiver Ready — The assertion of this bit indicates that one or more characters are available in the channel B receiver FIFO to be read by the CPU. See description of OMR[3]. RXRDY is initially reset (negated) by a chip reset or when a 'reset channel B receiver' command is invoked.

[3] Channel A External or Counter/Timer Status — This bit indicates that one of the following status bits is asserted: ICTSRA[6:4].

[2] Channel A Receiver or Transmitter Status — This bit indicates that one of the following status bits is asserted: RSRA[7:0], TRSRA[7:3].

[1] Channel A Transmitter Ready — The assertion of this bit indicates that one or more characters may be loaded into the channel A transmitter FIFO to be serialized by the transmit shift register. See description of OMR[4]. This bit can be asserted only when the transmitter is enabled. Disabling or resetting the transmitter negates TXRDY.

[0] Channel A Receiver Ready — The assertion of this bit indicates that one or more characters are available in the channel A receiver FIFO to be read by the CPU. See description of OMR[3]. RXRDY is initially reset (negated) by a chip reset or when a 'reset channel A receiver' command is invoked.

Channel Command Register (CCRA, CCRB)

Commands to the DUSCC are entered through the channel command register. The format of that register is shown in table 9. A read of this register returns the last invoked command (with bits 4 and 5 set to 1).

Table 9. COMMAND REGISTER BIT FORMAT

CHANNEL COMMAND REG

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
(CCRA, CCRB)					**Transmitter Command**			
	00 = Transmitter CMD		don't care	don't care	0000 – reset TX 0001 – reset TX CRC* 0010 – enable TX 0011 – disable TX 0100 – transmit SOM (TSOM) 0101 – transmit SOM with PAD (TSOMP) 0110 – transmit EOM (TEOM)* 0111 – transmit ABORT/BREAK (TABRK) 1000 – transmit DLE (TDLE)* 1001 – go active on poll 1010 – reset go active on poll 1011 – go on-loop 1100 – go off-loop 1101 – exclude from CRC*			
					Receiver Command			
	01 = Receiver CMD		don't care	don't care	0000 – reset RX 0001 – reserved 0010 – enable RX 0011 – disable RX			
					Counter/Timer Command			
	10 = C/T CMD		don't care	don't care	0000 – start 0001 – stop 0010 – preset to FFFF 0011 – preset from CTPRH/CTPRL			
					DPLL Command			
	11 = DPLL CMD		don't care	don't care	0000 – enter search mode 0001 – disable DPLL 0010 – set FM mode 0011 – set NRZI mode 0100 – reserved for test 0101 – reserved for test			

*FIFO'ed commands

Transmitter Commands

0000 Reset transmitter. Causes the transmitter to cease operation immediately. The transmit FIFO is cleared and the TXD output goes into the marking state. Also clears the transmitter status bits (TRSR[7:4]) and resets the TXRDY status bit (GSR[1] or GSR[5] for channels A and B respectively). The counter/timer and other registers are not affected.

0001 Reset transmit CRC. This command is appended to and FIFO'ed along with the next character loaded into the transmit FIFO. It causes the transmitter CRC generator to be reset to its initial state prior to beginning transmission of the appended character.

0010 Enable transmitter. Enables transmitter operation, conditioned by the state of the CTS ENABLE TX bit, TPR[2]. Has no effect if invoked when the transmitter has previously been enabled.

0011 Disable transmitter. Terminates transmitter operation and places the TXD output in the marking state at the next occurrence of a transmit FIFO empty condition. All characters currently in the FIFO, or any loaded subsequently prior to attaining an empty condition, will be transmitted.

0100 Transmit start of message. Used in COP and BOP modes to initiate transmission of a frame after the transmitter is first enabled, prior to sending the contents of the FIFO. Can also be used to precisely control the number of SYN/FLAGs at the beginning of transmission or in between frames.

When the transmitter is first enabled, transmission will not begin until this command (or the transmit SOM with PAD command, see below) is issued. The command causes the SYN (COP) or FLAG (BOP) pattern to be transmitted. SEND SOM ACK (TRSR[4]) is set when transmission of the SYN/FLAG begins. The CPU may then reinvoke the command if multiple SYN/FLAGs are to be transmitted. Transmission of the FIFO characters begins when the command is no longer reinvoked. If the FIFO is empty, SYN/FLAGs continue to be transmitted until a character is loaded into the FIFO, but the status bit is not set. Insertion of SYN/FLAGs between frames can be accomplished by invoking this command after the frame complete status bit (TRSR[5]) has been asserted in response to transmission of the end-of-message sequence.

0101 Transmit start of message with opening PAD. Used in COP and BOP modes after the transmitter is first enabled to send a bit pattern for DPLL synchronization prior to transmitting the opening SYN (COP) or FLAG (BOP). The SYN/FLAG is sent at the next occurrence of a transmit FIFO empty condition. All characters currently in the FIFO, or any loaded subsequently prior to attaining an empty condition, will be transmitted. While the PAD characters are transmitted, the character length is set to 8 bits, (regardless of the programmed length), and parity generation (COP), zero insertion (BOP), and LRC/CRC accumulation are disabled. SEND SOM ACK (TRSR[4]) is set when transmission of the SYN/FLAG begins. The CPU may then invoke the transmit SOM command if multiple SYN/FLAGs are to be transmitted.

The TSOM/TSOMP commands, described above, are sampled by the controller in alternate bit times of the transmitter clock. As a consequence, the first bit time of a COP/BOP frame will be transmitted on the TXD pin, after a maximum of three bit times, after the command is issued. (The additional 1-bit delay in the data path is due to the data encoding logic).

0110 Transmit end of message. This command is appended to the next character loaded into the transmit FIFO. It causes the transmitter to send the end-of-message sequence (selected FCS in COP modes, FCS – FLAG in BOP modes) after the appended character is transmitted. Frame complete (TRSR[5]) is set when transmission of the FCS begins. This command is also asserted automatically if the TEOM on zero count or done control bit (TPR[4]) is asserted, and the counter/timer is programmed to count transmitted characters when the character which causes the count to go to zero is loaded into the transmit FIFO.

0111 Transmit Abort BOP/Transmit Break ASYNC. In BOP modes, causes an abort (eight ones) to be transmitted after transmission of the character currently in the shift register is completed. The transmitter then sends MARKs or FLAGs depending on the state of underrun control (TPR[7:6]). Send SOM/abort ack (TRSR[4]) is set when the transmission of the abort begins. If the command is reasserted before transmission of the previous ABORT is completed, the process will be repeated. This can be used to send the idle sequence. The 'transmit SOM' command must be used to initiate transmission of a new message. In either mode, invoking this command causes the transmit FIFO to be flushed (characters are not transmitted).

In ASYNC mode, causes a break (space) to be transmitted after transmission of the character currently in the shift register is completed. Send break ack (TRSR[4]) is set when the transmission of the break begins. The transmitter keeps track of character times. If the command is reasserted, send break ack will be set again at the beginning of the next character time. The user can use this mechanism to control the length of the break in character time multiples. Transmission of the break is terminated by issuing a 'reset TX' or 'disable TX' command.

1000 Transmit DLE. Used in COP modes only. This command is appended to and FIFO'ed with the next character loaded into the transmitter FIFO. It causes the transmitter to send a DLE, (EBCDIC H'10', ASCII H'90') prior to transmitting the appended character. If the transmitter is operating in BISYNC transparent mode, the transmitter control logic automatically causes a second DLE to be transmitted whenever a DLE is detected at the top of the FIFO. In this case, the TDLE command should not be invoked. An extra (third) DLE, however, will not be sent if the transmit DLE command is invoked.

1001 Go active on poll. Used in BOP loop mode only. Causes the transmitter, if it is enabled, to begin sending when an EOP sequence consisting of a zero followed by seven ones is detected. The last one of the EOP is changed to zero, making it another FLAG, and then the transmitter operates as described in the detailed operation section. The loop sending status bit (TRSR[6]) is asserted concurrent with the beginning of transmission.

1010 Reset go active on poll. Clears the stored 'go active on poll' command.

1011 Go on-loop. Used in BOP loop mode to control the assertion of the LC_N output. This output provides the means of controlling external loop interface hardware to go on-loop and off-loop. When the command is asserted, the DUSCC will look for the receipt of seven contiguous ones, at which time it will assert the LC_N output and set the delta DCD/LC status bit (ICTSR[4]). This allows the DUSCC to break into the loop without affecting loop operation. This command must be used to initiate loop mode operation.

1100 Go off-loop. Used in BOP loop mode to control the negation of the LC_N output. This output provides the means of controlling external loop interface hardware to go on-loop and off-loop. When the command is asserted, the DUSCC will look for the receipt of eight contiguous ones, at which time it will negate the LC_N output and set the delta DCD/LC status bit (ICTSR[4]). This allows the DUSCC to get off the loop without affecting loop operation. This command is normally used to terminate loop mode operation.

1101 Exclude from CRC. This command is appended to and FIFO'ed along with the next character loaded into the transmit FIFO. It causes the transmitter CRC generator to be disabled while the appended character is being transmitted. Thus, that character is not included in the CRC accumulation.

Receiver Commands

0000 Reset Receiver. Causes the receiver to cease operation, clears the receiver FIFO, and clears the receiver status (RSR[7:0], TRSR[3:0], and either GSR[0] or GSR[4] for channels A and B, respectively). The counter/timer and other registers are not affected.

0001 Reserved.

0010 Enable receiver. Causes receiver operation to begin, conditioned by the state of the DCD ENABLE RX bit, RPR[2]. Receiver goes into START, SYN, or FLAG search mode depending on channel protocol mode. Has no effect if invoked when the receiver has previously been enabled.

0011 Disable receiver. Terminates operation of the receiver. Any character currently being assembled will be lost. Does not affect FIFO or any status.

Counter/Timer Commands

0000 Start. Starts the counter/timer and prescaler.

0001 Stop. Stops the counter/timer and prescaler. Since the command may be asynchronous with the selected clock source, the counter/timer and/

or prescaler may count one or more additional cycles before stopping.

0010 Preset to FFFF. Presets the counter timer to H'FFFF' and the prescaler to its initial value. This command causes the C/T output to go low.

0011 Preset from CTPRH/CTPRL. Transfers the current value in the counter/timer preset registers to the counter/timer and presets the prescaler to its initial value. This command causes the C/T output to go low.

Digital Phase Locked Loop Commands

0000 Enter Search Mode. In NRZI mode, this command causes the DPLL counter to be set to the value 15 and the clock output will be forced high. The counter will be disabled until a transition on the data line is detected, at which point it will start incrementing and the clock output will go from high to low. After the counter reaches a count of 31, it will reset to zero and cause the clock output to go from low to high. The DPLL will then continue normal operation. This allows the DPLL to be locked onto the data without pre-frame transitions. In FM mode, this command causes the DPLL counter to be set to zero and the DPLL output clock to go high. The counter will be disabled until a transition on the data line is detected, at which point the DPLL will start normal operation. This command should not be used if the DPLL is programmed to supply the clock for the transmitter and the transmitter is active.

0001 Disable DPLL. Disables operation of the DPLL.

0010 Set FM Mode. Sets the DPLL to the FM mode of operation, used when FM0, FM1, or Manchester (NRZ) is selected by CMR1[7:6] and starts operation of the DPLL.

0011 Set NRZI Mode. Sets the DPLL to the NRZI mode of operation, used when NRZ or NRZI is selected by CMR1[7:6], and starts operation of the DPLL.

0100 Reserved for test

0101 Reserved for test

DETAILED OPERATION

Interrupt Control

A single interrupt output (IRQN) is provided which is activated upon the occurrence of any of the following conditions:

Channel A external or C/T special condition

Channel B external or C/T special condition

Channel A RX/TX error or special condition

Channel B RX/TX error or special condition

Channel A TXRDY

Channel B TXRDY

Channel A RXRDY

Channel B RXRDY

Each of the above conditions occupies a bit in the general status register (GSR). If ICR[2] is set, the eight conditions are encoded into three bits which are inserted into bits [2:0] or [4:2] of the interrupt vector register. This forms the content of the IVRM during an interrupt acknowledge cycle. Unmodified and modified vectors can be read directly through specified registers. Two of the conditions are the inclusive OR of several other maskable conditions:

- Ext or C/T special condition: Delta DCD, delta CTS or C/T zero count (ICTSR[6:4]).
- RX/TX error or special condition: Any condition in the receiver status register (RSR[7:0]) or a transmitter or DPLL condition in the transmitter and receiver status register (TRSR[7:3]).

The TXRDY and RXRDY conditions are defined by OMR[4] and OMR[3], respectively. Also associated with the interrupt system are the interrupt enable register (IER), one bit in the counter/timer control register (CTCR), and the interrupt control register (ICR).

The IER is programmed to enable specified conditions or groups of conditions to cause an interrupt by asserting the corresponding bit. A negated bit prevents an interrupt from occurring when the condition is active and hence masks the interrupt. In addition to the IER, CTCR[7] could be programmed to enable or disable an interrupt upon the C/T zero count condition. The interrupt priorities within a channel are fixed. Priority between channels is controlled by ICR[7:6]. Refer to table 8 and ICR[7:6].

The ICR contains the master interrupt enables for each channel (ICR[1] and ICR[0]) which must be set if the corresponding channel is to cause an interrupt. The CPU vector mode is specified by ICR[5:4] which selects either vectored or non-vectored operation. If vectored mode is selected, the content of the IVR or IVRM is placed on the data bus when IACK is activated. If ICR[2] is set, the content of IVRM is output which contains the content of IVR and the encoded status of the interrupting condition.

Upon receiving an interrupt acknowledge, the DUSCC locks its current interrupt status until the end of the acknowledge cycle. If it has an active interrupt pending, it responds with the appropriate vector and then asserts DTACKN. If it does not have an interrupt, it propagates the acknowledge through its X2/IDCN output if this function is programmed in PCRA[7]; otherwise, the IACKN is ignored. Locking the interrupt status at the leading edge of IACKN prevents a device at a high position in the interrupt daisy chain from responding to an IACK issued for a lower priority device while the acknowledge is being propagated to that device.

DMA Control

The DMA control section provides the interface to allow the DUSCC to operate with an external DMA controller. One of four modes of DMA can be programmed for each channel independently via CMR2[5:3]:

- Half duplex single address. In this mode, a single pin provides both DMA read and write requests. Acknowledgement of the requests is via a single DMA acknowledge pin. The data transfer is accomplished in a single bus cycle — the DMA controller places the memory address of the source or destination of the data on the address bus and then issues the acknowledge signal, which causes the DUSCC to either write the data into its transmit FIFO (write request) or to output the contents of the top of the receive FIFO (read request). The cycle is completed when the DTCN input is asserted by the DMA controller. This mode can be used when channel operation is half duplex (e.g., BISYNC). It allows a single DMA channel to service the receiver and transmitter.
- Half duplex dual address. In this mode, a single pin provides both DMA read and write requests. Acknowledgement of the requests is via normal bus read and write cycles. The data transfer requires two bus cycles — the DMA controller acquires the data from the source (memory for a TX DMA or DUSCC for a RX DMA) on the first cycle and deposits it at the destination (DUSCC for a TX DMA or memory for a RX DMA) on the second bus cycle. This mode is used when channel operation is half duplex (e.g., BISYNC) and allows a single DMA channel to service the receiver and transmitter.
- Full duplex single address. This mode is similar to half duplex single address mode but provides separate request and acknowledge pins for the receiver and transmitter.
- Full duplex dual address. This mode is similar to half duplex dual address mode but provides separate request pins for the receiver and transmitter.

Figures 4 through 7 describe operation of the DUSCC in the various DMA environments.

Table 10. DMA REQ AND ACK PINS FOR OPERATIONAL MODES

FUNCTION	HALF DUPLEX SINGLE ADDR DMA	HALF DUPLEX DUAL ADDR DMA	FULL DUPLEX SINGLE ADDR DMA	FULL DUPLEX DUAL ADDR DMA
RCVR REQ	RTXDRQ_N	RTXDRQ_N	RTXDRQ_N	RTXDRQ_N
TRAN REQ	Same as RCVR REQ	Same as RCVR REQ	TXDRQ_N	TXDRQ_N
RCVR ACK	RTXDAK_N	Normal read RCVR FIFO	RTXDAK_N	Normal read RCVR FIFO
TRAN ACK	Same as RCVR ACK	Normal write TRAN FIFO	TXDAK_N	Normal write TRAN FIFO

Table 10 summarizes pins used for the DMA request and acknowledge function for the transmitter and receiver for the different DMA modes.

The DMA request signals are functionally identical to the TXRDY and RXRDY status signals for each serial channel except that in DMA the signals are negated on the leading edge of the acknowledge signal when the subsequent transfer causes the FIFO to become full (transmitter request) or empty (receiver request). In non DMA operation TXRDY and RXRDY signals are negated only after the transfer is completed. The DMA read request can be programmed through OMR[3] be asserted either when any character is in the receive FIFO or only when the receive FIFO is full. Likewise, the DMA write request can be programmed through OMR[4] to be asserted either when the transmit FIFO is not full or only when the transmit FIFO is empty (the transmitter must be enabled for a DMA request to be asserted). The request signals are negated when the respective data transfer cycle is completed. When the serial channel is not operating in DMA mode, the request and acknowledge pins for the channel can be programmed for other functions (see pin descriptions).

DMA DONEN Operation

As an input, DONEN is asserted by the DMA controller concurrent with the corresponding DMA acknowledge to indicate to the DUSCC that the character being transferred into the TX FIFO is the last character of the transmission frame. In synchronous modes, the DUSCC can be programmed through TPR[4] to automatically transmit the frame termination sequence (e.g., FCS – FLAG in BOP mode) upon receipt of this signal.

As an output, DONEN is asserted by the DUSCC under the following conditions:

a. In response to the DMA acknowledge for a receiver DMA request if the FIFO'ed RECEIVED EOM status bit (RSR[7]) is set for the character being transferred.
b. In response to the DMA acknowledge for a transmitter DMA request if the counter/timer has been programmed to count transmitted characters and the terminal count has occurred.

Block Transfers Using DTACK

The DTACKN line may be used to synchronize data transfers to and from the DUSCC utilizing a 'wait' state. Either the receiver or the transmitter or both may be programmed for this mode of operation, independently for each channel, via CMR2[5:3].

In this mode, if the CPU attempts a write to the transmit FIFO and an empty FIFO position is not available, the DTACK line will remain negated until a position empties. The data will then be written into the FIFO and DTACKN will be asserted to signify that the transfer is complete.

Similarly, a read of an empty receive FIFO will be held off until data is available to be transferred. Potentially, this mode can cause the microcomputer system to hang up if, for example, a read request was made and no further data was available.

Timing Circuits

The timing block for each channel consists of a crystal oscillator, a bit rate generator (BRG), a digital phase locked loop (DPLL) and a 16-bit counter/timer (C/T) (see figure 8).

Crystal Oscillator

The crystal oscillator operates directly from a crystal (normally 14.7456MHz if the internal BRG is to be used) connected across the X1/CLK and X2/IDCN pins with a minimum of external components. If an external clock of the appropriate frequency is available, it may be connected to the X1/CLK pin. This signal is divided by two to provide the internal system clock (a maximum of 16MHz input is allowed).

Bit Rate Generator

The BRG operates from the oscillator or external clock and is capable of generating 16 bit rates. These are available to the receiver, transmitter, DPLL, and C/T. The BRG output is at 32X the base bit rate. Since all sixteen rates are generated simultaneously, each receiver and transmitter may select its bit rate independently. The transmitter and receiver timing registers include a 4-bit field for this purpose (TTR[3:0], RTR[3:0]).

Digital Phase Locked Loop

Each channel of the DUSCC includes a DPLL used in synchronous modes to recover clock information from a received data stream. The DPLL is driven by a clock at nominally 32 times the data rate. This clock can be programmed, via RTR (7:4), to be supplied from an external input, from the receiver BRG, from the C/T, or directly from the crystal oscillator.

The DPLL uses this clock, along with the data stream to construct a data clock which may then be used as the DUSCC receive clock, transmit clock, or both. The output of the DPLL is a square wave at 1X the data rate. The derived clock can also be programmed to be output on a DUSCC pin; only the DPLL receiver output clock is available at the TRXC pin. Four commands are associated with DPLL operation: Enter search mode, set FM mode, set NRZI mode, and disable DPLL. The commands are described in the command register description. Waveforms associated with the DPLL are illustrated in figure 9.

NRZI Mode Operation

This mode is used with NRZ and NRZI data encoding. With this type of encoding, the transitions of the data stream occur at the beginning of the bit cell. The DPLL has a six bit counter which is incremented by a 32X clock. The first edge detected during search mode sets the counter to 16 and begins operation. The DPLL output clock then rises at a count of 0 and falls at 16. Data is sampled on the rising edge of the clock. When a transition in the data stream is detected, the count length is adjusted by one or two counts, depending on the counter value when the transition occurs (see table 11). A transition detection at the rollover point (third column in figure 11) is treated as a transition occurring at zero count.

The count length adjustments cause the rising edge of the DPLL output clock to converge to the nominal center of the bit cell. In the worst case, which occurs when a DPLL pulse is coincident with the data edge, the DPLL converges after 12 data transitions.

For NRZ encoded data, a stream of alternating ones and zeros should be used as a synchronizing pattern. For NRZI encoded data, a stream of zeros should be used.

FM Mode Operation

FM operation is used with FM0, FM1, and Manchester data encoding. With this type of encoding, transitions in the data stream always occur at the beginning of the bit cell for FM0 and FM1, or at the center of the bit cell for Manchester. The DPLL 6-bit counter is incremented by a 32X clock. The first edge detected during search mode sets the counter to 16 and begins operation. The DPLL receiver clock then rises on a count of 8 and

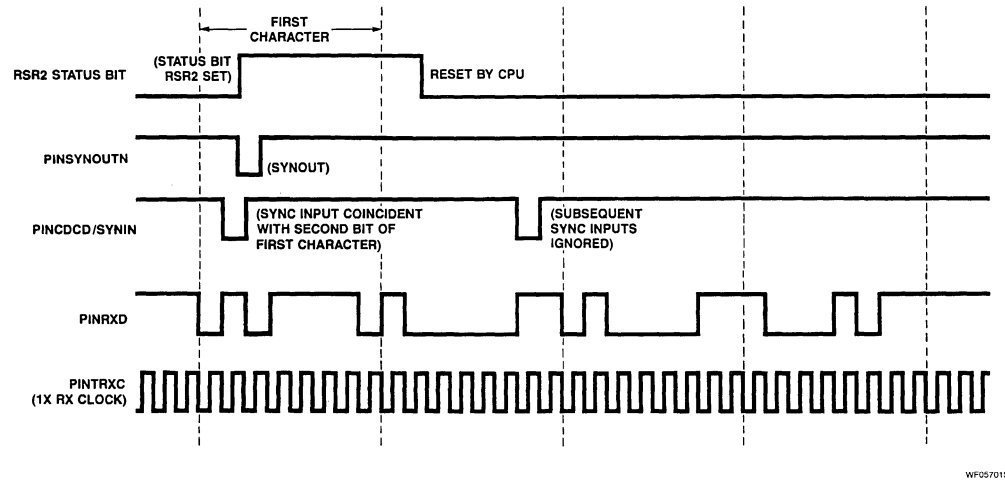


Figure 2. External Sync Mode

Table 11. NRZI MODE COUNT LENGTH

COUNT WHEN TRANSITION DETECTED	COUNT LENGTH ADJUST-MENT	COUNTER RESET AFTER COUNT REACHES
0 – 7	–2	29
8 – 15	–1	30
16 – 23	+1	32
24 – 30	+2	33
None detected	0	31

Table 12. FM MODE COUNT LENGTH

COUNT WHEN TRANSITION DETECTED	COUNT LENGTH ADJUST-MENT	COUNTER RESET AFTER COUNT REACHES
8 – 15	–1	30
16 – 23	+1	32
24 – 7	Disabled	
None detected	0	31

falls on 24. (The DPLL transmitter clock output falls on a count of 16. It rises on a count of 0 if a transition has been detected between counts of 16 and 23. For other cases, it rises 1/2 count of the 32X input clock sooner). This provides a 1X clock with edges positioned at the nominal centers of the two halves of the bit cell. The transition detection circuit is enabled between counts of 8 and 23 inclusive. When a transition is detected, the count length is adjusted by one, depending on when the transition occurs (see table 12).

If a transition is not detected for two consecutive data bits, the DPLL is forced into search mode and the DPLL error status bit (TRSR [3]) is asserted. This feature is disabled when the DPLL output is used only as the transmitter clock.

To prevent the DPLL from locking on the wrong edges of the data stream, an opening PAD sequence should be transmitted. For FM0, a stream of at least 16 ones should be sent initially. For FM1, a minimum stream of 16 zeros should be sent and for Manchester encoding the initial data stream should consist of alternating ones and zeros.

Counter/Timer

Each channel of the DUSCC contains a counter/timer (C/T) consisting of a 16-bit down counter, a 16-bit preset register, and associated control circuits. Operation of the counter/timer is programmed via the counter/timer control register (CTCR). There are also four commands associated with C/T operation, as described in the Command Description section.

The C/T clock source, clock prescaling, and operating mode are programmed via CTCR[2:0], CTCR[4:3], and CTCR[6] respectively. The preset register is loaded with a minimum of 2 by the CPU and its contents can be transferred into the down counter by a command, or automatically upon reaching terminal count if CTCR[6] is negated. Commands are also available to stop and start the C/T and to preset it to an initial value of FFFF. Counting is triggered by the falling edge of the clocking input. The C/T zero count status bit, ICTSR[6], is set when the C/T reaches the terminal count of zero and ICTSR[7] indicates whether the counter is currently enabled or not.

An interrupt is generated upon reaching zero count if CTCR[7] and the channel's master interrupt enable are asserted. The output of the C/T can be programmed to be output on the channel's RTXC or TRXC pin (via PCR[4:0]) as either a single pulse or a square wave, as programmed in CTCR[5]. The contents of the C/T can be read at any time by the CPU, but the C/T should normally be stopped before this is done. Several C/T operating modes can be selected by programming of the counter/timer control register. Typical applications include:

1. Programmable divider. The selected clock source, optionally prescaled, is divided by the contents of the preset register. The counter automatically reloads itself each time the terminal count is reached. In this mode, the C/T may be programmed to be used as the RX or TX bit rate generator, as the input to the DPLL, or it may be output on a pin as either a pulse or a square wave. The C/T interrupt should be disabled in this mode.
2. Periodic interrupt generator. This mode is similar to the programmable divider mode, except that the C/T interrupt is enabled, resulting in a periodic interrupt to the CPU.
3. Delay timer. The counter is preset from the preset register and a clock source, optionally prescaled, is selected. An in-

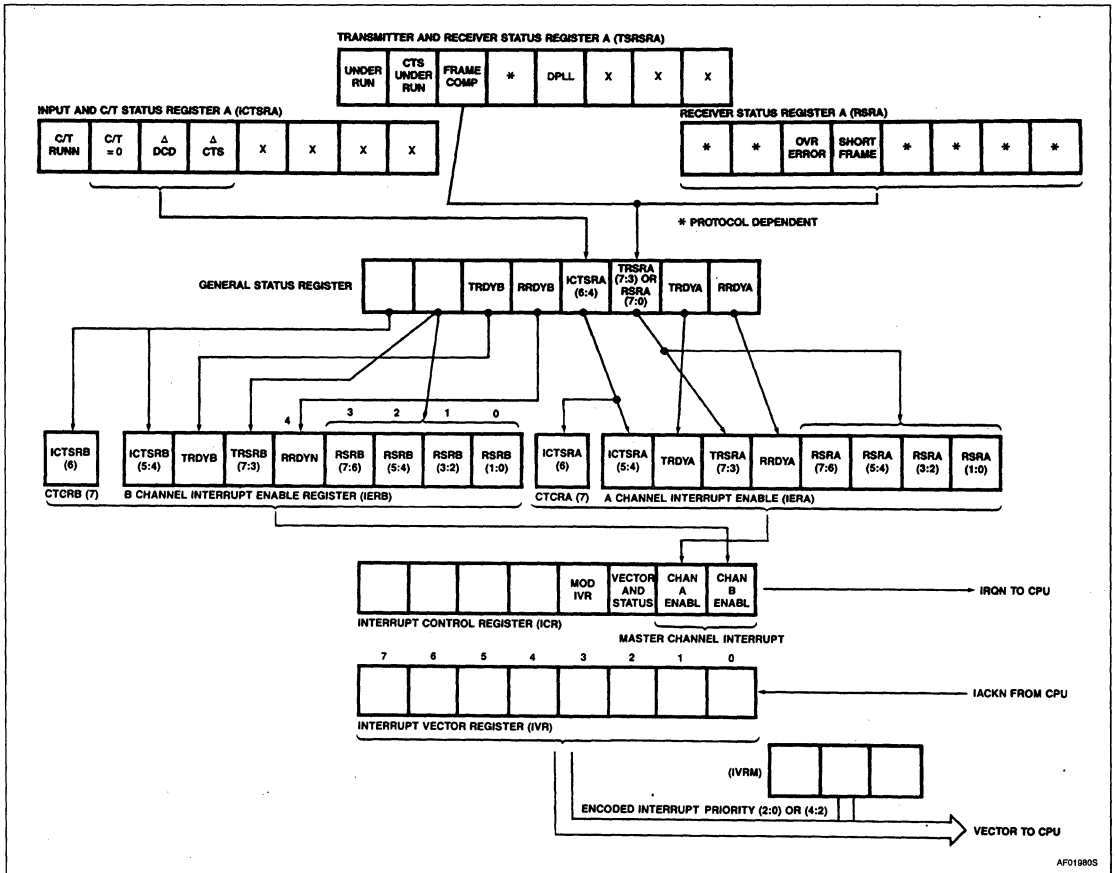


Figure 3. Interrupt Control and Status Register

terrupt is generated upon reaching terminal count. The C/T continues counting without reloading itself and its contents may be read by the CPU to allow additional delay past the zero count to be determined.

4. Character counter. The counter is preset to FFFF by command and the clock source becomes the internal signal used to control loading of the RX or TX characters. This operation is selected by CTCR_ [2:0]. The C/T counts characters loaded into the RXFIFO by the receiver or loaded into the transmit FIFO by the CPU respectively. The current character count can be determined by the CPU by reading the contents of the C/T and taking its ones complement. Optionally, a preset number may be loaded into the counter and an interrupt generated when the count is exhausted. When counting TX characters, the terminal count condition can be programmed through TPR_ [4] to cause an end of message sequence to be transmitted. When counting received characters, the FIFO'ed EOM status bit is asserted when the character which causes the count to go to zero is loaded into the receive FIFO. The channel's 'reset TX' or 'reset RX' commands have no effect on the operation of the C/T.
5. External event counter. The counter is preset to FFFF by command and an external clock source is selected. The current count can be determined by the CPU by reading the contents of the C/T and taking its ones complement. Optionally, a preset number may be loaded into the counter and an interrupt generated when the count is exhausted.
6. Bit length measurement. The counter is preset to FFFF by command and the X1/CLK/4 clock input gated by RXD mode (optionally prescaled) is programmed. The C/T starts counting when RXD goes low and stops counting when RXD goes high. At this time, status and an interrupt (if enabled) are generated. The resulting count in the counter can be read by the CPU to determine the bit rate of the input data. Normally this function is used for asynchronous operation.

Communication Channels A and B

Each communication channel of the DUSCC is a full duplex receiver and transmitter that supports ASYNC, COP, and BOP transmission formats. The bit rate clock for each receiver and transmitter can be selected independently to come from the bit rate generator, C/T, DPLL, or an external input (such as a modem generated clock).

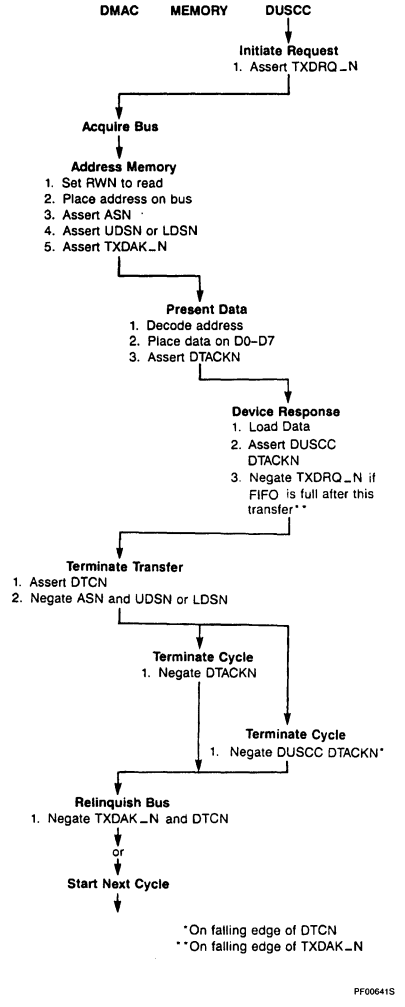


Figure 4. Transmitter DMA Request Operation — Single Address Mode

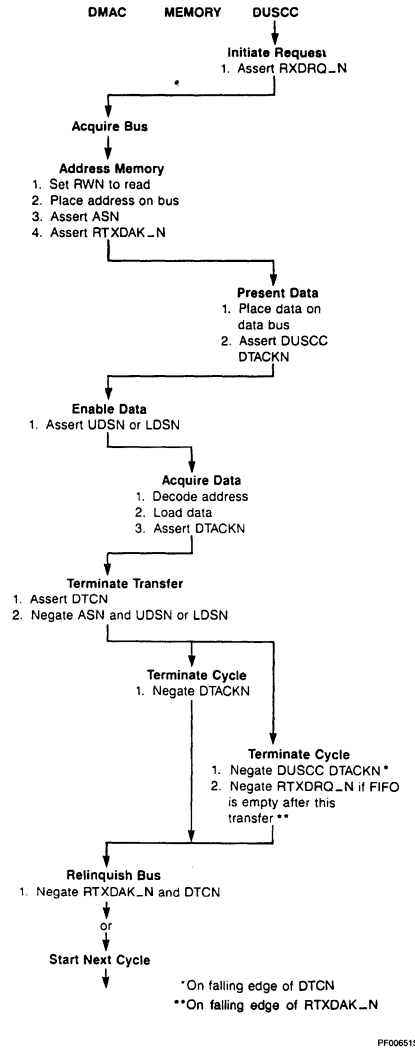


Figure 5. Receiver DMA Request Operation — Single Address Mode

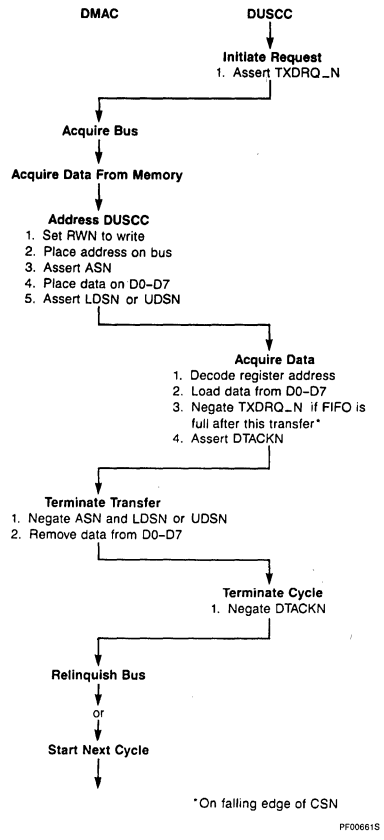


Figure 6. Transmitter DMA Request Operation — Dual Address Mode

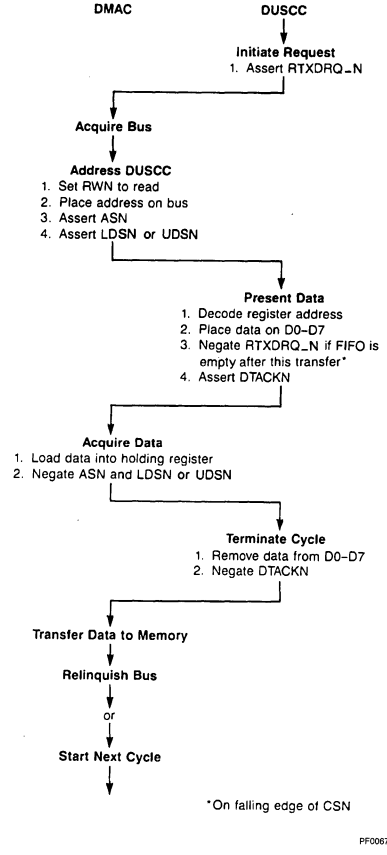
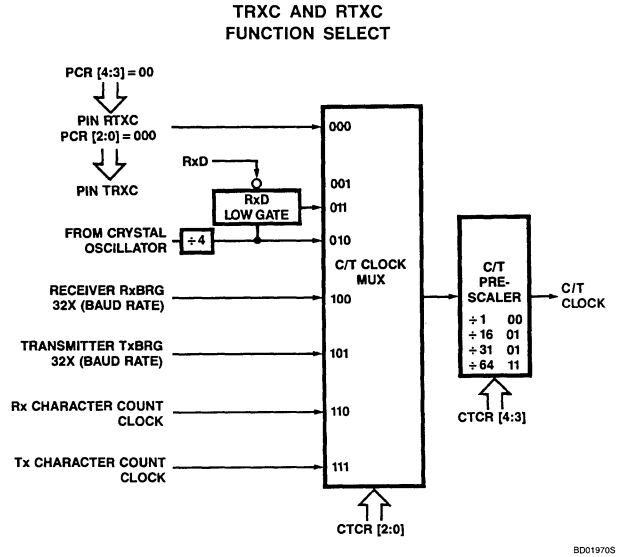
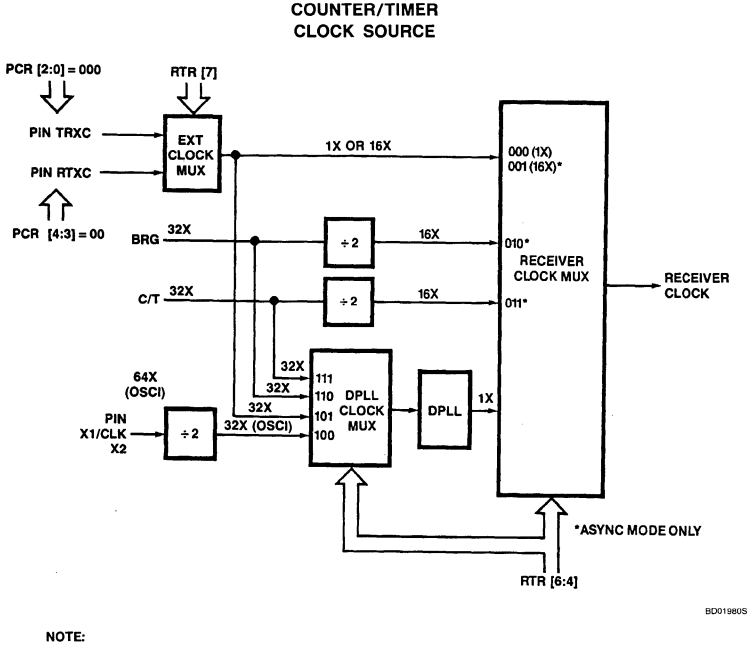


Figure 7. Receiver DMA Request Operation — Dual Address Mode

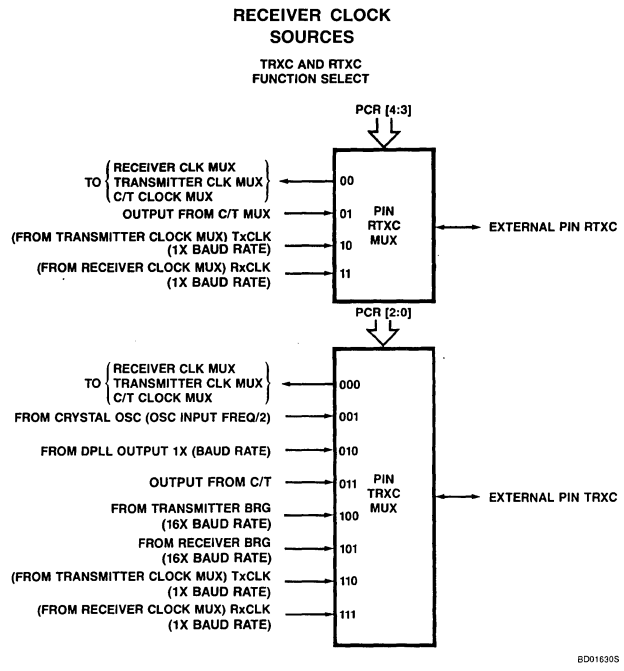
2





NOTE:
OSCI = crystal or EXT generator input

Figure 8. Timing Block



NOTE: OSCI = CRYSTAL OR EXT GENERATOR INPUT

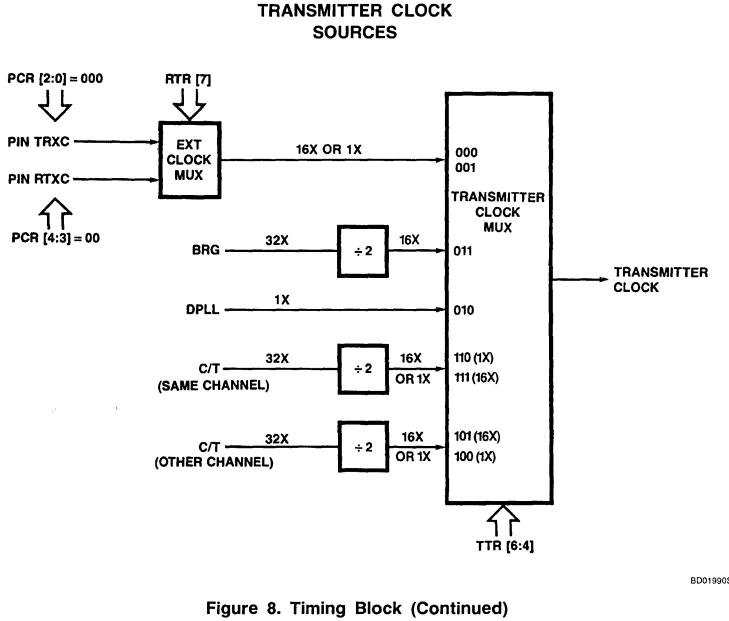


Figure 8. Timing Block (Continued)

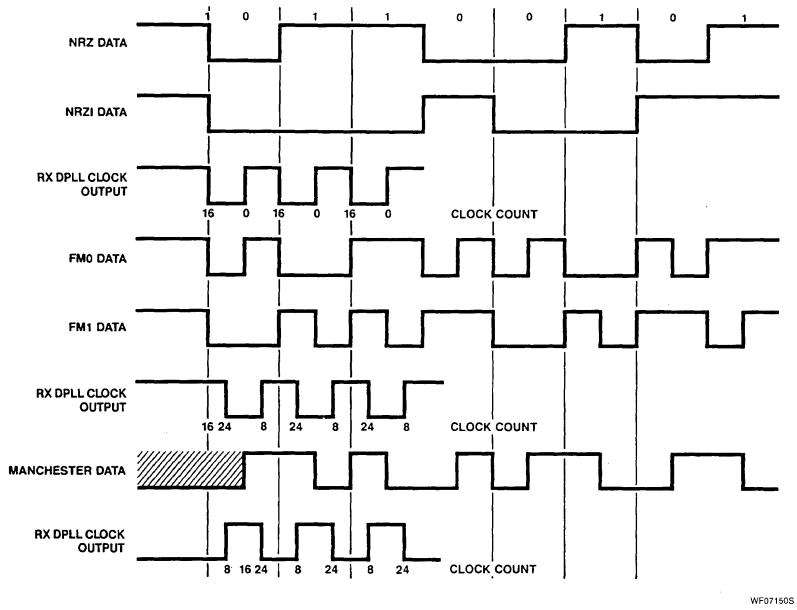


Figure 9. DPLL Waveforms

TRANSMITTER

Transmitter TXFIFO and TXRDY

The transmitter accepts parallel data from the data bus and loads it into the TXFIFO, which consists of four 8-bit holding registers. This data is then moved to the transmitter shift register, TXSR, which serializes the data according to the transmission format programmed. The TXSR is loaded from the TXFIFO, from special character logic, or from the CRC/LRC generator. The LSB is transmitted first, which requires right justification of characters by the CPU. TXRDY (GSR[5] or GSR[1]) and underrun (TRSR[7]) indicate the state of the TXFIFO. The TXFIFO may be addressed at any of four consecutive locations (see table 1) to allow use of multiple byte word instructions. A write to any valid address always writes data to the next empty FIFO location.

TXRDY is set when the transmitter is enabled and there is an empty position in the TXFIFO (OMR[4] = 0) or when the TXFIFO becomes empty (OMR[4] = 1). The CPU may reset TXRDY through a status reset write cycle. If this is done, it will not be reasserted until a character is transferred to the TXSR (OMR[4] = 0) or when the TXFIFO becomes empty again (OMR[4] = 1). The assertion of TXRDY, enabling of the IER [6] and the enabling of the channel master interrupt ICR [0] or [1] allow an interrupt to be generated.

If DMA operation is programmed, either RTXDRN (half duplex) or TXDRQN (full duplex) follows the state of TXRDY if the transmitter is enabled. These operations differ from normal ready in that the request signal is negated on the leading edge of the DMA acknowledge signal when the subsequent transfer causes the transmit FIFO to become full, while the TXRDY signal is negated only after the transfer is completed. Underrun status TRS[7] set indicates that one or more data characters (not PAD characters) have been transmitted and the TXFIFO and TXSR are both empty.

In 'wait on TX', a write to a full FIFO causes the write cycle to be extended until a FIFO position is available. DTACKN is asserted to acknowledge acceptance of the data. In non-wait modes, if an attempt is made to load data into a full TXFIFO, the TXFIFO data is preserved and the overrun data character(s) is lost. A normal DTACKN will be issued, and no indication of this occurrence is provided. The transmitter is enabled by the enable transmitter command. When the disable transmitter command is issued, the transmitter continues to operate until the TXFIFO becomes empty. The TXRDY does not become valid until the transmitter is enabled. Characters can be loaded into the FIFO while disabled. However, if the FIFO is full when the transmitter is enabled, TXRDY is not asserted.

TX RTS Control

If TX RTS CONTROL, TPR[3], is programmed, the channel's RTS output is negated five bit times after the last bit (stop bit in ASYNC mode) of the last character is transmitted. RTS is normally asserted and negated by writing to OMR_[0]. The assertion of TPR[3] causes RTS to be reset automatically (if the transmitter is not enabled) after all characters in the transmitter FIFO (if any) are transmitted and five bit times after the 'last character' is shifted out. This feature can be used to automatically terminate the transmission of a message as follows:

- Program auto-reset mode: TPR_[3] = 1.
- Enable transmitter.
- Assert RTS_N: OMR_[0] = 1.
- Send message.
- Disable transmitter after the last character is loaded into the TXFIFO.
- The last character will be transmitted and OMR_ [0] will be reset five bit times after the last bit, causing RTS_N to be negated. The TXD output will remain in the marking state until the transmitter is enabled again.

The 'last bit' in ASYNC is simply the last stop bit of the character. In BOP and COP, the last character is defined either explicitly by either appending it with TEOM or implicitly through the selection of the frame underrun control

sequence, TPR[7:6] (transmitter parameter register). Table 13 summarizes the relationship of the selected underrun sequence and the protocol mode.

TX CTS Operation

If CTS enable TX, TPR[2], is set, the CTSN input must be asserted for the transmitter to operate. Changes in CTSN while a character is being transmitted do not affect transmission of that character. However, if the CTS input becomes negated when TPR[2] is set and the transmitter is enabled and ready to start sending a new character, CTS underrun, TRSR[6], is asserted and the TXD output is placed in the marking (high) state. In ASYNC mode, operation resumes when CTSN is asserted again. In COP and BOP modes, the transmission of the message is terminated and operation of the transmitter will not resume until CTS is asserted and a TSOM or TSOMP command is invoked. Prior to issuing the command and retransmitting the message, the transmitter must be reset. After a change-of-state CTS is established by the input sampling circuits (refer to the description of ICTSR[4]), it is sampled by the TX controller 1½ bit times before each new character is serialized out of the TX shift register. (This is 2½ bits before the LSB of the new character appears on the TXD pin; there is an additional 1-bit delay in the transmitter data path due to the data encoding logic.)

TX Special Bit Pattern Transmission

The DUSCC provides features to transmit special bit patterns (see table 14).

The TXD pin is held marking after a hardware reset, a reset TX command, when the transmitter is not enabled, and during underrun/idle, if this feature is selected through TPR[7:5]. The TXD pin is also held marking if the transmitter is enabled, and the TXFIFO is empty (ASYNC), or if a TSOM or TSOMP command has not been issued (SYNC modes).

The following command bits can be appended to characters in the TXFIFO: TEOM, TDLE, exclude from CRC, and reset TXCRC. An invoked command(s) is appended to the next character loaded into the TXFIFO and follows the character through the FIFO until that character is ready to be loaded into the TXSR. The transmitter data path is shown in figure 10. The following describes the operation of the transmitter for the various protocols.

TX ASYNC Mode

Serialization begins when the TXFIFO data is loaded into the TXSR. The transmitter first sends a start bit, then the programmed number of bits/character (TPR[1,0]), a parity bit (if specified), and the programmed number of stop bits. Following the transmission of the stop bits, if a new character is not available in the TXFIFO, the TXD output goes to marking and the underrun condition (TRSR[7]) is set.

Transmission resumes when the CPU loads a new character into the TXFIFO or issues a send break command. The send break command clears the TXFIFO and forces a continuous space (low) on the TXD output after the character in TXSR (if any) is serialized. A send break acknowledge (TRSR[4]) is returned to the CPU to facilitate reassertion of the send break command in order to send an integral number of break characters. The send break condition is cleared when the reset TX or disable TX command is issued.

TX COP Modes

Transmitter commands associated with all COP modes are: transmit SOM (TSOM, transmit start of message), transmit SOM with PAD (TSOMP), transmit EOM (TEOM, transmit end of message), reset TX CRC, exclude from CRC, and transmit DLE.

A TSOM or send TSOMP command must be issued to start COP transmission. TSOM (without PAD) causes the TX CRC/LRC generator to be initialized and one or two SYN characters from S1R/S2R to be loaded into the TXSR and shifted out on the TXD output. A parity bit, if specified, is appended to each SYN character after the MSB. Send SOM acknowledge (TRSR[4]) is asserted when the SYN output begins. The user may reinvoke the command to cause multiple SYNs to be transmitted. If the command is not reinvoked and the TXFIFO is empty, SYN patterns continue to be transmitted until the TXFIFO is loaded. If data is present in the FIFO, the first character is loaded into the TXSR and serialization of the data begins. Note that the TXFIFO may be preloaded with data before the TSOM is issued.

The TSOMP command causes all characters in the TXFIFO (PAD characters) to be loaded into the TXSR and serialized if the TX is enabled. Unlike the transmit SOM without PAD command, data (non-PAD characters) cannot be preloaded into the TXFIFO. While the PAD is transmitted, parity is disabled and character length is automatically set to 8 bits regardless of the value in TPR[1:0]. When the TXFIFO becomes empty after the PAD, the TX CRC/LRC generator is initialized, the SYN character(s) are transmitted with optional parity appended, and send SOM acknowledge asserted. Operation then proceeds in the same manner as the TSOM command; the user has the option to invoke the TSOM command to cause multiple SYNs to be transmitted.

After the TSOM/TSOMP command is executed, characters in the TXFIFO are loaded into the TXSR and shifted out with a parity bit, if specified, appended after the MSB. If, after the opening SYN(s) and at least one data has

Table 13. ABORT SEQUENCE – PROTOCOL MODE

TPRA [7:6]	PROTOCOL	LAST CHARACTER
00	BOP	FLAG following either FCS (if selected) or last data character
	COP	Last byte of FCS before line begins SYN or MARKing
10	BOP	Abort sequence (11111111) prior to MARKing
	COP	Last byte of FCS before line begins SYN or MARKing
11	BOP	Abort sequence (11111111) prior to FLAG
	COP	First SYN of SYN sequence

Table 14. SPECIAL BIT PATTERNS

PROTOCOL	BIT PATTERN
ASYNC-BREAK	An all 0's character including parity bit (if specified) and stop bits. Used for send break command.
COP-SYN	Contained in S1R (single SYN mode) or in S1R/S2R (dual SYN modes). Used for TSOM and TSOMP commands and for non-transparent mode linefill and IDLE.
COP-DLE	Used for TDLE command and for BISYNC transparent mode linefill and to generate BISYNC control sequences.
COP-CRC	16/8 bits from the CRC/LRC accumulator used for TEOM command or for auto-EOM modes.
BOP-FLAG	01111110. Used for TSOM, TSOMP, and TEOM commands, for auto-EOM modes, and as an IDLE line fill.
BOP-ABORT	11111111. Used for send ABORT command or during TXFIFO underrun.
BOP-CRC	16 bits from the CRC accumulator used for TEOM command or for auto-EOM modes.
BOP/COP MARK	All 1's pattern on data line.

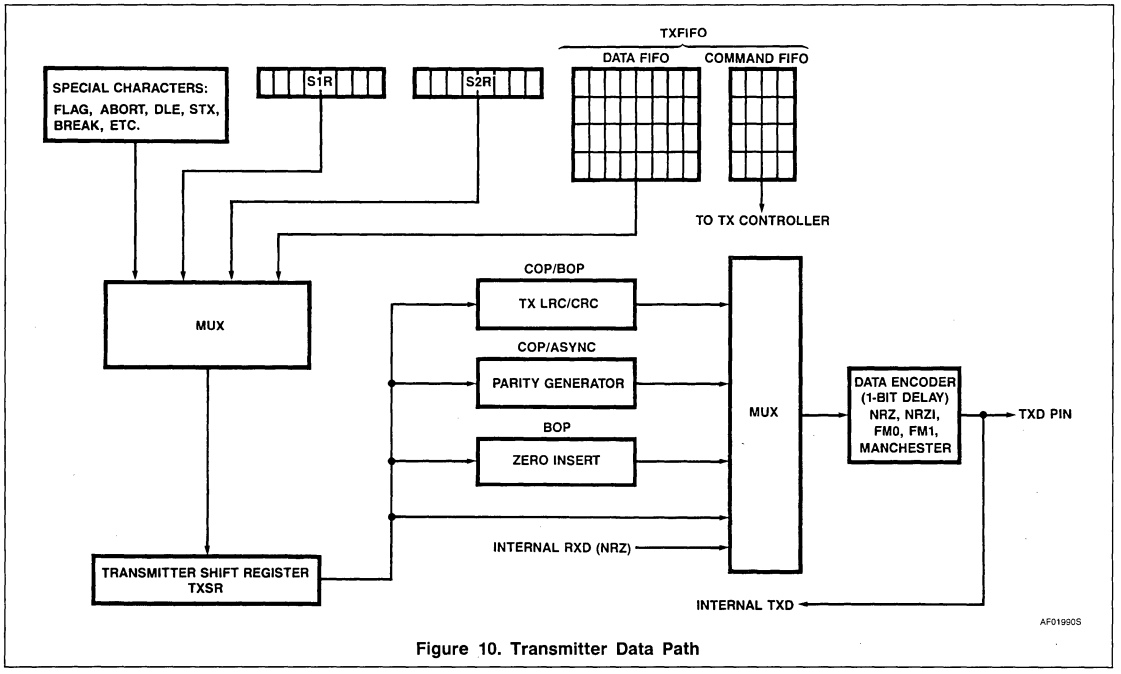


Figure 10. Transmitter Data Path

been transmitted, the TXFIFO is empty, a data underrun condition results and TRSR[7] is asserted. The transmitter's action on data underrun is determined by TPR[7:6] and the COP protocol. If TPR_[7:6] = '10', the transmitter linefills with MARK characters until a character is loaded into the FIFO. If TPR_[7:6] = '11' is selected, the transmitter linefills with SYN, SYN1-SYN2, or DLE-SYN1 for mono-sync, dual sync, and BISYNC transparent modes respectively. If TPR[7:6] = '00', the BCC characters are transmitted and frame complete (TRSR[6]) is set. TXD then assumes the programmed idle state (TPR[5]) of MARKs or SYN1/SYN1-SYN2.

Operation resumes with the transmission of a SYN sequence when a TSOM command is invoked. A TSOMP command is ignored unless the transmitter is disabled and then reenabled.

An appended TEOM command also terminates the frame as described above. It occurs after transmission of the character to which the TEOM is appended. The TEOM command can be explicitly asserted through the channel command register. If TPR[4] = '1', the TEOM is automatically appended to a character in DMA mode, if the DONEN input is asserted when that character is loaded into the TXFIFO, or if the counter/timer is counting transmitted characters when the character which causes the counter to reach zero count is loaded.

The TDLE command when appended to a character in the TXFIFO, causes the DLE character to be loaded into the TXSR and serialized before the TXFIFO character is loaded into the TXSR and serialized. This feature is particularly useful for BISYNC operation. The DLE character will be excluded from the CRC accumulation in BISYNC transparent mode (see below), but will be included in all other COP modes.

In BISYNC mode, transmission of a DLE-STX character sequence (either via a send TDLE command appended to the STX character, or via DLE and STX loaded into the TXFIFO) puts the transmitter into the transparent text mode of operation. In this mode, normally restricted character sequences can be transmitted as 'normal' bit sequences. The switch occurs after transmission of the two characters, so that the DLE and STX are included in the BCC accumulation. If the DLE-STX is to be excluded from the CRC, the user should issue a 'reset CRC' command prior to loading the next character.

Another method of excluding the two characters from the CRC is to invoke the 'exclude from CRC' command prior to loading the character(s) into the FIFO. While in transparent mode, the transmitter line fills with DLE-SYN1 and automatically transmits an extra DLE if it finds a DLE in the TX FIFO ('DLE stuffing'). The transmitter reverts to non-transparent mode when the frame complete status is set in TRSR[5].

CRC/LRC accumulation can be specified in all COP modes; the type of CRC is specified via CMR2[2:0]. The TSOM/TSOMP commands set the CRC/LRC accumulator to its initial state and accumulation begins with the first non-SYN character after the initial SYN(s) are transmitted. PAD characters are not subject to CRC accumulation. In non-BISYNC or BISYNC normal modes, all transmitted characters except linefill characters (SYNs or MARKs) are subject to accumulation. In BISYNC transparent mode, odd (stuffed) DLEs and the DLE-SYN1 linefill are excluded from the accumulation. Characters can be selectively excluded from the accumulation by invoking the 'exclude from CRC' command prior to loading the character into the FIFO.

Accumulation stops when transmission of the first character of the BCC begins. The CPU can set the accumulator to its initial state prior to the transmission of any character by using the appended reset CRC command. The CRC generator is also automatically initialized after the EOM is sent.

TX BOP Modes

Transmitter commands associated with BOP modes are TSOM, TSOMP, TEOM, and trans-

mit ABORT (TABRK). The TSOM and TSOMP commands are identical to COP modes except that a FLAG character (01111110) is used as the start of message sequence instead of the SYNs, and FLAG(s) that continue to be sent until the TXFIFO is loaded. There is no zero insertion (see below) during transmission of the PAD characters, and they are not preceded by a FLAG or accumulated in the CRC. Character length is automatically set to 8 bits regardless of TPR[1:0].

The first characters loaded into the TXSR from the TXFIFO are the address and control fields, which have fixed character lengths of eight bits. The number of address field bytes is determined by CMR1[4:3]. If extended address field is specified, the field is terminated if the first address octet is H'00' or if the LSB of the octet is a 1. The number of control field bytes is selected by CMR1_[5]. If any information field characters follow the control field (forming an I field), they are transmitted with the number of bits per character programmed in TPR[1:0]. The TEOM command can be appended to the last character either explicitly or automatically as described for COP mode. When the character with the appended TEOM is loaded from the TXFIFO, it is transmitted with the character length specified by OMR[7:5]. In this way, a residual character of 1 – 8 bits is transmitted without requiring the CPU to change the TX character length for this last character.

After the opening FLAG and first address octet have been transmitted, an underrun occurs (TRSR[7] = 1) if the TXFIFO is empty when the transmitter requires a new character. The underrun control bits (TPR[7:6]) determine whether the transmitter line fills with either ABORT-MARKs, ABORT-FLAGs (see below), or ends transmission with the 'normal' end of message sequence.

EOM on underrun is functionally similar to EOM due to an appended TEOM command. If the EOM is due to underrun, the normal character length applies to the last data character. After the last character is transmitted, the FCS (inverted CRC) and closing FLAG are sent, frame complete (TRSR[5]) is set, and the TX CRC is initialized. If the TXFIFO is empty after the closing FLAG has been sent, TXD will assume the programmed idle state of FLAGs or MARKs (TPR[5]) and wait for a character to be loaded into the FIFO or for a TSOM command to be issued. If the TXFIFO is not empty at that time, the TXFIFO data will be loaded into the TXSR and serialized. In that case, the closing FLAG is the opening FLAG of the next frame.

The user can control the number of FLAGs between frames by invoking the TSOM command after frame complete is asserted. The DUSCC then operates in the same manner as for transmission of multiple FLAGs at the beginning of a frame. When the command is no longer reinvoked, transmission of the TXFIFO data will begin. If the FIFO is empty, FLAGs continue to be transmitted.

The DUSCC provides automatic zero insertion in the data stream to prevent erroneous transmission of the FLAG sequence. All data characters loaded into the TXSR from the TXFIFO and characters transmitted from the CRC generator are subject to zero insertion. For this feature a zero is inserted in the serial data stream each time five consecutive ones (regardless of character boundaries) have been transmitted.

A send ABORT command clears the TXFIFO and inserts an ABORT character of eight ones (not subject to zero insertion) into the TXSR for transmission after the current character has been serialized. A send abort ack (TRSR[4]) facilitates reassertion of send abort by the user to guarantee transmission of multiple abort characters. This feature can be used to send the 15-ones idle sequence. The transmitter sends either marks or FLAGs after the abort character(s) has been transmitted, depending on TPR[7:6]. Operation resumes with the transmission of a FLAG when a TSOM command is invoked. A TSOMP command is ignored unless the transmitter is disabled and then reenabled.

CRC accumulation can be specified in all BOP modes. The type of CRC is specified via CMR2[2:0], and is normally selected as CRC-CCITT preset to ones, although any option is valid. Note that LRC8 option is not allowed in BOP modes.

The TSOM/TSOMP command sets the CRC accumulator to its initial state and accumulation begins with the first address octet after the initial FLAG(s). Accumulation stops when transmission of the first character of the FCS begins. The CPU can set the accumulator to its initial state prior to the transmission of any character by using the appended reset CRC command and can exclude any character from the accumulation by use of the exclude from CRC command, but these features would not normally be used in BOP modes. The CRC generator is also automatically initialized after the EOM or an ABORT are sent.

TXBOP Loop Mode

The loop modes are used by secondary stations on the loop, while the primary station operates in the BOP primary mode. Both the transmitter and receiver must be enabled. Loop operation is initiated by issuing the 'go on-loop' command. The receiver looks for the receipt of seven contiguous ones and then asserts the LC_N output to cause external loop control hardware to put the DUSCC into the loop, with the TXD output echoing the RXD input with a three bit time delay. The echoing process continues until a go active on poll (GAP) command is invoked. The DUSCC then looks for receipt of an EOP bit pattern (a zero followed by seven ones, 11111110) and changes the last one of the EOP into a zero making it an opening FLAG. Loop sending (TRSR[6]) is asserted at that same time. The action of the transmitter after sending the initial FLAG depends on the status of the transmit FIFO.

If the transmit FIFO is not empty, a normal frame transmission begins. The operation is then similar to normal BOP operation with the following differences:

1. An ABORT command, an underrun, or receipt of the turnaround sequence (H'00') or a FLAG cause the transmitter to cease operation and to revert to echoing the RXD input with a three bit time delay. A new transmission cannot begin until the GAP command is reinvoked and a new EOP sequence is received.
2. Subsequent to sending the EOM sequence of FCS-FLAG, the DUSCC examines the internal GAP flip-flop. If it is not set (having been reset by the 'reset GAP' command), the DUSCC reverts to echoing the received data. If the internal GAP flip-flop is still set, transmission of a new frame begins, with the user having control of sending multiple FLAGs between frames by use of the 'send SOM' command. If the FIFO is empty at this time, the DUSCC continues to send FLAGs until the data is loaded into the FIFO or until GAP is reset. If the latter occurs, it reverts to echoing RXD.

   When the DUSCC reverts to echoing RXD in any of the above cases, the last transmitted zero and seven ones will form an EOP for the next station down the loop.

   If the TXFIFO is empty when the EOP is recognized, the transmitter continues to send FLAGs until there is data in the FIFO. If a turnaround sequence or the reset GAP command is received before the FIFO is loaded, the transmitter switches to echoing RXD without any data transmission. Otherwise a frame transmission begins as above when a character is loaded into the FIFO. The mechanism provides time for the CPU to examine the received frame (the frame preceding the EOP) to determine if it should respond or not, while holding its option to initiate a transmission.

   Termination of operation in the loop mode should be accomplished by use of the 'go off-loop' command. When the command is invoked, the DUSCC looks

for the receipt of eight contiguous ones. It then negates the LC_N output to cause the external loop control hardware to remove the DUSCC from the loop without affecting operation of other units remaining on the loop.

RECEIVER

The receiver data path includes two 9-bit holding registers, HSRH and HSRL, an 8-bit character comparison register, CCSR, two synchronizing flip flops, a receiver shift register, RXSR, the programmable SYN comparison registers, S1R and S2R, and BISYNC character comparison logic. The DUSCC configures this circuitry and utilizes it according to the operational mode selected for the channel through the two mode registers CMR1 and CMR2. For all data paths, character data is assembled according to the character bit count, in the RXSR, and is moved to the RXFIFO with any appended statuses when assembly is complete. Figure 1 depicts the four data paths created in the DUSCC for the previous protocols.

Receiver RXFIFO, RXRDY

The receiver converts received serial data on RXD (LSB first) into parallel data according to the transmission format programmed. Data is shifted through a synchronizing flip flop and one or more shift registers, the last of which is the 8-bit receiver shift register (RXSR). Bits are shifted into the RXSR on the rising edge of each 1X receive clock until the LSB is in RXSR[0]. Hence, the received character is right justified, with all unused bits in the RXSR cleared to zero. A receive character length counter generates a character boundary signal for synchronization of character assembly, character comparisons, break detection (ASYNC), and RXSR to RXFIFO transfers (except for BOP residual characters). During COP and BOP hunt phases, the SYN/FLAG comparison is made each receive bit time, as abort, and idle comparisons in BOP modes.

An internal clock from the BRG, the DPLL or the counter/timer, or an external 1x or 16x clock may be used as the receiver clock in ASYNC mode. The BRG or counter/timer cannot be used directly for the receiver clock in synchronous modes, since these modes require a 1X receive clock that is in phase with the received data. This clock may come externally from the RTXC or TRXC pins, or it may be derived internally from the DPLL. Encoded data is internally converted to NRZ format for the receiver circuits by using clock pulses generated by the DPLL.

When a complete character has been assembled in the RXSR, it is loaded into the receive FIFO with appended status bits. The most significant data bits of the character are set to zero if the character length is less than eight bits. In ASYNC and COP modes the user may select, via RPR[3], whether the data transferred to the FIFO includes the received parity bit or not. The receiver indicates to the CPU or DMA controller that it has data in the FIFO by asserting the channel's RXRDY status bit (GSR[4] or GSR[0]) and, if in DMA mode, the corresponding receiver DMA request pin.

The RXFIFO consists of four 8-bit holding registers with appended status bits for character count complete indications (all modes), character compare indication (ASYNC), EOM indication (BISYNC/BOP), and parity, framing, and CRC errors. Data is loaded into the RXFIFO from the RXSR and extracted (read) by the CPU or DMA controller via the data bus. An RXFIFO read creates an empty RXFIFO position for new data from the RXSR.

RXRDY assertion depends on the state of OMR[3]:

1. If OMR[3] is 0 (FIFO not empty), RXRDY is asserted each time a character is transferred from the receive shift register to the receive FIFO. If it is not reset by the CPU, RXRDY remains asserted until the receive FIFO becomes empty, at which time it is automatically negated. If it is reset by the CPU, it will remain negated, regardless of the current state of the receive FIFO, until a new character is transferred from the RXSR to the RXFIFO.
2. If OMR[3] is 1 (FIFO full), RXRDY is asserted:
   a. when a character transfer from the receive shift register to the receive FIFO causes it to become full
   b. when a character with a tagged EOM status bit is loaded into the FIFO (BISYNC or BOP) regardless of RXFIFO full condition.
   c. when the counter/timer is programmed to count received characters and the character which causes it to reach zero count is loaded into the FIFO (ICTSR[6]).
   d. when the beginning of a break is detected in ASYNC mode regardless of the RXFIFO full condition.

If it is not reset by the CPU, RXRDY remains asserted until the FIFO becomes empty, at which time it is automatically negated. If it is reset by the CPU, it will remain negated regardless of the current state of the receive FIFO, until it is asserted again due to one of the above conditions.

The assertion of RXRDY causes an interrupt to be generated if IER[4] and the channel's master interrupt enable (ICR[0] or ICR[1]) are asserted.

When DMA operation is programmed, the RXRDY status bit is routed to the DMA control circuitry for use as the channel receiver DMA request. Assertion of RXRDY results in assertion of RTXDRQN output.

Several status bits are appended to each character in the RXFIFO. When the FIFO is read, causing it to be 'popped', the status bits associated with the new character at the top of the RXFIFO are logically OR'ed into the RSR. Therefore, the user should read RSR before reading the RXFIFO in response to RXRDY activation. If character-by-character status is desired, the RSR should be read and cleared each time a new character is received. The user may elect to accumulate status over several characters or over a frame by clearing RSR at appropriate times. This mode would normally also be used when operating in DMA mode. If the RXFIFO is empty when a read is attempted, and wait mode as specified in CMR2[5:3], is not being used, a H'FF' is output on the data bus.

In all modes, the DUSCC protects the contents of the FIFO and the RXSR from overrun. If a character is received while the FIFO is full and a character is already in the RXSR waiting to be transferred into the FIFO, the overrunning character is discarded and the OVERRUN status bit (RSR[5]) is asserted. If the overrunning character is an end-of-message character, the character is lost but the FIFO'ed EOM status bit will be asserted when the character in the RXSR is loaded into the FIFO.

Operation of the receiver is controlled by the enable receiver command. When this command is issued, the DUSCC goes into the search for start bit state (ASYNC), search for SYN state (COP modes), or search for FLAG state (BOP modes). When the disable receiver command is issued, the receiver ceases operation immediately. The RXFIFO is cleared on master reset, or by a reset receiver command. However, disabling the receiver does not affect the RXFIFO, RXRDY, or DMA request operation.

Receiver DCD and RTS Controls

If DCD enable RX, RPR[2], is asserted, the DCD input must be asserted for the receiver to operate. If RPR[2] is asserted and the sampling circuit detects that the DCD input has been negated, the receiver ceases operation immediately. Operation resumes when the sampled DCD is asserted again. A change of state detector is provided on the DCD input of each channel. The required duration of the DCD level change is described in the discussion of ICTSR[5]. The user may program a change of state to cause an interrupt to be generated (master interrupt enable ICR[0] or [1] and IER [7] must be set) so that appropriate action can be taken.

In ASYNC mode, RPR[4] can be programmed to control the deactivation of the RTS_N output by the receiver. RTS_N can be manually asserted and negated by writing to OMR_[0]. However, the assertion of RPR[4] causes RTS to be negated automatically upon receipt of a valid start bit if the channel's receive FIFO is already full. When this occurs, the RTSN negated status bit, RSR[6], is set. This may be used as a flow control feature to prevent overrun in the receiver by using the RTS_N output signal to control the CTS_N input of the remote transmitter. The new character will be assembled in the RXSR, but its transfer to the FIFO will be delayed until the CPU reads the FIFO, making the FIFO position available for the new character.

Once enabled, receiver operation depends on channel protocol mode. The following describes the receiver operation for the various protocols:

RX ASYNC Mode

When first enabled, the receiver goes into the search for start bit state, looking for a high to low (mark to space) transition of the start bit on the RXD input. If a transition is detected, the state of the RXD pin is sampled again each 16X clock for 7½ clocks (16X clock mode) or at the next rising edge of the bit time clock (1X clock mode). If RXD is sampled high, the start bit is invalid and the search for a valid start bit begins again.

If RXD is still low, a valid start bit is assumed and the receiver continues to sample the input at one bit time intervals (16 periods of the 16X RX clock; one period of the 1X RX clock) at the theoretical center of the bit, until the proper number of data bits and the parity bit (if specified) have been assembled, and the first stop bit has been detected.

The assembled character is then transferred to the RXFIFO with appended parity error (if parity is specified) and framing error status bits. The DUSCC can be programmed to compare this character to the contents of S1R. The appended character compare status bit, RSR[7], is set if the data matches and there is no parity error.

After the stop bit is sampled, the receiver will immediately look for the next start bit. However, if a non-zero character was received without a stop bit (i.e. framing error) and RXD remains low for one half of the bit period after the stop bit was sampled, then the receiver operates as if a new start bit transition had been detected at that point (one half bit time after the stop bit was sampled).

If a break condition is detected (RXD low for entire character time including optional parity and first stop bit), only one character consisting of all zeros will be loaded into the RXFIFO and break start detect, RSR[2], will be set. The RXD input must return to a high condition for at least one half of a bit time (16X clock mode) or for one bit time (1X clock mode) before the break condition is terminated and the search for the next start bit begins. At that time, the break end detect condition, RSR[3], is set.

RX COP Modes

When the receiver is enabled in COP modes, it first goes into the SYN hunt phase, testing the received data each bit time for receipt of the appropriate SYN bit pattern, plus parity if specified, to establish character boundaries. Receipt of the SYN bit pattern terminates hunt phase and places the receiver in the data phase, in which all leading SYNs are stripped and the RXFIFO begins to load starting with the first non-SYN character. In COP single SYN protocol mode, S1R contains the SYN character required to establish character synchronization. In COP dual SYN and BISYNC protocol modes, S1R and S2R contain the first and second SYN characters, respectively, required to establish character synchronization. The SYN character length is the same as the character length programmed in RPR[1:0], plus the parity bit if parity is specified. SYN characters received with a parity error, when parity is specified, are considered invalid and will not cause synchronization to be achieved.

If external synchronization is programmed (RPR[4] = 1), the internal SYN detection and special character recognition logic are disabled and receipt of SYN characters is not required. A pulse on the SYNI input pin will establish character synchronization and terminate hunt phase. The SYNI pin is ignored after the first input on the SYNIN pin is received. The receiver must be disabled and then reenabled to resynchronize or to return to normal mode. This must be programmed in conjunction with CMR1[2:0] = 110. Refer to the description of RPR[4] for further details.

The SYN detect status bit, RSR[2], is set whenever SYN1, SYN1-SYN2, or DLE-SYN1 is detected for single SYN, dual SYN/BISYNC normal, and BISYNC transparent modes, respectively, and the SYNOUT pin will go active for one receive clock period one bit time after SYN detection in HSRH/HSRL. After character sync has been attained, the receiver enters the data phase and assembles characters in the RXSR, beginning with the first non-SYN character, with the least significant bit received first. It computes the BCC if specified, checks parity if specified, and checks for overrun errors.

The operation of the BCC (CRC/LRC) logic depends on the particular COP mode in use. The BCC is initialized upon first entering the data phase. For non-BISYNC modes, all received characters after entering data phase are included in the BCC computation, except for leading SYNs and SYNs which are specified to be stripped by RPR[7]. As each received character is transferred from the RXSR to the FIFO, the current value of the BCC characters is checked and the CRC ERROR status bit (RSR[1]) is set if the value of the CRC remainder is not the expected value. The EOM status bit, RSR[7], is not set since there is no defined end-of-message character. The receiver computes the BCC for text messages automatically when operating in BISYNC protocol mode.

BISYNC Features

The DUSCC provides support for both BISYNC normal and transparent operations. The following summarizes the features provided. Both EBCDIC and ASCII text messages can be handled by the DUSCC as selected by CMR1[5]. The receiver has the capability of recognizing special characters for the BISYNC protocol mode (see table 15).

All sequences in table 15, except SOH and STX, when detected explicitly cause a status to be affected. The following describes the conditions when this occurs.

The first character received when entering data phase for a header or text message should be an SOH, an STX, or a DLE-STX two character sequence. Receipt of any of these initializes the CRC generator and starts the CRC accumulation. The SOH places the receiver in header mode, receipt of the STX places it in text mode, and receipt of the DLE-STX sequence (at any time) automatically places the receiver in transparent mode and sets the XPNT mode status bit, TRSR[0]. There is no explicit status associated with SOH and STX. If any other characters are received when entering the data phase, the message is treated as a control message and will not be accumulated in CRC.

After the data phase is established, the receiver searches the data stream for an end of message control character(s):

Header field: ENQ, ETB, or ITB

Normal text field: ENQ, ETX, ETB, or ITB

Transparent text field: DLE-ENQ, DLE-ETX, DLE-ETB, or DLE-ITB

Control message field: EOT, NAK, ACK0, ACK1, WACK, RVI or TTD

Detection of any one of these sequences causes the EOM status bit, RSR[7], to be set. Also if RPR[5] is set and the receiver does not detect a closing PAD (four 1's) after the 'EOT' or 'NAK', the PAD error status bit, RSR[6], is set. When the abort sequence ENQ or DLE-ENQ is detected, the character is tagged with an EOM status and transferred to the FIFO, but the appended CRC error

status bit should be ignored. For the other EOM control sequences, the receiver waits for the next two bytes (the CRC bytes) to be received, checks the value of the CRC generator, and tags the transferred character with a CRC error, RSR[1], if the CRC remainder is not correct. See figure 11 for an example of BCC accumulation in various BISYNC messages.

The CRC bytes are normally not transferred to the FIFO, unless the transfer FCS to FIFO control bit, RPR[6], is asserted. In this case the EOM and CRC error status bits will be tagged onto the last byte of the last FCS byte instead of to the last character of the message. After detecting one of the end-of-message (EOM) character sequences and setting RSR[7], the receiver automatically goes into auto hunt mode for the SYNC characters and PAD check if RRP[5] is set.

SYN Pattern Stripping

Leading SYNs (before a message) are always stripped and excluded from the FCS, but SYN patterns within a message are treated by the receiver according to the RPR[7] bit. SYN character patterns are defined for the various COP modes as follows:

COP single SYN mode – SYN1

COP dual SYN mode – SYN1, and SYN2 when immediately preceded by SYN1.

BISYNC normal mode – SYN1, and SYN2 when immediately preceded by SYN1. SYN1 is always stripped, even if it is not followed by SYN2 when stripping is selected.

BISYNC transparent mode – DLE – SYN1, where the DLE is the last of an odd number of consecutive DLEs.

0 Strip only RPR[7] leading the SYN and do not accumulate in FCS.

1 Strip all SYNs. Additionally, strip odd DLEs when operating in BISYNC transparent mode. Do not accumulate stripped characters in FCS.

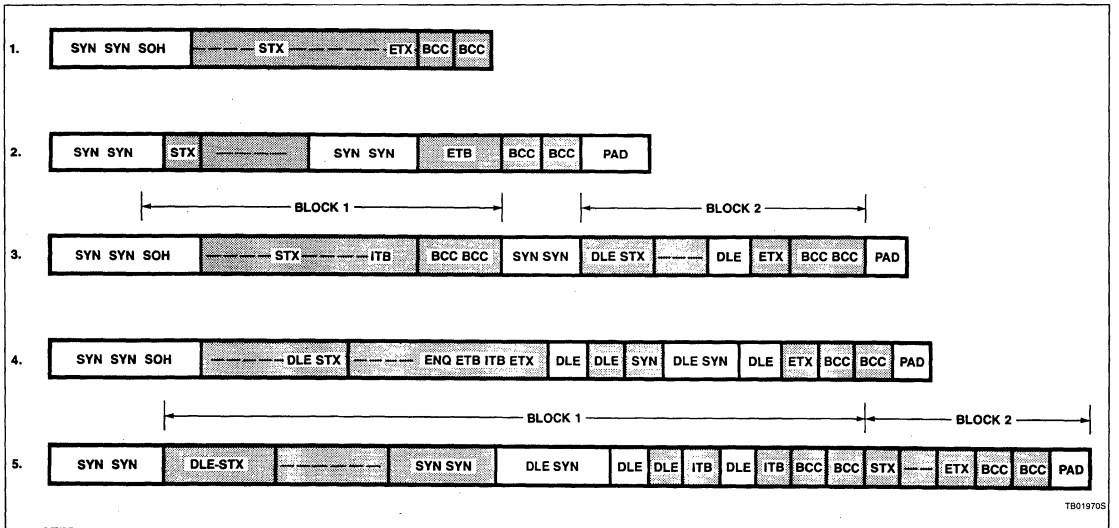
Processing of the SYN patterns is determined by the RPR[7] bit, the COP mode, and the position of the pattern in the frame. This is summarized in table 16.

The value of the RPR[7] field does not affect the setting of the SYN DETECT status bit, RSR[2], and the generation of a SYNOUT pulse when a SYN pattern is received.

RX BOP Mode

In BOP protocol mode, the receiver may be in any one of four phases: hunt phase, address field (A) phase, control field (C) phase, or information field (I) phase. The character length for the A and C phases is always 8 bits. The I field character length is specified in RPR[1:0].

After an enable receiver command is executed, the receiver enters hunt phase, in which a comparison for the string (01111110) is done every RX bit time. The FLAG delineates the beginning (and end) of a received frame and establishes the character boundary. Each FLAG match in CCSR causes the FLAG detect status bit (RSR[2]) to be set and SYNOUT N pin to be activated one bit time



NOTES:
1. The BCC accumulator is stopped after the end of message character sequence is accumulated.
2. ENQ (DLE – ENQ) in a text message should be treated as an abort.
3. Opening SYNs will be stripped by the receiver.

Figure 11. Example of BCC Accumulation in Various BISYNC Messages

Table 15. BISYNC FEATURES

BISYNC – Single Character Sequences			
Sequence	**ASCII**	**EBCDIC**	**Description**
SOH	H'81	H'01	Start of header
STX	H'82	H'02	Start of text
ETX	H'83	H'03	End of text
EOT	H'84	H'37	End of transmission
ENQ	H'85	H'2D	Enquiry
DLE	H'90	H'10	Data link escape
NAK	H'95	H'3D	Negative ack
ETB	H'97	H'26	End of transmission block
ITB	H'9F	H'1F	End of intermediate transmission block
BISYNC – Two Character Sequences			
Sequence	**ASCII**	**EBCDIC**	**Description**
ACK0	H'90,B0	H'10,70	Acknowledge 0
ACK1	H'90,B1	H'10,61	Acknowledge 1
WACK	H'90,BB	H'10,6B	Wait before transmit positive ack
RVI	H'90,BC	H'10,7C	Reverse interrupt
TTD	H'82,85	H'02,2D	Temporary text delay
BISYNC – (Transparent Text Mode) – Two Character Sequences			
Sequence	**ASCII**	**EBCDIC**	**Description**
DLE-ENQ	H'90,85	H'10,2D	Enquiry
DLE-ITB	H'90,9F	H'10,1F	End of intermediate transmission block
DLE-ETB	H'90,97	H'10,26	End of transmission block
DLE-ETX	H'90,83	H'10,03	End of text
DLE-STX	H'90,82	H'10,02	Start of transparent text mode

Table 16. SYN PATTERN PROCESSING

MODE	RPR [7]	LEADING SYNs	WITHIN A MESSAGE
BISYNC	0	no FCS no FIFO	no FCS Pattern into FIFO
	1	no FCS no FIFO	no FCS no FIFO
COP	0	no FCS	Accumulate in FCS
		no FIFO	Pattern into FIFO
	1	no FCS no FIFO	no FCS no FIFO

later for one receive clock period. FLAGs with an overlapping zero will be detected. All FLAGs are deleted from the data stream.

Once a FLAG has been detected, the receiver will exit hunt phase and enter address phase. The handling of the address field is determined by the values programmed in CMR1[2:0], which selects one of the BOP modes. The BOP secondary address modes are selected by CMR1 [4:3] and function as in the description that follows.

Single Octet Address
For receive, the address comparison for a secondary station is made on the first octet following the opening FLAG. A match occurs if the first octet after the FLAG matches the contents of S1R, or if all parties address (RPR[3]) is asserted and the first octet is equal to H'FF'.

Dual Octet Address
For receive, the address comparison for a secondary station is made on the first two octets following the opening FLAG.

A match occurs if the first two octets after the FLAG match the contents of S1R and S2R respectively, or if all parties address (RPR[3]) is asserted and the first two octets are equal to H'FF, FF'.

Dual Address with Group Mode
For receive, the address comparison for a secondary station is made on the first two octets following the opening FLAG. A match occurs for one of three possible conditions. If the first two octets after the FLAG match the contents of S1R and S2R respectively, or if the first octet is H'FF' and the second matches the contents of S2R (group mode), or when all parties address (RPR[3]) is asserted and the first two octets are equal to H'FF, FF'. The second condition (group mode) allows a selected group of stations to receive a message.

Extended Address Mode
Extend address field to the next octet if the LSB of the current address octet is zero. Address field is terminated if the LSB of the address is a one. The address field will be terminated after the first octet if the null address H'00' is received/transmitted as the first address octet. For this mode the receiver does not perform an address comparison (all received characters after the opening FLAG are transferred to the FIFO) but does determine when the address field is terminated.

The length of the A field may be a single octet, a dual octet, or more octets, as described above. A primary station or an extended address secondary station does not perform an address comparison, and all characters in the A, C, and I fields after the flag are transferred to the FIFO. Although address field comparisons are not performed, the length of the address field is still determined by CMR1[4:3]. For the other secondary address modes, if there is a match, or the received character(s) match either of the other enabling conditions (group or all – parties address), all characters in the A, C, and I fields are transferred to the FIFO. If there is no match, the receiver returns to the FLAG hunt phase.

C phase begins after A phase is terminated. The receiver receives one or two control characters, CMR1[5]. After this phase is terminated, the character length is switched automatically from 8 bits to the number of bits specified in RPR[1:0] and the information field phase is entered.

The frame is terminated when a closing FLAG is detected. The same FLAG can also serve as the opening FLAG of the next frame. The 16 bits received prior to the closing FLAG form the frame check sequence (if an FCS is specified in CMR2[2:0]). All non-FLAG characters of the frame are accumulated in the CRC checker and the result is compared to the expected remainder. Failure to match will cause a CRC error. EOM detect RSR[7], RCL not zero RSR[0], and CRC error RSR[1] are normally FIFO'ed with the last character of the I field. RCL not zero RSR[0] is set if the length of the last character of the I field does not have the length programmed in RPR[1:0]. The residual character length in TRSR[2:0] is also valid at that time. The CRC characters themselves are normally not passed to the RXFIFO. However, if the transfer FCS to FIFO control bit RPR[6] is asserted, the FCS bytes will be transferred to the FIFO. In this case the EOM, CRC error, and RCL not zero status bits will be tagged onto the last byte of the CRC sequence instead of to the last character of the message.

If the closing FLAG is received prior to receipt of the appropriate number of A field, C field as programmed in CMR1[5:3], and FCS field octets, a short frame will be detected and RSR[4] will be set. The I field need not be present in a valid frame. An abort (11111110) comparison is done after an opening FLAG has been received and up to receipt of the closing FLAG. A match causes the abort detect status bit (RSR[6]) to be set. The receiver then enters FLAG search mode. The abort is stripped from the received data stream.

If a zero followed by 15 contiguous ones is detected, the idle detect status bit RSR[3] is set. This comparison is done whenever the receiver is enabled. Therefore, it can occur before or after a received frame.

Zero deletion is performed during BOP receive. A zero after 5 contiguous ones is deleted from the data stream regardless of character boundaries. Deleted zeroes are not subject to CRC accumulation. FLAG, ABORT, and IDLE comparisons are done prior to zero deletion.

If external synchronization is programmed (RPR[4] = 1), the internal FLAG detection and address comparison logic is disabled and receipt of FLAGs is not required. In this arrangement, a pulse on the SYNI input pin will establish synchronization and terminate hunt phase. The receiver will then go immediately into the I-field mode with zero deletion disabled, assembling and transferring characters into the FIFO with the character length specified in RPR[1:0]. The SYNI pin is ignored after the first input on the SYNI pin is received. The receiver must be disabled and then reenabled to resynchronize or to return to normal operating mode.

This mode must be programmed in conjunction with CMR1[2:0] = 110. Refer to the description of RPR[4] for further details.

BOP Loop Mode

Operation of the receiver in BOP loop protocol mode is similar to operation in other BOP modes, except that only certain frame formats are supported. Several character detection functions that interact with the operation of the transmitter or transmitter commands are added:

1. When the 'go on-loop' command is invoked, the receiver looks for the receipt of seven contiguous ones and then asserts the LC_N output.
2. When the 'go off-loop' command is invoked, the receiver looks for the receipt of eight contiguous ones and then negates the LC_N output.
3. The TXD output normally echoes the receive input with a three bit time delay. When the 'go active on poll' command is asserted, the receiver looks for an EOP (a zero followed by seven ones) and then switches the TXD output line to the normal transmitter output. Receipt of an EOP or an ABORT sets RSR[6].
4. Receipt of a turnaround sequence (eight contiguous zeros) terminates the transmitter operation, if any, and returns the TXD output to echoing the RXD input. RSR[3] is set if a turnaround is received.

See transmitter operation for additional details.

2

SUMMARY OF COP FEATURES

COP Dual SYN Mode	
SYN detect	SYN1-SYN2
Linefill	SYN1-SYN2
SYN stripping	SYN1-SYN2 used to establish character sync, i.e. leading SYNs. Subsequent to this (after receiving first non-SYN character), SYN1 and SYN1-SYN2 if stripping is specified by RPR[7].
Excluded from FCS**	SYN1 and SYN1-SYN2 before beginning of message, i.e. leading SYNs and, if SYN stripping is specified by RPR[7] anywhere else in the message for the RX; linefill SYN1-SYN2 for TX regardless of RPR[7]. (If SYN stripping is not specified, then SYNs within a message will be included in FCS by RX).
BISYNC normal mode	
SYN detect	SYN1-SYN2
Linefill	SYN1-SYN2
SYN stripping	SYN1-SYN2 used to establish character sync, i.e. leading SYNs. Subsequent to this (after receiving first non-SYN character), SYN1 and SYN1-SYN2 if stripping is specified by RPR[7].
Excluded from FCS	All SYNs either before or within a message, regardless of RPR[7], plus additional characters as required by the protocol.
BISYNC transparent mode	
SYN detect	*DLE-SYN1
Linefill	DLE-SYN1
SYN/DLE stripping	*DLE-SYN1 and odd DLEs if stripping is specified by RPR[7].
Excluded from FCS	*DLE-SYN1 and odd DLEs, regardless of RPR[7] plus additional characters as required by the protocol.
COP single SYN mode	
SYN detect	SYN1
Linefill	SYN1
SYN stripping	SYN1 used to establish character sync, i.e. leading SYNs. Subsequent to this, SYN1 if stripping is specified by RPR[7].
Excluded from FCS**	SYN1 before beginning of message, i.e. leading SYNs, and if SYN stripping is specified by RPR[7], anywhere else in the message for the RX; linefill SYN1 for TX regardless of RPR[7]. (If SYN stripping is not specified, then SYNs within a message will be included in FCS by RX).

*DLE indicates last DLE of an odd number of consecutive DLEs.

**In non-BISYNC COP modes (single or dual SYN case), if SYN stripping is off, i.e. RPR[7] = 0, then SYNs within a message will be included in FCS by receiver. Therefore, the remote DUSCC transmitter should be careful not to let the TXFIFO underrun since the linefill SYN characters are not accumulated in FCS by the transmitter regardless of RPR[7]. Letting the TXFIFO underrun will result in a CRC error in the receiver.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−55 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 10\%$[4, 5, 6]

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage					
	All except X1/CLK		2.0			V
	X1/CLK		$0.9V_{CC}$		V_{CC}	V
V_{OL}	Output low voltage	$I_{OL} = 2.4mA$			0.4	V
V_{OH}	Output high voltage					
	(except o.c. outputs)	$I_{OH} = -400\mu A$	2.4			V
I_{XIL}	X1/CLK low input current	$V_{IN} = 0$, X2 grounded	−4.0	−2.0	0.0	mA
		$V_{IN} = 0$, X2 floated	−3.0	−1.5	0.0	mA
I_{XIH}	X1/CLK high input current	$V_{IN} = V_{CC}$, X2 grounded	−1.0	0.2	1.0	mA
		$V_{IN} = V_{CC}$, X2 floated	0.0	3.5	10.0	mA
I_{X2L}	X2 low input current	$V_{IN} = 0$, X1/CLK floated	−100	−30	0.0	μA
I_{X2H}	X2 high input current	$V_{IN} = V_{CC}$, X1/CLK floated	0.0	+30	100	μA
I_{IL}	Input leakage current	$V_{IN} = 0$ to V_{CC}	−10		10	μA
I_{LL}	Data bus 3-state leakage current	$V_0 = 0$ to V_{CC}	−10		10	μA
I_{OC}	Open collector output leakage current	$V_0 = 0$ to V_{CC}	−10		10	μA
I_{CC}	Power supply current				150	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other conditions above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature and thermal resistance of 35°C/W junction to ambient for ceramic DIP, 35°C/W for plastic DIP, and 41°C/W for PLCC.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all inputs except X1/CLK swing between 0.4V and 2.4V with a transition time of 20ns maximum. For X1/CLK, this swing is between 0.4V and 4.4V. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical supply voltages, and typical processing parameters.
7. Test condition for outputs: $C_L = 150pF$, except interrupt outputs. Test conditions for interrupt outputs: $C_L = 50pF$, $R_L = 2.7k\Omega$ to V_{CC}.
8. This specification will impose maximum 68000 CPU CLK to 6MHz. Higher CPU CLK can be used if repeating bus reads are not performed.
9. Data is latched by either the rising edge of CSN or the falling edge of DTACKN, whichever occurs first. The hold time applies in either case.
10. Write data including commands are latched as described in note 10. Execution of the valid command (after it is latched) requires two complete periods of the PHI clock or three falling edges of X1 (see figure 14).
11. In single address DMA mode write operation, data is latched either by the falling edge of DTCN of when DTACKN (RDYN) goes low. If DTACKN latches the data, the data hold time with respect to the falling edge of DTACKN is zero.
12. Tests for open drain outputs are intended to guarantee switching of the output transistor. Measurement of this response is referenced from the midpoint of the switching signal to a point 0.5V above V_{OL}. This point represents noise margin that assures true switching has occurred. Beyond this level, the effects of external circuitry and test environment are pronounced and can greatly affect the resultant measurement.

2

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 10\%$[4,5,6,7]

NO.	FIGURE	PARAMETER	TENTATIVE LIMITS			UNIT
			Min	Typ	Max	
1	12	RESETN pulse width	3.0			μs
2	13, 15	A0 – A6 set-up time to CSN low	10			ns
3	13, 15	A0 – A6 hold time from CSN high	0			ns
4	13, 15	RWN set-up time to CSN low	0			ns
5	13, 15	RWN hold time to CSN high	0			ns
6	13, 15	CSN high pulse width[8]	160			ns
7	13, 15, 16	CSN or IACKN high from DTACKN low	30			ns
8	13, 16	Data valid from CSN or IACKN low		125	300	ns
9	13	Data bus floating from CSN high		100	270	ns
10	15	Data hold time from CSN high[9,10,11]	0			ns
11	13, 16	DTACKN low from read data ready	0			ns
12	13, 15	DTACKN low from CSN low		200	360	ns
13	13, 15	DTACKN high from CSN high			235	ns
14	13, 15	DTACKN high impedance from CSN high			360	ns
15	16	DTACKN low from IACKN low		300	650	ns
16	17	Port input set-up time to CSN low	20			ns
17	17	Port input hold time from CSN high	0			ns
18	17	Port output valid from CSN high			560	ns
		IRQN high from:[12]				
19	18	Read RXFIFO (RXRDY interrupt)			300	ns
		Write TXFIFO (TXRDY interrupt)			300	ns
		Write RSR (receiver condition interrupt)			650	ns
		Write TRSR (receiver/transmitter interrupt)			650	ns
		Write ICTSR (port change and timer/counter interrupt)			650	ns
20	19	X1/CLK high or low time	25			ns
		X1/CLK frequency	2.0	14.7456	16	MHz
		CTCLK high or low time	100			ns
		CTCLK frequency	0		4.0	MHz
		RXC high or low time	110			ns
		RXC frequency (16X)	0		4.0	MHz
		(1X)	0		4.0	MHz
		TXC high or low time	110			ns
		TXC frequency (16X)	0		4.0	MHz
		(1X)	0		4.0	MHz
21	20	TXD output from TXC input low			360	ns
22	20	TXD output from TXC output low	0		50	ns
23	21	RXD data set-up time to RXC high	50			ns
24	21	RXD data hold time from RXC high	0			ns
25	22	IACKN low to daisy chain low			350	ns
26	24	Data valid from receive DMA ACKN			320	ns
27	23, 24	DTCN width	100			ns
28	23, 24	RDYN low to DTCN low	80			ns
29	24	Data bus float from DTCN low			300	ns
30	23, 24	DMA ACKN low to RDYN (DTACKN) low	120		340	ns
31	23, 24	RDYN high from DTCN low			420	ns
32	23, 24	RDYN high impedance from DTCN low			530	ns
33	24	Receive DMA REQN high from DMA ACKN low			500	ns
34	24	Receive DMA ACKN width	150			ns
35	23, 24	Receive DMA ACKN low to DONEN low			250	ns
36	23	Data set-up from RDYN low			300	ns
37	23	Data hold from DTCN low[11]			230	ns
38	23	Transmit DMA REQN high from ACKN low			550	ns
39	23	Transmit DMA ACKN width	150			ns
40	23	Transmit DMA ACKN low to DONEN low output			250	ns
41	25	CSN low to transmit DONEN low output			400	ns
42	25	CSN low to transmit DMA REQ negated			620	ns
43	25	CSN low to receive DONEN low			400	ns
44	25	CSN low to receive DMA REQ negated			620	ns

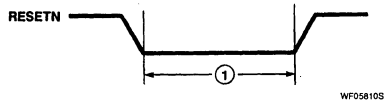


Figure 12. Reset Timing

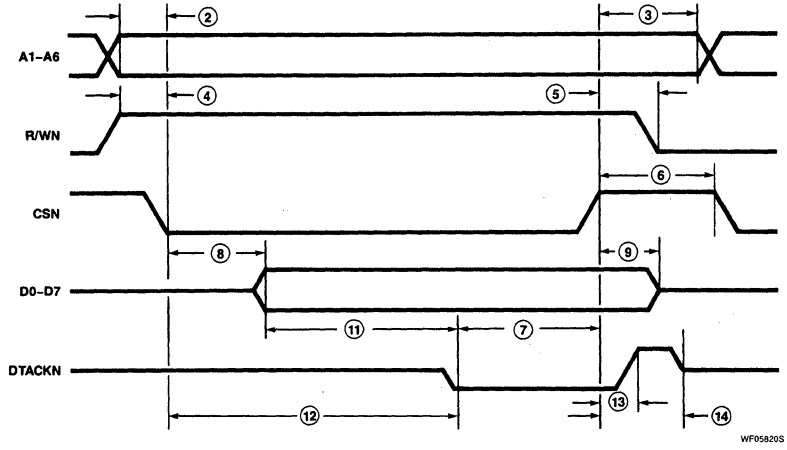


Figure 13. Bus Timing (Read Cycle)

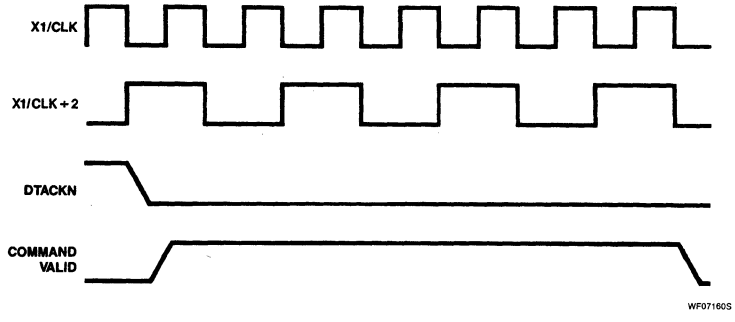


Figure 14. Command Timing

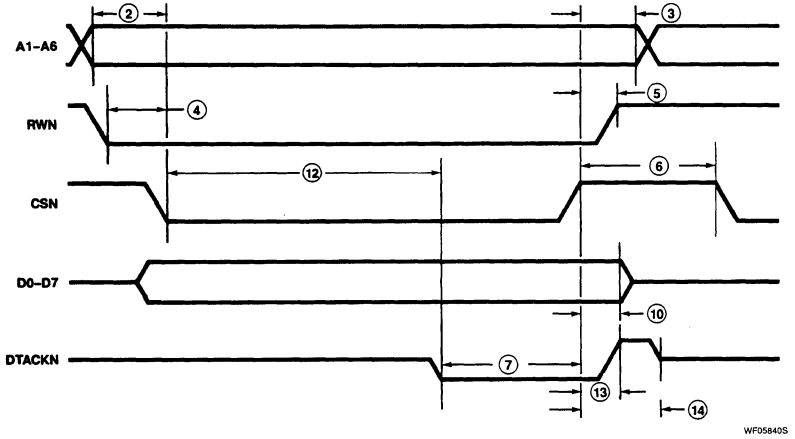


Figure 15. Bus Timing (Write Cycle)

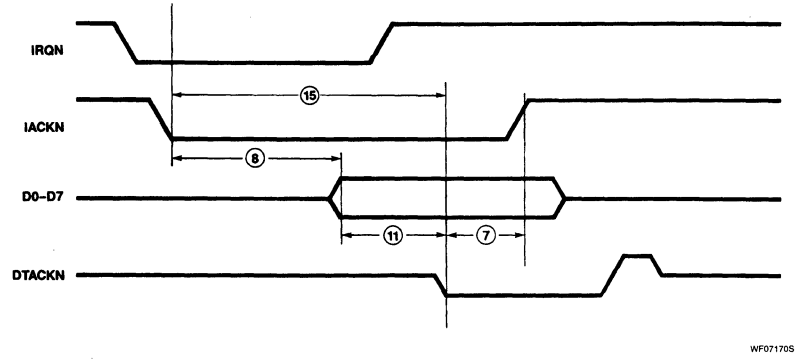


Figure 16. Interrupt Cycle Timing

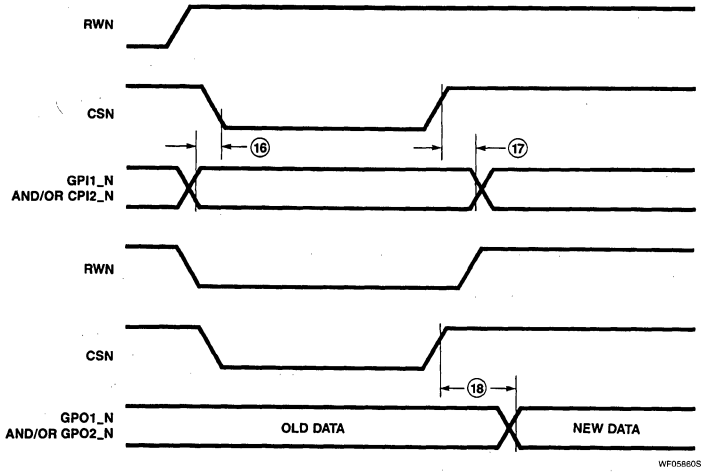


Figure 17. Port Timing

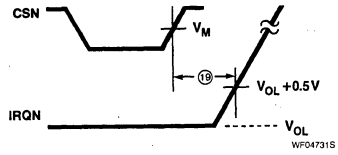


Figure 18. Interrupt Timing

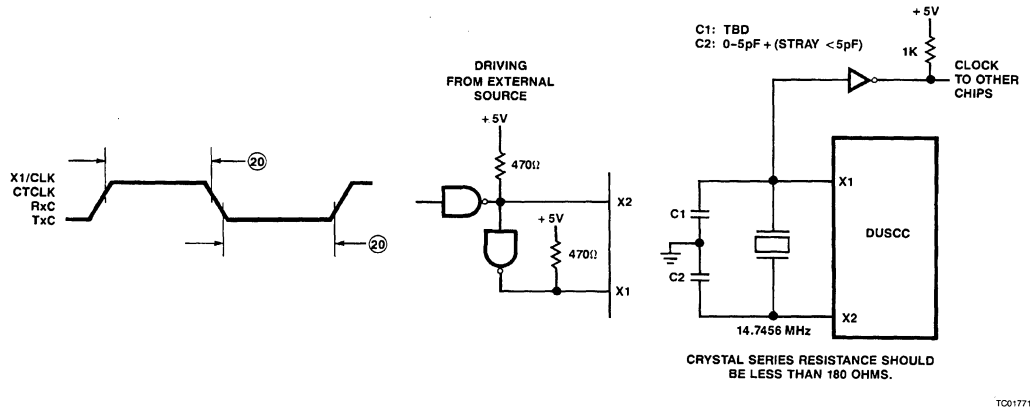


Figure 19. Clock Timing

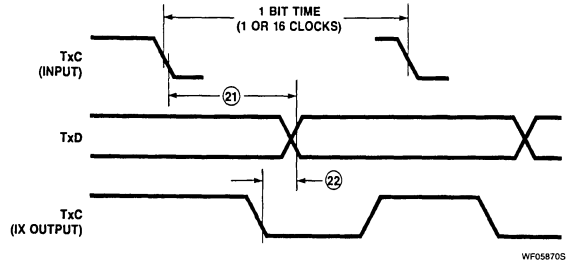


Figure 20. Transmit Timing

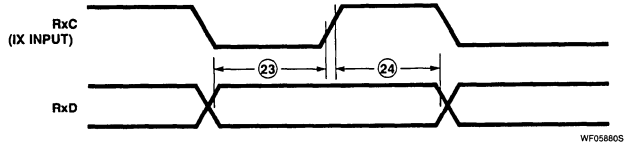


Figure 21. Receive Timing

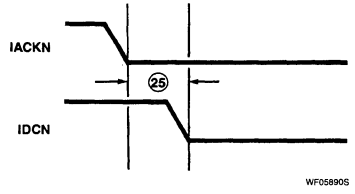


Figure 22. Interrupt Daisy Chain Timing

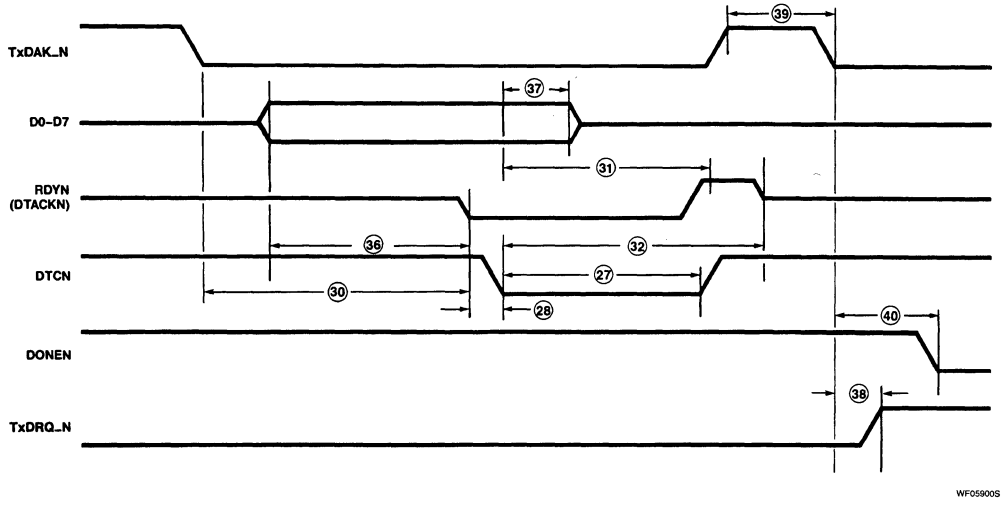


Figure 23. DMA Transmit Write Timing — Single Address DMA Mode

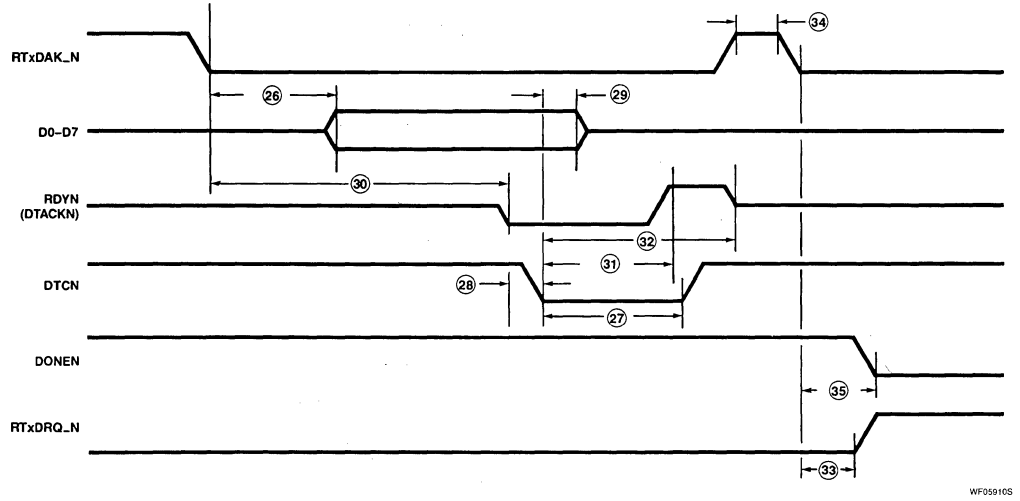


Figure 24. DMA Receiver Read Timing — Single Address DMA Mode

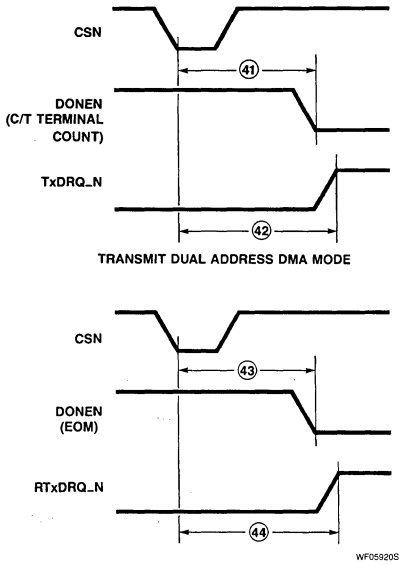


Figure 25. Dual Address DMA Mode Timing

SCN68681
Dual Asynchronous Receiver/ Transmitter (DUART)

Product Specification

DESCRIPTION

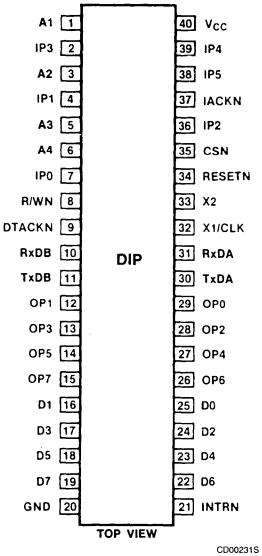
The Signetics SCN68681 Dual Universal Asynchronous Receiver/Transmitter (DUART) is a single chip MOS-LSI communications device that provides two independent full-duplex asynchronous receiver/transmitter channels in a single package. It is compatible with other S68000 family devices, and can also interface easily with other microprocessors. The DUART can be used in polled or interrupt driven systems.

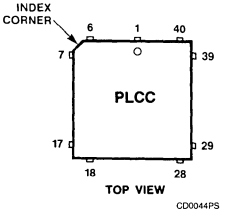
The operating mode and data format of each channel can be programmed independently. Additionally, each receiver and transmitter can select its operating speed as one of eighteen fixed baud rates, a 16x clock derived from a programmable counter/timer, or an external 1x or 16x clock. The baud rate generator and counter/timer can operate directly from a crystal or from external clock inputs. The ability to independently program the operating speed of the receiver and transmitter make the DUART particularly attractive for dual-speed channel applications such as clustered terminal systems.

FEATURES

- **S68000 bus compatible**
- **Dual full-duplex asynchronous receiver/transmitter**
- **Quadruple buffered receiver data registers**
- **Programmable data format**
  - **5 to 8 data bits plus parity**
  - **Odd, even, no parity or force parity**
  - **1, 1.5 or 2 stop bits programmable in 1/16 bit increments**
- **Programmable baud rate for each receiver and transmitter selectable from:**
  - **18 fixed rates: 50 to 38.4K baud**
  - **One user defined rate derived from programmable timer/ counter**
  - **External 1x or 16x clock**
- **Parity, framing, and overrun error detection**
- **False start bit detection**
- **Line break detection and generation**
- **Programmable channel mode**
  - **Normal (full duplex)**
  - **Automatic echo**
  - **Local loopback**
  - **Remote loopback**
- **Multi-function programmable 16-bit counter/timer**
- **Multi-function 6-bit input port**
  - **Can serve as clock or control inputs**
  - **Change of state detection on four inputs**
- **Multi-function 8-bit output port**
  - **Individual bit set/reset capability**
  - **Outputs can be programmed to be status/interrupt signals**
- **Versatile interrupt system**
  - **Single interrupt output with eight maskable interrupting conditions**
  - **Interrupt vector output on interrupt acknowledge**
  - **Output port can be configured to provide a total of up to six separate wire-OR'able interrupt outputs**
- **Maximum data transfer: 1X – 1MB/sec, 16X – 125KB/sec**
- **Automatic wake-up mode for multidrop applications**
- **Start-end break interrupt/status**
- **Detects break which originates in the middle of a character**
- **On-chip crystal oscillator**
- **TTL compatible**
- **Single +5V power supply**

PIN CONFIGURATION





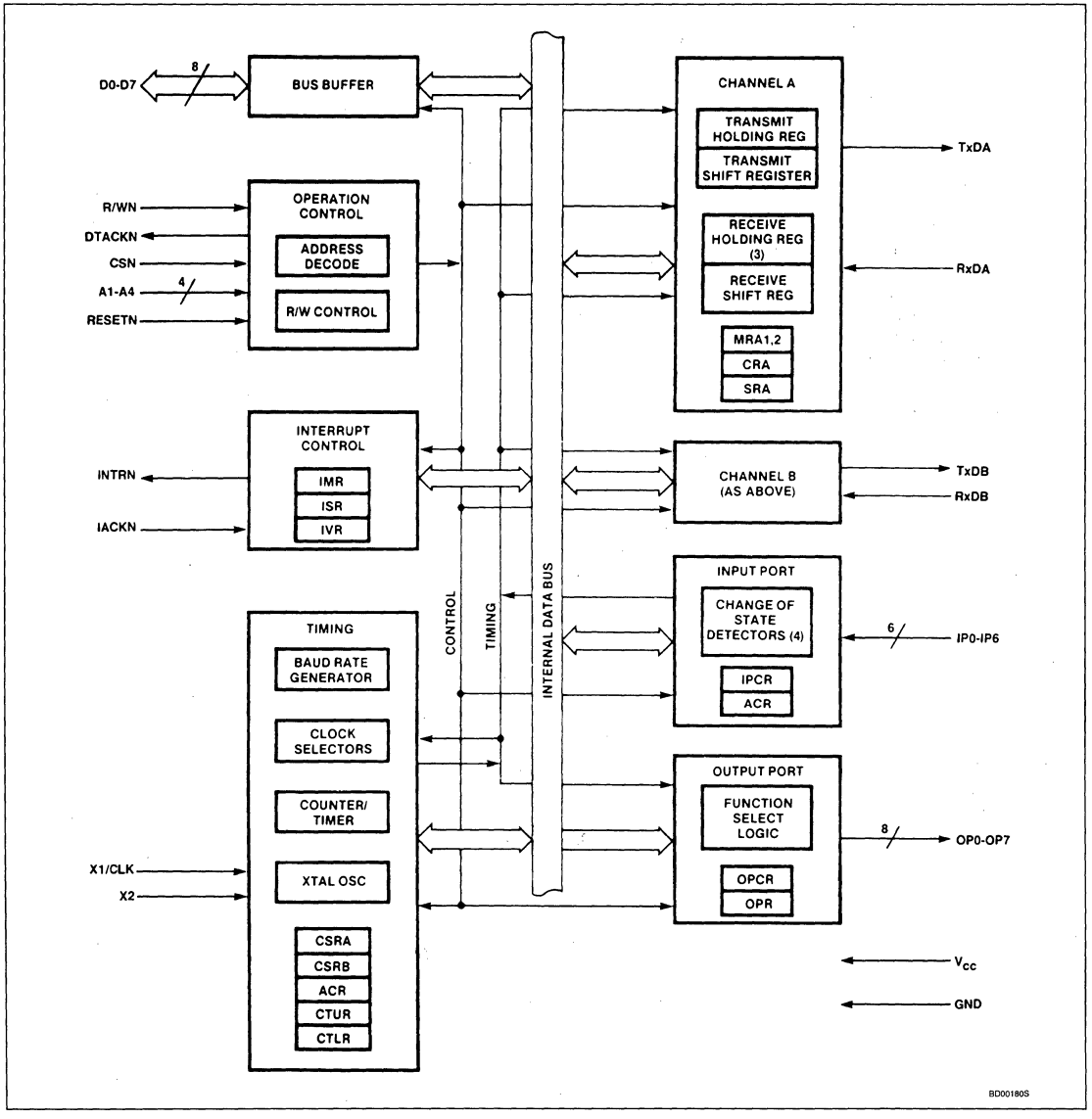
Pin	Function	Pin	Function
1	NC	23	NC
2	A1	24	INTRN
3	IP3	25	D6
4	A2	26	D4
5	IP1	27	D2
6	A3	28	D0
7	A4	29	OP6
8	IP0	30	OP4
9	R/WN	31	OP2
10	DTACKN	32	OP0
11	RXDB	33	TXDA
12	NC	34	NC
13	TXDB	35	RXDA
14	OP1	36	X1/CLK
15	OP3	37	X2
16	OP5	38	RESET
17	OP7	39	CSN
18	D1	40	IP2
19	D3	41	IACKN
20	D5	42	IP5
21	D7	43	IP4
22	GND	44	V_{CC}

Dual Asynchronous Receiver/Transmitter (DUART) SCN68681

ORDERING CODE

PACKAGES	V_{CC} = 5V ±5%, T_A = 0°C to 70°C
Ceramic DIP	SCN68681C1I40
Plastic DIP	SCN68681C1N40
Plastic LCC	SCN68681CIA44

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
D0 – D7	25, 16, 24 17, 23, 18 22, 19	28, 18, 27 19, 26, 20 25, 21	I/O	**Data Bus:** Bidirectional 3-state data bus used to transfer commands, data and status between the DUART and the CPU. D0 is the least significant bit.
CSN	35	39	I	**Chip Select:** Active low input signal. When low, data transfers between the CPU and the DUART are enabled on D0 – D7 as controlled by the R/WN and A1 – A4 inputs. When high, places the D0 – D7 lines in the 3-state condition.
R/WN	8	9	I	**Read/Write:** A high input indicates a read cycle and a low input indicates a write cycle, when a cycle is initiated by assertion of the CSN input.
A1 – A4	1, 2 5, 6	2, 4 6, 7	I	**Address Inputs:** Select the DUART internal registers and ports for read/write operations.
RESETN	34	38	I	**Reset:** A low clears internal registers (SRA, SRB, IMR, ISR, OPR, OPCR), initializes the IVR to hex 0F, puts OP0 – OP7 in the high state, stops the counter/timer, and puts channel A and B in the inactive state, with the TxDA and TxDB outputs in the mark (high) state.
DTACKN	9	10	O	**Data Transfer Acknowledge:** Three-state active low output asserted in write, read, or interrupt cycles to indicate proper transfer of data between the CPU and the DUART.
INTRN	21	24	O	**Interrupt Request:** Active low, open drain output which signals the CPU that one or more of the eight maskable interrupting conditions are true.
IACKN	37	41	I	**Interrupt Acknowledge:** Active low input indicating an interrupt acknowledge cycle. In response, the DUART will place the interrupt vector on the data bus and will assert DTACKN if it has an interrupt pending.
X1/CLK	32	36	I	**Crystal 1:** Crystal or external clock input. A crystal or clock of the specified limits must be supplied at all times. If a crystal is used, a capacitor must be connected from this pin to ground (see figure 7).
X2	33	37	I	**Crystal 2:** Connection for other side of the crystal. If a crystal is used, a capacitor must be connected from this pin to ground (see figure 7).
RxDA	31	35	I	**Channel A Receiver Serial Data Input:** The least significant bit is received first. 'Mark' is high, 'space' is low.
RxDB	10	11	I	**Channel B Receiver Serial Data Input:** The least significant bit is received first. 'Mark' is high, 'space' is low.
TxDA	30	33	O	**Channel A Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the 'mark' condition when the transmitter is disabled, idle, or when operating in local loopback mode. 'Mark' is high, 'space' is low.
TxDB	11	13	O	**Channel B Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the 'mark' condition when the transmitter is disabled, idle, or when operating in local loopback mode. 'Mark' is high, 'space' is low.
OP0	29	32	O	**Output 0:** General purpose output, or channel A request to send (RTSAN, active low). Can be deactivated automatically on receive or transmit.
OP1	12	14	O	**Output 1:** General purpose output, or channel B request to send (RTSBN, active low). Can be deactivated automatically on receive or transmit.
OP2	28	40	O	**Output 2:** General purpose output, or channel A transmitter 1X or 16X clock output, or channel A receiver 1X clock output.
OP3	13	15	O	**Output 3:** General purpose output, or open drain, active low counter/timer output, or channel B transmitter 1X clock output, or channel B receiver 1X clock output.
OP4	27	30	O	**Output 4:** General purpose output, or channel A open drain, active low, RxRDYA/FFULLA output.
OP5	14	16	O	**Output 5:** General purpose output or channel B open drain, active low, RxRDYB/FFULLB output.
OP6	26	29	O	**Output 6:** General purpose output, or channel A open drain, active low, TxRDYA output.
OP7	15	17	O	**Output 7:** General purpose output, or channel B open drains, active low, TxRDY B output.
IP0	7	8	I	**Input 0:** General purpose input, or channel A clear to send active low input (CTSAN).
IP1	4	5	I	**Input 1:** General purpose input, or channel B clear to send active low input (CTSBN).

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
IP2	36	40	I	**Input 2:** General purpose input, or channel B receiver external clock input (RxCB), or counter/timer external clock input. When the external clock is used by the receiver, the received data is sampled on the rising edge of the clock.
IP3	2	3	I	**Input 3:** General purpose input, or channel A transmitter external clock input (TxCA). When the external clock is used by the transmitter, the transmitted data is clocked on the falling edge of the clock.
IP4	39	43	I	**Input 4:** General purpose input, or channel A receiver external clock input (RxCA). When the external clock is used by the receiver, the received data is sampled on the rising edge of the clock.
IP5	38	42	I	**Input 5:** General purpose input, or channel B transmitter external clock input (TxCB). When the external clock is used by the transmitter, the transmitted data is clocked on the falling edge of the clock.
V_{CC}	40	44	I	**Power Supply:** +5V supply input.
GND	20	22	I	**Ground**

Each receiver is quadruply buffered to minimize the potential of receiver overrun or to reduce interrupt overhead in interrupt driven systems. In addition, a flow control capability is provided to disable a remote DUART transmitter when the buffer of the receiving device is full.

Also provided on the SCN68681 are a multipurpose 6-bit input port and a multipurpose 8-bit output port. These can be used as general purpose I/O ports or can be assigned specific functions (such as clock inputs or status/interrupt outputs) under program control.

BLOCK DIAGRAM

The SCN68681 DUART consists of the following eight major sections: data bus buffer, operation control, interrupt control, timing, communications channels A and B, input port and output port. Refer to the block diagram.

Data Bus Buffer

The data bus buffer provides the interface between the external and internal data busses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the DUART.

Operation Control

The operation control logic receives operation commands from the CPU and generates appropriate signals to internal sections to control device operation. It contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer. The DTACKN output is asserted during write and read cycles to indicate to the CPU that data has been latched on a write cycle, or that valid data is present on the bus on a read cycle.

Interrupt Control

A single active low interrupt output (INTRN) is provided which is activated upon the occurrence of any of eight internal events. Associated with the interrupt system are the interrupt mask register (IMR), the interrupt status register (SR), the auxiliary control register (ACR), and the interrupt vector register (IVR). The IMR may be programmed to select only certain conditions to cause INTRN to be asserted. The ISR can be read by the CPU to determine all currently active interrupting conditions. When IACKN is asserted, and the DUART has an interrupt pending, the DUART responds by placing the contents of the IVR register on the data bus and asserting DTACKN.

Outputs OP3-OP7 can be programmed to provide discrete interrupt outputs for the transmitters, receivers, and counter/timer.

Timing Circuits

The timing block consists of a crystal oscillator, a baud rate generator, a programmable 16-bit counter/timer, and four clock selectors. The crystal oscillator operates directly from a 3.6864NHz crystal connected across the X1/CLK and X2 inputs. If an external clock of the appropriate frequency is available, it may be connected to X1/CLK. The clock serves as the basic timing reference for the baud rate generator (BRG), the counter/timer and other internal circuits. A clock signal within the limits specified in the specifications section of this data sheet must always be supplied to the DUART. If an external signal is used instead of a crystal, both X1 and X2 should be driven using a configuration similar to the one in figure 7.

The baud rate generator operates from the oscillator or external clock input and is capable of generating 18 commonly used data communications baud rates ranging from 50 to 38.4K baud. The clock outputs from the BRG are at 16X the actual baud rate. The counter/timer can be used as a timer to produce a 16X clock for any other baud rate by counting down the crystal clock or an external clock. The four clock selectors allow the independent selection, for each receiver and transmitter, of any of these baud rates or an external timing signal.

The counter/timer (C/T) can be programmed to use one of several timing sources as its input. The output of the C/T is available to the clock selectors and can also be programmed to be output at OP3. In the counter mode, the contents of the C/T can be read by the CPU and it can be stopped and started under program control. In the timer mode, the C/T acts as a programmable divider.

Communications Channels A and B

Each communications channel of the SCN68681 comprises a full duplex asynchronous receiver/transmitter (DUART). The operating frequency for each receiver and transmitter can be selected independently from the baud rate generator, the counter timer, or from an external input.

The transmitter accepts parallel data from the CPU, converts it to a serial bit stream, inserts the appropriate start, stop, and optional parity bits and outputs a composite serial stream of data on the TxD output pin. The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for start bit, stop bit, parity bit (if any), or break condition and sends an assembled character to the CPU.

The input port pulse detection circuitry uses a 38.4KHz sampling clock derived from one of

the baud rate generator taps. This results in a sampling period of slightly more than 25μsec (assuming that the clock input is 3.6864MHz). The detection circuitry, in order to guarantee a true change in level has occurred, requires that two successive samples at the new logic level be observed. As a consequence, the minimum duration of the signal change is 25μsec if the transition occurs coincident with the first sample pulse. The 50μsec time refers to the situation in which the change of state is just missed and the first change of state is not detected until 25μsec later.

Input Port

The inputs to this unlatched 6-bit port can be read by the CPU by performing a read operation at address D_{16}. A high input results in a logic 1 while a low input results in a logic 0. D7 will always be read as a logic 1 and D6 will reflect the level of IACKN. The pins of this port can also serve as auxiliary inputs to certain portions of the DUART logic.

Four change-of-state detectors are provided which are associated with inputs IP3, IP2, IP1, and IP0. A high-to-low or low-to-high transition of these inputs lasting longer than 25–50μs will set the corresponding bit in the input port change register. The bits are cleared when the register is read by the CPU. Any change of state can also be programmed to generate an interrupt to the CPU.

Output Port

The 8-bit multi-purpose output port can be used as a general purpose output port, in which case the outputs are the complements of the output port register (OPR). OPR[n] = 1 results in OP[n] = low and vice-versa. Bits of the OPR can be individually set and reset. A bit is set by performing a write operation at address E_{16} with the accompanying data specifying the bits to be set (1 = set, 0 = no change). Likewise, a bit is reset by a write at address F_{16} with the accompanying data specifying the bits to be reset (1 = reset, 0 = no change).

Outputs can be also individually assigned specific functions by appropriate programming of the channel A mode registers (MR1A, MR2A), the channel B mode registers (MR1B, MR2B), and the output port configuration register (OPCR).

OPERATION

Transmitter

The SCN68681 is conditioned to transmit data when the transmitter is enabled through the command register. The SCN68681 indicates to the CPU that it is ready to accept a character by setting the TxRDY bit in the status register. This condition can be programmed to generate an interrupt request at OP6 or OP7 and INTRN. When a character is loaded into the transmit holding register (THR), the above conditions are negated. Data is transferred from the holding register to the transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again which means one full character time of buffering is provided. Characters cannot be loaded into the THR while the transmitter is disabled.

The transmitter converts the parallel data from the CPU to a serial bit stream on the TxD output pin. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The least significant bit is sent first. Following the transmission of the stop bits, if a new character is not available in the THR, the TxD output remains high and the TxEMT bit in the status register (SR) will be set to 1. Transmission resumes and the TxEMT bit is cleared when the CPU loads a new character into the THR. If the transmitter is disabled, it continues operating until the character currently being transmitted is completely sent out. The transmitter can be forced to send a continuous low condition by issuing a send break command.

The transmitter can be reset through a software command. If it is reset, operation ceases immediately and the transmitter must be enabled through the command register before resuming operation. If CTS operation is enabled, the CTSN input must be low in order for the character to be transmitted, if it goes high in the middle of a transmission, the character in the shift register is transmitted and TxDA then remains in the marking state until CTSN goes low. The transmitter can also control the deactivation of the RTSN output. If programmed, the RTSN output will be reset one bit time after the character in the transmit shift register and transmit holding register (if any) are completely transmitted, if the transmitter has been disabled.

Receiver

The SCN68681 is conditioned to receive data when enabled through the command register. The receiver looks for a high to low (mark to space) transition of the start bit on the RxD input pin. If a transition is detected, the state of the RxD pin is sampled each 16X clock for 7-1/2 clocks (16X clock model) or at the next rising edge of the bit time clock (1X clock mode). If RxD is sampled high, the start bit is invalid and the search for a valid start bit begins again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input at one bit time intervals at the theoretical center of the bit, until the proper number of data bits and the parity bit (if any) have been assembled, and one stop bit has been detected. The least significant bit is received first. The data is then transferred to the receive holding register (RHR) and the RxRDY bit in the SR is set to a 1. This condition can be programmed to generate an interrupt at OP4 or OP5 and INTRN. If the character length is less than eight bits, the most significant unused bits in the RHR are set to zero.

After the stop bit is detected, the receiver will immediately look for the next start bit. However, if a non-zero character was received without a stop bit (framing error) and RxD remains low for one half of the bit period after the stop bit was sampled, then the receiver operates as if a new start bit transition had been detected at that point (one-half bit time after the stop bit was sampled).

The parity error, framing error, overrun error and received break state (if any) are strobed into the SR at the received character boundary, before the RxRDY status bit is set. If a break condition is detected (RxD is low for the entire character including the stop bit), a character consisting of all zeros will be loaded into the RHR and the received break bit in the SR is set to 1. The RxD input must return to a high condition for at least one-half bit time before a search for the next start bit begins.

The RHR consists of a first-in-first-out (FIFO) stack with a capacity of three characters. Data is loaded from the receive shift register into the topmost empty position of the FIFO. The RxRDY bit in the status register is set whenever one or more characters are available to be read, and a FFULL status bit is set if all three stack positions are filled with data. Either of these bits can be selected to cause an interrupt. A read of the RHR outputs the data at the top of the FIFO. After the read cycle, the data FIFO and its associated status bits (see below) are 'popped' thus emptying a FIFO position for new data.

In addition to the data word, three status bits (parity error, framing error, and received break) are also appended to each data character in the FIFO (overrun is not). Status can be provided in two ways, as programmed by the error mode control bit in the mode register. In the 'character' mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these three bits is the logical OR of the status for all characters coming to the top of the FIFO since the last 'reset error' command was issued. In either mode reading the SR does not affect the FIFO. The FIFO is 'popped' only when the RHR is read. Therefore the status register should be read prior to reading the FIFO.

If the FIFO is full when a new character is received, that character is held in the receive shift register until a FIFO position is available.

Table 1. 68681 REGISTER ADDRESSING

A4	A3	A2	A1	READ (R/WN = 1)	WRITE (R/WN = 0)
0	0	0	0	Mode Register A (MR1A, MR2A)	Mode Register A (MR1A, MR2A)
0	0	0	1	Status Register A (SRA)	Clock Select Reg. A (CSRA)
0	0	1	0	*Reserved*	Command Register A (CRA)
0	0	1	1	RX Holding Register A (RHRA)	TX Holding Register A (THRA)
0	1	0	0	Input Port Change Reg. (IPCR)	Aux. Control Register (ACR)
0	1	0	1	Interrupt Status Reg. (ISR)	Interrupt Mask Reg. (IMR)
0	1	1	0	Counter/Timer Upper (CTU)	C/T Upper Register (CTUR)
0	1	1	1	Counter/Timer Lower (CTL)	C/T Lower Register (CTLR)
1	0	0	0	Mode Register B (MR1B, MR2B)	Mode Register B (MR1B, MR2B)
1	0	0	1	Status Register B (SRB)	Clock Select Reg. B (CSRB)
1	0	1	0	*Reserved*	Command Register B (CRB)
1	0	1	1	RX Holding Register B (RHRB)	TX Holding Register B (THRB)
1	1	0	0	Interrupt Vector Reg. (IVR)	Interrupt Vector Reg. (IVR)
1	1	0	1	Input Port	Output Port Conf. Reg. (OPCR)
1	1	1	0	Start Counter Command	Set Output Port Bits Command
1	1	1	1	Stop Counter Command	Reset Output Port Bits Command

If an additional character is received while this state exits, the contents of the FIFO are not affected: the character previously in the shift register is lost and the overrun error status bit (SP[4]) will be set upon receipt of the start bit of the new (overruning) character.

The receiver can control the deactivation of RTS. If programmed to operate in this mode, the RTSN output will be negated when a valid start bit was received and the FIFO is full. When a FIFO position becomes available, the RTSN output will be re-asserted automatically. This feature can be used to prevent an overrun, in the receiver, by connecting the RTSN output to the CTSN input of the transmitting device.

If the receiver is disabled, the FIFO characters can be read. However, no additional characters can be received until the receiver is enabled again. If the receiver is reset, the FIFO and all of the receiver status, and the corresponding output ports and interrupt are reset. No additional characters can be received until the receiver is enabled again.

Multidrop Mode

The DUART is equipped with a wake up mode used for multidrop applications. This mode is selected by programming bits MR1A[4:3] or MR1B[4:3] to '11' for channels A and B respectively. In this mode of operation, a 'master' station transmits an address character followed by data characters for the addressed 'slave' station. The slave stations, with receivers that are normally disabled, examine the received data stream and 'wake-up' the CPU (by setting RxRDY) only upon receipt of an address character. The CPU compares the received address to its station address and enables the receiver if it wishes to receive the subsequent data characters. Upon receipt of another address character, the CPU may disable the receiver to initiate the process again.

A transmitted character consists of a start bit, the programmed number of data bits, an address/data (A/D) bit, and the programmed number of stop bits. The polarity of the transmitted A/D bit is selected by the CPU by programming bit MR1A[2]/MR1B[2]. MR1A[2]/MR1B[2] = 0 transmits a zero in the A/D bit position, which identifies the corresponding data bits as data, while MR1A[2]/MR1B[2] = 1 transmits a one in the A/D bit position, which identifies the corresponding data bits as an address. The CPU should program in the mode register prior to loading the corresponding data bits into the THR.

In this mode, the receiver continuously looks at the received data stream, whether it is enabled or disabled. If disabled, it sets the RxRDY status bit and loads the character into the RHR FIFO if the received A/D bit is a one (address tag), but discards the received character if the received A/D bit is a zero (data tag). If enabled, all received characters are transferred to the CPU via the RHR. In either case, the data bits are loaded into the data FIFO while the A/D bit is loaded into the status FIFO position normally used for parity error (SRA[5] or SRB[5]). Framing error, overrun error, and break detect operate normally whether or not the receiver is enabled.

PROGRAMMING

The operation of the DUART is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The addressing of the registers is described in table 1.

The contents of certain control registers are initialized to zero on RESET. Care should be exercised if the contents of a register are changed during operation, since certain changes may cause operational problems. For example, changing the number of bits per character while the transmitter is active may cause the transmission of an incorrect character. In general, the contents of the MR, the CSR, and the OPCR should only be changed while the receiver(s) and transmitter(s) are not enabled, and certain changes to the ACR should only be made while the C/T is stopped.

Mode registers 1 and 2 of each channel are accessed via independent auxiliary pointers. The pointer is set to MR1x by RESET or by issuing a 'reset pointer' command via the corresponding command register. Any read or write of the mode register while the pointer is at MR1x switches the pointer to MR2x. the pointer then remains at MR2x, so that subsequent accesses are always to MR2x unless the pointer is reset to MR1x as described above.

Mode, command, clock select, and status registers are duplicated for each channel to provide total independent operation and control. Refer to table 2 for register bit descriptions. The reserved registers at addresses H'02' and H'0A' should never be read during normal operation since they are reserved for internal diagnostics.

2

Table 2. REGISTER BIT FORMATS

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RX RTS CONTROL	RX INT SELECT	ERROR MODE	PARITY MODE		PARITY TYPE	BITS PER CHAR.	
MR1A MR1B	0=no 1=yes	0=RXRDY 1=FFULL	0=char 1=block	00=with parity 01=force parity 10=no parity 11=multi-drop mode		0=even 1=odd	00=5 01=6 10=7 11=8	

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	CHANNEL MODE		Tx RTS CONTROL	CTS ENABLE Tx	STOP BIT LENGTH*			
MR2A MR2B	00=Normal 01=Auto echo 10=Local loop 11=Remote loop		0=no 1=yes	0=no 1=yes	0=0.563 1=0.625 2=0.688 3=0.750	4=0.813 5=0.875 6=0.938 7=1.000	8=1.563 9=1.625 A=1.688 B=1.750	C=1.813 D=1.875 E=1.938 F=2.000

*Add 0.5 to values shown for 0–7 if channel is programmed for 5 bits/char.

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RECEIVER CLOCK SELECT				TRANSMITTER CLOCK SELECT			
CSRA CSRB	See Text				See Text			

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
		MISCELLANEOUS COMMANDS			DISABLE Tx	ENABLE Tx	DISABLE Rx	ENABLE Rx
CRA CRB	not used– must be 0	See Text			0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RECEIVED BREAK	FRAMING ERROR	PARITY ERROR	OVERRUN ERROR	TxEMT	TxRDY	FFULL	RxRDY
SRA SRB	0=no 1=yes *	0=no 1=yes *	0=no 1=yes *	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes

*These status bits are appended to the corresponding data character in the receive FIFO. A read of the status register provides these bits (7:5) from the top of the FIFO together with bits 4:0. These bits are cleared by a 'reset error status' command. In character mode they are discarded when the corresponding data character is read from the FIFO.

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	OP7	OP6	OP5	OP4	OP3		OP2	
OPCR	0 = OPR[7] 1=TxRDYB	0 = OPR[6] 1=TxRDYA	0 = OPR[5] 1=RxRDY/ FFULLB	0 = OPR[4] 1=RxRDY/ FFULLA	00 = OPR[3] 01 = C/T OUTPUT 10 = TxCB(1X) 11 = RxCB(1X)		00 = OPR[2] 01 = TxCA(16X) 10 = TxCA(1X) 11 = RxCA(1X)	

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	BRG SET SELECT	COUNTER/TIMER MODE AND SOURCE			DELTA IP3 INT	DELTA IP2 INT	DELTA IP1 INT	DELTA IP0 INT
ACR	0 = set1 1 = set2	See table 4			0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on

Table 2. REGISTER BIT FORMATS (Continued)

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
IPCR	DELTA IP3	DELTA IP2	DELTA IP1	DELTA IP0	IP3	IP2	IP1	IP0
	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = low 1 = high	0 = low 1 = high	0 = low 1 = high	0 = low 1 = high

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ISR	INPUT PORT CHANGE	DELTA BREAK B	RxRDY/ FFULLB	TxRDYB	COUNTER READY	DELTA BREAK A	RxRDY/ FFULLA	TxRDYA
	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
IMR	IN. PORT CHANGE INT	DELTA BREAK B INT	RxRDY/ FFULLB INT	TxRDYB INT	COUNTER READY INT	DELTA BREAK A INT	RxRDY/ FFULLA INT	TxRDYA INT
	0=off 1=on	0=off 1=on	0=off 1=on	0=off 1=on	0=off 1=on	0=off 1=on	0=off 1=on	0=off 1=on

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CTUR	C/T[15]	C/T[14]	C/T[13]	C/T[12]	C/T[11]	C/T[10]	C/T[9]	C/T[8]

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CTLR	C/T[7]	C/T[6]	C/T[5]	C/T[4]	C/T[3]	C/T[2]	C/T[1]	C/T[0]

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
IVR	IVR[7]	IVR[6]	IVR[5]	IVR[4]	IVR[3]	IVR[2]	IVR[1]	IVR[0]

MR1A – Channel A Mode Register 1

MR1A is accessed when the channel A MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRA. After reading or writing MR1A, the pointer will point to MR2A.

MR1A[7] – Channel A Receiver Request-to-Send Control

This bit controls the deactivation of the RTSAN output (OP0) by the receiver. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR1A[7] = 1 causes RTSAN to be negated upon receipt of a valid start bit if the channel A FIFO is full. However, OPR[0] is not reset and RTSAN will be asserted again when an empty FIFO position is available. This feature can be used for flow control to prevent overrun in the receiver by using the RTSAN output signal to control the CTSN input of the transmitting device.

MR1A[6] – Channel A Receiver Interrupt Select

This bit selects either the channel A receiver ready status (RXRDY) or the channel A FIFO full status (FFULL) to be used for CPU interrupts. It also causes the selected bit to be output on OP4 if it is programmed as an interrupt output via the OPCR.

MR1A[5] – Channel A Error Mode Select

This bit selects the operating mode of the three FIFOed status bits (FE, PE, received break) for channel A. In the 'character' mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these bits is the accumulation (logical OR) of the status for all characters coming to the top of the FIFO since the last 'reset error' command for channel A was issued.

MR1A[4:3] – Channel A Parity Mode Select

If 'with parity' or 'force parity' is selected, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR1A[4:3] = 11 selects channel A to operate in the special multidrop mode described in the Operation section.

MR1A[2] – Channel A Parity Type Select
This bit selects the parity type (odd or even) if the 'with parity' mode is programmed by MR1A[4:3], and the polarity of the forced parity bit if the 'force parity' mode is programmed. It has no effect if the 'no parity' mode is programmed. In the special multidrop mode it selects the polarity of the A/D bit.

MR1A[1:0] – Channel A Bits per Character Select
This field selects the number of data bits per character to be transmitted and received. The character length does not include the start, parity, and stop bits.

MR2A – Channel A Mode Register 2

MR2A is accessed when the channel A MR pointer points to MR2, which occurs after any access to MR1A. Accesses to MR2A do not change the pointer.

MR2A[7:6] – Channel A Mode Select
Each channel of the DUART can operate in one of four modes. MR2A[7:6] = 00 is the normal mode, with the transmitter and receiver operating independently. MR2A[7:6] = 01 places the channel in the automatic echo mode, which automatically retransmits the received data. The following conditions are true while in automatic echo mode:

1. Received data is reclocked and retransmitted on the TxDA output.
2. The receive clock is used for the transmitter.
3. The receiver must be enabled, but the transmitter need not be enabled.
4. The channel A TxRDY and TxEMT status bits are inactive.
5. The received parity is checked, but is not regenerated for transmission, i.e., transmitted parity bit is as received.
6. Character framing is checked, but the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.
8. CPU to receiver communication continues normally, but the CPU to transmitter link is disabled.

Two diagnostic modes can also be configured. MR2A[7:6] = 10 selects local loopback mode. In this mode:

1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver.
3. The TxDA output is held high.
4. The RxDA input is ignored.
5. The transmitter must be enabled, but the receiver need not be enabled.
6. CPU to transmitter and receiver communications continue normally.

The second diagnostic mode is the remote loopback mode, selected by MR2A[7:6] = 11. In this mode:

1. Received data is reclocked and retransmitted on the TxDA output.
2. The receive clock is used for the transmitter.
3. Received data is not sent to the local CPU, and the error status conditions are inactive.
4. The received parity is not checked and is not regenerated for transmission, i.e., transmitted parity bit is as received.
5. The receiver must be enabled.
6. Character framing is not checked, and the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.

The user must exercise care when switching into and out of the various modes. The selected mode will be activated immediately upon mode selection, even if this occurs in the middle of a received or transmitted character. Likewise, if a mode is deselected, the device will switch out of the mode immediately. An exception to this is switching out of autoecho or remote loopback modes: if the de-selection occurs just after the receiver has sampled the stop bit (indicated in autoecho by assertion of RxRDY), and the transmitter is enabled, the transmitter will remain in autoecho mode until the entire stop bit has been retransmitted.

MR2A[5] – Channel A Transmitter Request-to-Send Control
This bit controls the deactivation of the RTSAN output (OP0) by the transmitter. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR2A[5] = 1 causes OPR[0] to be reset automatically one bit time after the characters in the channel A transmit shift register and in the THR, if any, are completely transmitted, including the programmed number of stop bits, if the transmitter is not enabled. This feature can be used to automatically terminate the transmission of a message as follows:

1. Program auto-reset mode: MR2A[5] = 1.
2. Enable transmitter.
3. Assert RTSAN: OPR[0] = 1.
4. Send message.
5. Verify the message is sent by waiting until the transmit ready status (TxRDY) is asserted. Disable transmitter after the last character is loaded into the channel A THR.
6. The last character will be transmitted and OPR[0] will be reset one bit time after the last stop bit, causing RTSAN to be negated.

MR2A[4] – Channel A Clear-to-Send Control
If this bit is 0, CTSAN has no effect on the transmitter. If this bit is a 1, the transmitter checks the state of CTSAN (IP0) each time it is ready to send a character. If IP0 is asserted (low), the character is transmitted. If it is negated (high), the TxDA output remains in the marking state and the transmission is delayed until CTSAN goes low. Changes in CTSAN while a character is being transmitted do not affect the transmission of that character.

MR2A[3:0] – Channel A Stop Bit Length Select
This field programs the length of the stop bit appended to the transmitted character. Stop bit lengths of 9/16 to 1 and 1-9/16 to 2 bits, in increments of 1/16 bit, can be programmed for character lengths of 6, 7, and 8 bits. For a character length of 5 bits, 1-1/16 to 2 stop bits can be programmed in increments of 1/16 bit. The receiver only checks for a 'mark' condition at the center of the first stop bit position (one bit time after the last data bit, or after the parity bit if parity is enabled) in all cases.

If an external 1X clock is used for the transmitter, MR2A[3] = 0 selects one stop bit and MR2A[3] = 1 selects two stop bits to be transmitted.

MR1B – Channel B Mode Register 1

MR1B is accessed when the channel B MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRB. After reading or writing MR1B, the pointer will point to MR2B.

The bit definitions for this register are identical to the bit definitions for MR1A, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

MR2B – Channel B Mode Register 2

MR2B is accessed when the channel B MR pointer points to MR2, which occurs after any access to MR1B. Accesses to MR2B do not change the pointer.

The bit definitions for this register are identical to the bit definitions for MR2A, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

CSRA – Channel A Clock Select Register

CSRA[7:4] – Channel A Receiver Clock Select
This field selects the baud rate clock for the channel A receiver as follows:

	Baud Rate Clock = 3.6864MHz	
CSRA[7:4]	ACR[7] = 0	ACR[7] = 1
0 0 0 0	50	75
0 0 0 1	110	110
0 0 1 0	134.5	134.5
0 0 1 1	200	150
0 1 0 0	300	300
0 1 0 1	600	600
0 1 1 0	1,200	1,200
0 1 1 1	1,050	2,000
1 0 0 0	2,400	2,400
1 0 0 1	4,800	4,800
1 0 1 0	7,200	1,800
1 0 1 1	9,600	9,600
1 1 0 0	38.4K	19.2K
1 1 0 1	Timer	Timer
1 1 1 0	IP4–16X	IP4–16X
1 1 1 1	IP4–1X	IP4–1X

The receiver clock is always a 16X clock except for CSRA[7:4] = 1111.

CSRA[3:0] – Channel A Transmitter Clock Select
This field selects the baud rate clock for the channel A transmitter. The field definition is as per CSRA[7:4] except as follows:

	Baud Rate	
CSRA[3:0]	ACR[7] = 0	ACR[7] = 1
1 1 1 0	IP3 – 16X	IP3 – 16X
1 1 1 1	IP3 – 1X	IP3 – 1X

The transmitter clock is always a 16X clock except for CSRA[3:0] = 1111.

CSRB – Channel B Clock Select Register

CSRB[7:4] – Channel B Receiver Clock Select
This field selects the baud rate clock for the channel B receiver. The field definition is as per CSRA[7:4] except as follows:

	Baud Rate	
CSRB[7:4]	ACR[7] = 0	ACR[7] = 1
1 1 1 0	IP2 – 16X	IP2 – 16X
1 1 1 1	IP2 – 1X	IP2 – 1X

The receiver clock is always a 16X clock except for CSRB[7:4] = 1111.

CSRB[3:0] – Channel B Transmitter Clock Select
This field selects the baud rate clock for the channel B transmitter. The field definition is as per CSRA[7:4] except as follows:

	Baud Rate	
CSRB[3:0]	ACR[7] = 0	ACR[7] = 1
1 1 1 0	IP5 – 16X	IP5 – 16X
1 1 1 1	IP5 – 1X	IP5 – 1X

The transmitter clock is always a 16X clock except for CSRB[3:0] = 1111

CRA – Channel A Command Register

CRA is a register used to supply commands to channel A. Multiple commands can be specified in a single write to CRA as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

CRA[6:4] – Channel A Miscellaneous Commands
The encoded value of this field may be used to specify a single command as follows:

CRA[6:4]	COMMAND
0 0 0	No command.
0 0 1	Reset MR pointer. Causes the channel A MR pointer to point to MR1.
0 1 0	Reset receiver. Resets the channel A receiver as if a hardware reset had been applied. The receiver is disabled and the FIFO is flushed.
0 1 1	Reset transmitter. Resets the channel A transmitter as if a hardware reset had been applied.
1 0 0	Reset error status. Clears the channel A Received Break, Parity Error, Framing Error, and Overrun Error bits in the status register (SRA[7:4]). Used in character mode to clear OE status (although RB, PE, and FE bits will also be cleared) and in block mode to clear all error status after a block of data has been received.
1 0 1	Reset channel A break change interrupt. Causes the channel A break detect change bit in the interrupt status register (ISR[2]) to be cleared to zero.
1 1 0	Start break. Forces the TXDA output low (spacing). If the transmitter is empty the start of the break condition will be delayed up to two bit times. If the transmitter is active the break begins when transmission of the character is completed. If a character is in the THR, the start of the break will be delayed until that character, or any others loaded subsequently are transmitted. The transmitter must be enabled for this command to be accepted.
1 1 1	Stop Break. The TXDA line will go high (marking) within two bit times. TXDA will remain high for one bit time before the next character, if any, is transmitted.

CRA[3] – Disable Channel A Transmitter
This command terminates transmitter operation and resets the TxRDY and TxEMT status bits. However, if a character is being transmitted or if a character is in the THR when the transmitter is disabled, the transmission of the character(s) is completed before assuming the inactive state.

CRA[2] – Enable Channel A Transmitter
Enables operation of the channel A transmitter. The TxRDY status bit will be asserted.

CRA[1] – Disable Channel A Receiver
This command terminates operation of the receiver immediately – a character being received will be lost. The command has no effect on the receiver status bits or any other control registers. If the special multidrop mode is programmed, the receiver operates even if it is disabled. See Operation section.

CRA[0] – Enable Channel A Receiver
Enables operation of the channel A receiver. If not in the special wakeup mode, this also forces the receiver into the search for start-bit state.

CRB – Channel B Command Register

CRB is a register used to supply commands to channel B. Multiple commands can be specified in a single write to CRB as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

The bit definitions for this register are identical to the bit definitions for CRA, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

SRA – Channel A Status Register

SRA[7] – Channel A Received Break
This bit indicates that an all zero character of the programmed length has been received without a stop bit. Only a single FIFO position is occupied when a break is received: further entries to the FIFO are inhibited until the RxDA line returns to the marking state for at least one-half a bit time (two successive edges of the internal or external 1x clock).

When this bit is set, the channel A 'change in break' bit in the ISR (ISR[2]) is set. ISR[2] is also set when the end of the break condition, as defined above, is detected.

The break detect circuitry can detect breaks that originate in the middle of a received character. However, if a break begins in the middle of a character, it must persist until at least the end of the next character time in order for it to be detected.

SRA[6] – Channel A Framing Error
This bit, when set, indicates that a stop bit was not detected when the corresponding data character in the FIFO was received. The stop bit check is made in the middle of the first stop bit position.

SRA[5] – Channel A Parity Error
This bit is set when the 'with parity' or 'force parity' mode is programmed and the corresponding character in the FIFO was received with incorrect parity.

In the special multidrop mode the parity error bit stores the received A/D bit.

SRA[4] – Channel A Overrun Error
This bit, when set, indicates that one or more characters in the received data stream have been lost. It is set upon receipt of a new character when the FIFO is full and a character is already in the receive shift register waiting for an empty FIFO position. When this occurs, the character in the receive shift register (and its break detect, parity error and framing error status, if any) is lost.

This bit is cleared by a 'reset error status' command.

SRA[3] – Channel A Transmitter Empty (TxEMTA)
This bit will be set when the channel A transmitter underruns, i.e., both the transmit holding register (THR) and the transmit shift register are empty. It is set after transmission of the last stop bit of a character if no character is in the THR awaiting transmission. It is reset when the THR is loaded by the CPU or when the transmitter is disabled.

SRA[2] – Channel A Transmitter Ready (TxRDYA)
This bit, when set, indicates that the THR is empty and ready to be loaded with a character. This bit is cleared when the THR is loaded by the CPU and is set when the character is transferred to the transmit shift register. TxRDY is reset when the transmitter is disabled and is set when the transmitter is first enabled, viz., characters loaded into the THR while the transmitter is disabled will not be transmitted.

SRA[1] – Channel A FIFO Full (FFULLA)
This bit is set when a character is transferred from the receive shift register to the receive FIFO and the transfer causes the FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full. FFULL will not be reset when the CPU reads the RHR.

SRA[0] – Channel A Receiver Ready (RxRDYA)
This bit indicates that a character has been received and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR, if after this read there are no more characters still in the FIFO

SRB – Channel B Status Register

The bit definitions for this register are identical to the bit definitions for SRA, except that all status applies to the channel B receiver and transmitter and the corresponding inputs and outputs.

OPCR – Output Port Configuration Register

OPCR[7] – OP7 Output Select
This bit programs the OP7 output to provide one of following:
- The complement of OPR[7]
- The channel B transmitter interrupt output, which is the complement of TxRDYB. When in this mode OP7 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[6] – OP6 Output Select
This bit programs the OP6 output to provide one of the following:
- The complement of OPR[6]
- The channel A transmitter interrupt output, which is the complement of TxRDYA. When in this mode OP6 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[5] – OP5 Output Select
This bit programs the OP5 output to provide one of the following:
- The complement of OPR[5]
- The channel B receiver interrupt output, which is the complement of ISR[5]. When in this mode OP5 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[4] – OP4 Output Select
This bit programs the OP4 output to provide one of the following:
- The complement of OPR[4]
- The channel A receiver interrupt output, which is the complement of ISR[1]. When in this mode OP4 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[3:2] – OP3 Output Select
This field programs the OP3 output to provide one of the following:
- The complement of OPR[3]
- The counter/timer output, in which case OP3 acts as an open collector output. In the timer mode, this output is a square wave at the programmed frequency. In the counter mode, the output remains high until terminal count is reached, at which time it goes low. The output returns to the high state when the counter is stopped by a stop counter command. Note that this output is not masked by the contents of the IMR.
- The 1X clock for the channel B transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a free running 1X clock is output.
- The 1X clock for the channel B receiver, which is the clock that samples the received data. If data is not being received, a free running 1X clock is output.

OPCR[1:0] – OP2 Output Select
This field programs the OP2 output to provide one of the following:
- The complement of OPR[2]
- The 16X clock for the channel A transmitter. This is the clock selected by CSRA[3:0], and will be 1X clock if CSRA[3:0] = 1111.
- The 1X clock for the channel A transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a free running 1X clock is output.
- The 1X clock for the channel A receiver, which is the clock that samples the received data. If data is not being received, a free running 1X clock is output.

ACR – Auxiliary Control Register

ACR[7] – Baud Rate Generator Set Select
This bit selects one of two sets of baud rates to be generated by the BRG:

Set 1: 50, 110, 134.5, 200, 300, 600, 1.05K, 1.2K, 2.4K, 4.8K, 7.2K, 9.6K, and 38.4K baud.

Set 2: 75, 110, 134.5, 150, 300, 600, 1.2K, 1.8K, 2.0K, 2.4K, 4.8K, 9.6K, and 19.2K baud.

The selected set of rates is available for use by the channel A and B receivers and transmitters as described in CSRA and CSRB. Baud rate generator characteristics are given in table 3.

ACR[6:4] – Counter/Timer Mode and Clock Source Select
This field selects the operating mode of the counter/timer and its clock source as shown in table 4.

ACR[3:0] – IP3, IP2, IP1, IP0 Change of State Interrupt Enable
This field selects which bits of the Input Port Change register (IPCR) cause the input change bit in the interrupt status register (ISR[7]) to be set. If a bit is in the 'on' state, the setting of the corresponding bit in the IPCR will also result in the setting of ISR[7], which results in the generation of an interrupt output if IMR[7] = 1. If a bit is in the 'off' state, the setting of that bit in the IPCR has no effect on ISR[7].

Table 3. BAUD RATE GENERATOR CHARACTERISTICS CRYSTAL or CLOCK = 3.6864MHz

NOMINAL RATE (BAUD)	ACTUAL 16X CLOCK (KHz)	ERROR (PERCENT)
50	0.8	0
75	1.2	0
110	1.759	-0.069
134.5	2.153	0.059
150	2.4	0
200	3.2	0
300	4.8	0
600	9.6	0
1050	16.756	-0.260
1200	19.2	0
1800	28.8	0
2000	32.056	0.175
2400	38.4	0
4800	76.8	0
7200	115.2	0
9600	153.6	0
19.2K	307.2	0
38.4K	614.4	0

NOTE:
Duty cycle of 16X clock is 50% ±1%

Table 4. ACR [6:4] FIELD DEFINITION

ACR [6:4]	MODE	CLOCK SOURCE
0 0 0	Counter	External (IP2)[1]
0 0 1	Counter	TXCA – 1X clock of channel A transmitter
0 1 0	Counter	TXCB – 1X clock of channel B transmitter
0 1 1	Counter	Crystal or external clock (X1/CLK) divided by 16
1 0 0	Timer	External (IP2)[1]
1 0 1	Timer	External (IP2) divided by 16[1]
1 1 0	Timer	Crystal or external clock (X1/CLK)
1 1 1	Timer	Crystal or external clock (X1/CLK) divided by 16

[1]In these modes, the channel B receiver clock should normally be generated from the baud rate generator.

IPCR – Input Port Change Register

IPCR[7] – IP3, IP2, IP1, IP0 Change of State
These bits are set when a change of state, as defined in the Input Port section of this data sheet, occurs at the respective input pins. They are cleared when the IPCR is read by the CPU. A read of the IPCR also clears ISR[7], the input change bit in the interrupt status register.

The setting of these bits can be programmed to generate an interrupt to the CPU.

IPCR[3:0] – IP3, IP2, IP1, IP0 Current State
These bits provide the current state of the respective inputs. The information is unlatched and reflects the state of the input pins at the time the IPCR is read.

ISR – Interrupt Status Register

This register provides the status of all potential interrupt sources. The contents of this register are masked by the interrupt mask register (IMR). If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the reading of the ISR – the true status will be provided regardless of the contents of the IMR. The contents of this register are initalized to 00_{16} when the DUART is reset.

ISR[7] – Input Port Change Status
This bit is a '1' when a change of state has occurred at the IP0, IP1, IP2, or IP3 inputs and that event has been selected to cause an interrupt by the programming of ACR[3:0]. The bit is cleared when the CPU reads the IPCR.

ISR[6] – Channel B Change in Break
This bit, when set, indicates that the channel B receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a channel B 'reset break change interrupt' command.

ISR[5] – Channel B Receiver Ready or FIFO Full
The function of this bit is programmed by MR1B[6]. If programmed as receiver ready, it indicates that a character has been received in channel B and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the channel B FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

ISR[4] – Channel B Transmitter Ready
This bit is a duplicate of TxRDYB (SRB[2]).

ISR[3] – Counter Ready
In the counter mode, this bit is set when the counter reaches terminal count and is reset when the counter is stopped by a stop counter command.

In the timer mode, this bit is set once each cycle of the generated square wave (every other time that the counter/timer reaches zero count). The bit is reset by a stop counter command. The command, however, does not stop the counter/timer.

ISR[2] – Channel A Change in Break
This bit, when set, indicates that the channel A receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a channel A 'reset break change interrupt' command.

ISR[1] – Channel A Receiver Ready or FIFO Full
The function of this bit is programmed by MR1A[6]. If programmed as receiver ready, it indicates that a character has been received in channel A and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the channel A FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because

the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

ISR[0] – Channel A Transmitter Ready
This bit is a duplicate of TxRDYA (SRA[2]).

IMR – Interrupt Mask Register

The programming of this register selects which bits in the ISR cause an interrupt output. If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the programmable interrupt outputs OP3–OP7 or the reading of the ISR.

CTUR and CTLR – Counter/ Timer Registers

The CTUR and CTLR hold the eight MSB's and eight LSB's respectively of the value to be used by the counter/timer in either the counter or timer modes of operation. The minimum value which may be loaded into the CTUR/CTLR registers is 0002_{16}. Note that these registers are write-only and cannot be read by the CPU.

In the timer (programmable divider) mode, the C/T generates a square wave with a period of twice the value (in clock periods) of the CTUR and CTLR. If the value in CTUR or CTLR is changed, the current half-period will not be affected, but subsequent half periods will be. In this mode the C/T runs continuously. Receipt of a start counter command (read with A4–A1 = 1110) causes the counter to terminate the current timing cycle and to begin a new cycle using the values in CTUR and CTLR. The counter ready status bit (ISR[3]) is set once each cycle of the square wave. The bit is reset by a stop counter command (read with A4–A1 = 1111). The command, however, does not stop the C/T. The generated square wave is output on OP3 if it is programmed to be the C/T output.

On power up and after reset, the timer/ counter runs in timer mode and can only be restarted. Because it cannot be shut off or stopped, and runs continuously in timer mode, it is recommended that at initialization, the output port, OP3, should be masked off through the OPCR[3:2] = 00 until the T/C is programmed to the desired operational state.

In the counter mode, the C/T counts down the number of pulses loaded into CTUR and CTLR by the CPU. Counting begins upon receipt of a start counter command. Upon reaching terminal count (0000_{16}), the counter ready interrupt bit (ISR[3]) is set. The counter continues counting past the terminal count until stopped by the CPU. If OP3 is programmed to be the output of the C/T, the output remains high until terminal count is reached, at which time it goes low. The output returns to the high state and ISR[3] is cleared when the counter is stopped by a stop counter command. The CPU may change the values of CTUR and CTLR at any time, but the new count becomes effective only on the next start counter command. If new values have not been loaded, the previous count values are preserved and used for the next count cycle.

In the counter mode, the current value of the upper and lower 8 bits of the counter (CTU, CTL) may be read by the CPU. It is recommended that the counter be stopped when reading to prevent potential problems which may occur if a carry from the lower 8-bits to the upper 8-bits occurs between the times that both halves of the counter are read. However, note that a subsequent start counter command will cause the counter to begin a new count cycle using the values in CTUR and CTLR.

IVR – Interrupt Vector Register

This register contains the interrupt vector. The register is initialized to H'0F' by RESET. The contents of the register are placed on the data bus when the DUART responds to a valid interrupt acknowledge cycle.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage (except X1/CLK)		2.0			V
V_{IH}	Input high voltage (X1/CLK)		4.0			V
V_{OL}	Output low voltage	$I_{OL} = 2.4mA$			0.4	V
V_{OH}	Output high voltage (except o.c. outputs)	$I_{OH} = -400\mu A$	2.4			V
I_{IL}	Input leakage current	$V_{IN} = 0$ to V_{CC}	−10		10	μA
I_{LL}	Data bus 3-state leakage current	$V_O = 0.4$ to V_{CC}	−10		10	μA
I_{X1L}	X1/CLK low input current	$V_{IN} = 0$, X2 grounded	−4.0	−2.0	0.0	mA
		$V_{IN} = 0$, X2 floated	−3.0	−1.5	0.0	mA
I_{X1H}	X1/CLK high input current	$V_{IN} = V_{CC}$, X2 grounded	−1.0	0.2	1.0	mA
		$V_{IN} = V_{CC}$, X2 floated	0.0	3.5	10.0	mA
I_{X2L}	X2 low input current	$V_{IN} = 0$, X1/CLK floated	−100	−30	0.0	μA
I_{X2H}	X2 high input current	$V_{IN} = V_{CC}$, X1/CLK floated	0.0	+30	100	μA
I_{OC}	Open collector output leakage current	$V_O = 0.4$ to V_{CC}	−10		10	μA
I_{CC}	Power supply current				150	mA

NOTES:

4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all inputs except X1/CLK swing between 0.4V and 2.4V with a transition time of 20ns maximum. For X1/CLK, this swing is between 0.4V and 4.4V. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical supply voltages, and typical processing parameters.
7. Test condition for outputs: $C_L = 150pF$, except interrupt outputs. Test condition for interrupt outputs: $C_L = 50pF$, $R_L = 2.7K$ ohm to V_{CC}.
8. This specification will impose maximum 68000 CPU CLK to 6MHz. Higher CPU CLK can be used if repeating bus reads are not performed. Consecutive write operations to the same command register require at least three edges of the X1 clock between writes.
9. This specification imposes a lower bound on CSN and IACKN low, guaranteeing that it will be low for at least 1 CLK period. This requirement is made on CSN only to insure assertion of DTACKN and not to guarantee operation of the part.
10. This specification is made only to insure that DTACKN is asserted with respect to the rising edge of the X1/CLK pin as shown in the timing diagram, not to guarantee operation of the part. If the set-up time is violated, DTACKN may be asserted as shown, or may be asserted one clock cycle later.

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4, 5, 6, 7]

PARAMETER		LIMITS			UNIT
		Min	Typ	Max	
Reset Timing (figure 1)					
t_{RES}	RESETN pulse width	1.0			μs
Bus Timing (figures 2, 3, 4)					
t_{AS}	A1–A4 set-up time to CSN low	10			ns
t_{AH}	A1–A4 hold time from CSN high	0			ns
t_{RWS}	RWN set-up time to CSN high	0			ns
t_{RWH}	RWN holdup time to CSN high	0			ns
t_{CSW}[8]	CSN high pulse width	160			ns
t_{CSD}[9]	CSN or IACKN high from DTACKN low	20			ns
t_{DD}	Data valid from CSN or IACKN low			175	ns
t_{DF}	Data bus floating from CSN or IACKN high			100	ns
t_{DS}	Data set-up time to CLK high	100			ns
t_{DH}	Data hold time from CSN high	0			ns
t_{DAL}	DTACKN low from read data valid	0			ns
t_{DCR}	DTACKN low (read cycle) from CLK high			125	ns
t_{DCW}	DTACKN low (write cycle) from CLK high			125	ns
t_{DAH}	DTACKN high from CSN or IACKN high			100	ns
t_{DAT}	DTACKN high impedance from CSN or IACKN high			125	ns
t_{CSC}[10]	CSN or IACKN set-up time to clock high	90			ns
Port Timing (figure 5)					
t_{PS}	Port input set-up time to CSN low	0			ns
t_{PH}	Port input hold time from CSN high	0			ns
t_{PD}	Port output valid from CSN high			400	ns
Interrupt Reset Timing (figure 6)					
t_{IR}	INTRN, or OP3–OP7 when used as interrupts, negated from:				
	Read RHR (RxRDY/FFULL interrupt)			300	ns
	Write THR (TxRDY interrupt)			300	ns
	Reset command (delta break interrupt)			300	ns
	Stop C/T command (counter interrupt)			300	ns
	Read IPCR (input port change interrupt)			300	ns
	Write IMR (clear of interrupt mask bit)			300	ns
Clock Timing (figure 7)					
t_{CLK}	X1/CLK high or low time	100			ns
f_{CLK}	X1/CLK frequency	2.0	3.6864	4.0	MHz
t_{CTC}	CTCLK high or low time	100			ns
f_{CTC}	CTCLK frequency	0		4.0	MHz
t_{RX}	RXC high or low time	220			ns
f_{RX}	RXC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
t_{TX}	TXC high or low time	220			ns
f_{TX}	TXC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
Transmitter Timing (figure 8)					
t_{TXD}	TXD output delay from TXC low			350	ns
t_{TCS}	Output delay from TxC low to TxD data output			150	ns
Receiver Timing (figure 9)					
t_{RXS}	RXD data set-up time to RXC high	240			ns
t_{RXH}	RXD data hold time from RXC high	200			ns

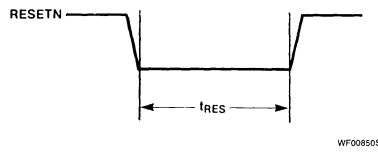


Figure 1. Reset Timing

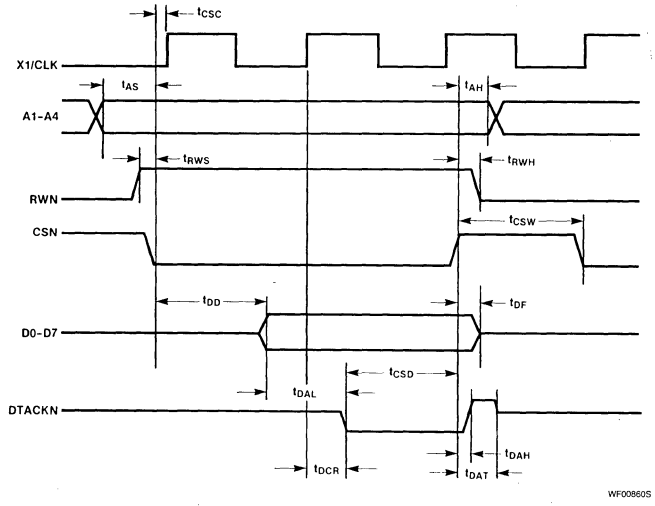


Figure 2. Bus Timing (Read Cycle)

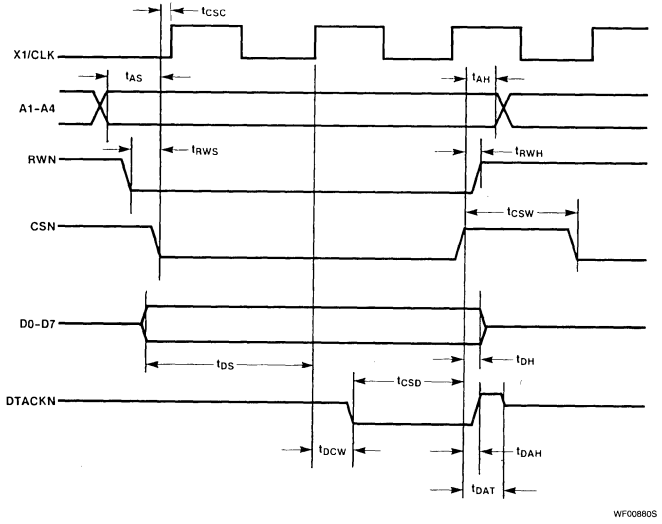


Figure 3. Bus Timing (Write Cycle)

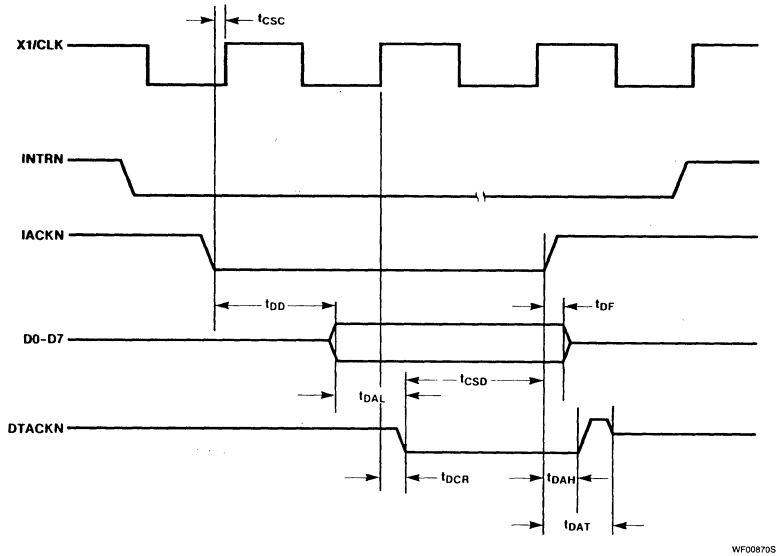


Figure 4. Interrupt Cycle Timing

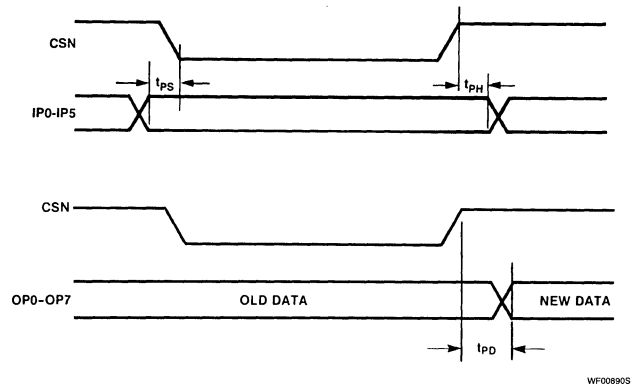
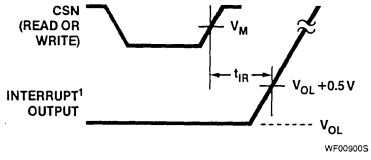


Figure 5. Port Timing



NOTES:
1. INTRN or OP3 – OP7 when used as interrupt outputs.
2. The test for open drain outputs is intended to guarantee switching of the output transistor. Measurement of this response is referenced from the midpoint of the switching signal, V_M, to a point 0.5 volts above V_{OL}. This point represents noise margin that assures true switching has occurred. Beyond this level, the effects of external circuitry and test environment are pronounced and can greatly affect the resultant measurement.

Figure 6. Interrupt Timing

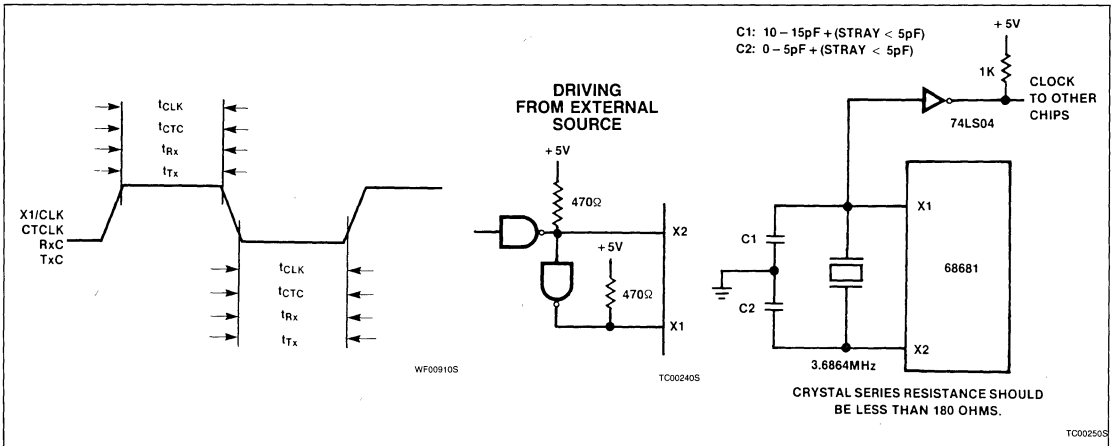


Figure 7. Clock Timing

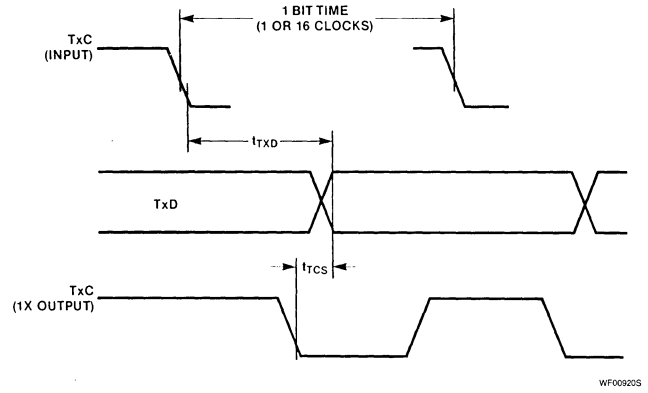


Figure 8. Transmit Timing

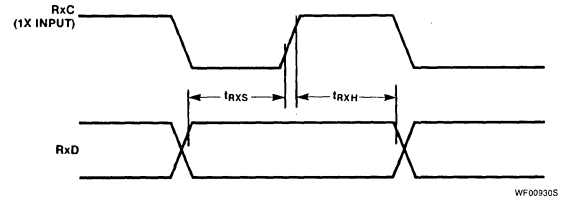
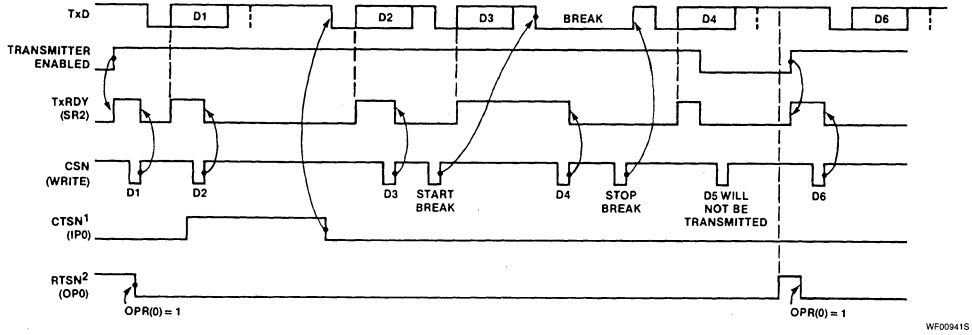
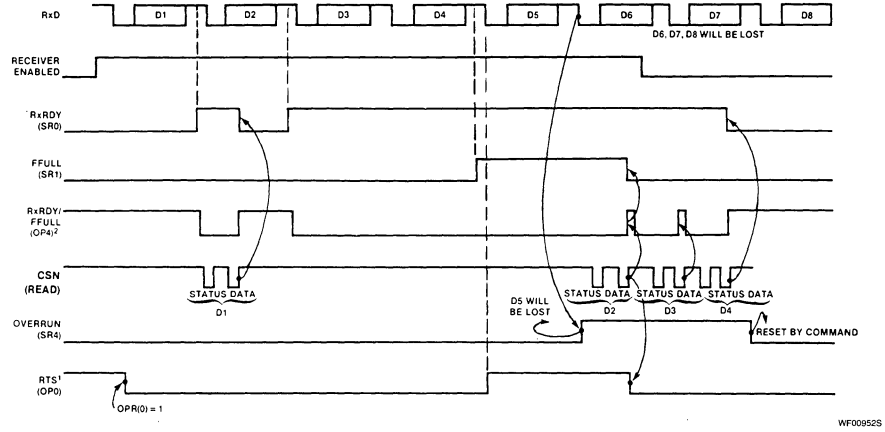


Figure 9. Receive Timing



NOTES:
1. Timing shown for MR2(4) = 1.
2. Timing shown for MR2(5) = 1.

Figure 10. Transmitter Timing



NOTES:
1. Timing shown for MR1(7) = 1.
2. Shown for OPCR(4) and MR1(6) = 0.

Figure 11. Receiver Timing

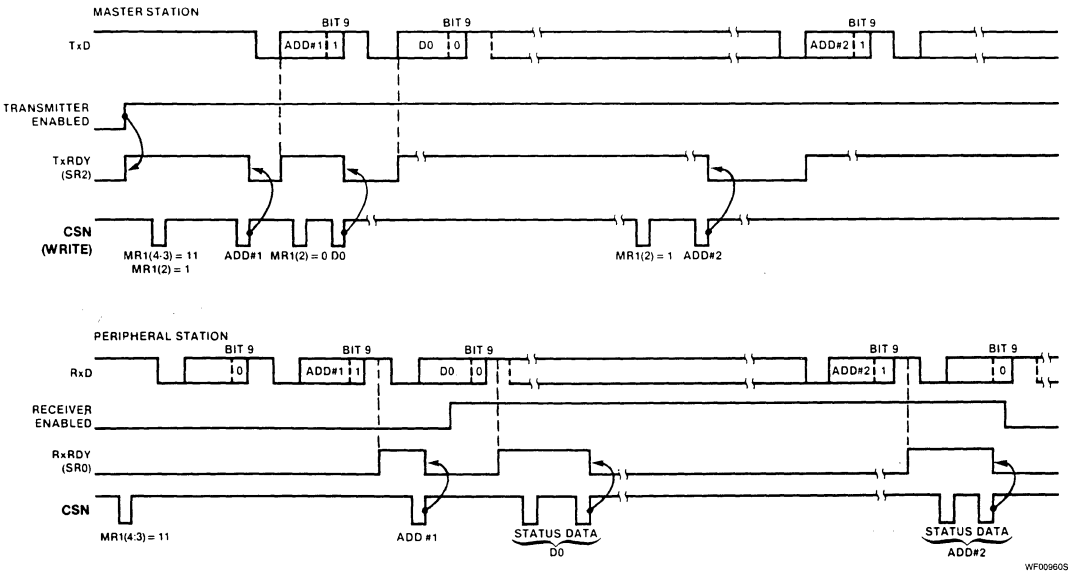


Figure 12. Wake up Mode

Signetics

SCC68905
Basic Memory Access Controller (BMAC)

Preliminary Specification

Microprocessor Products

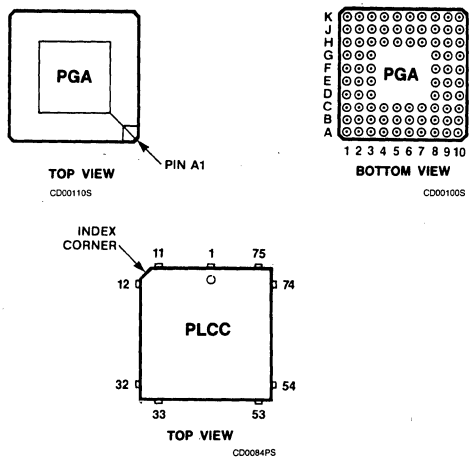
DESCRIPTION

The Signetics SCC68905 Basic Memory Access Controller (BMAC) is designed for the S68000 family (68000, 68010 and 68012) of microprocessors. It has a 24-bit logical address and a 24-bit physical address space. The BMAC mediates all accesses between the processor and system memory by controlling the memory hierarchy, translating virtual addresses into real addresses, supporting context switches and providing protection against unauthorized memory access. The operating system allocates the system memory and the BMAC implements these functions dynamically. The BMAC also provides support for a local memory. The SCC68905 is constructed using Signetics CMOS VLSI technology.

FEATURES

- **Mediates memory reference between processor and memory**
- **Functions as a demand-driven MMU and a cache controller**
- **No wait states on cache hits**
- **MMU action occurs in parallel with cache action**
- **Little mutual bus interference with up to four processors**
- **Ensures correct data in required local memory**
- **Variable length segments with paging, variable length contiguous segments, or paged only**
- **Allows partitioning the virtual address space into one to four regions**
- **Selective flushing by regions**
- **Provides four protection types at the page or segment level**
- **Maintains four history bits**
- **Configurable cache block size of 2, 4, 8, or 16 bytes wide**
- **Configurable cache depth of 512, 1024, or 2048 blocks**
- **Configurable page size of 1k, 2k, 4k, or 8k bytes**
- **Can use sequential accesses to fill cache if available on the system**

PIN CONFIGURATION



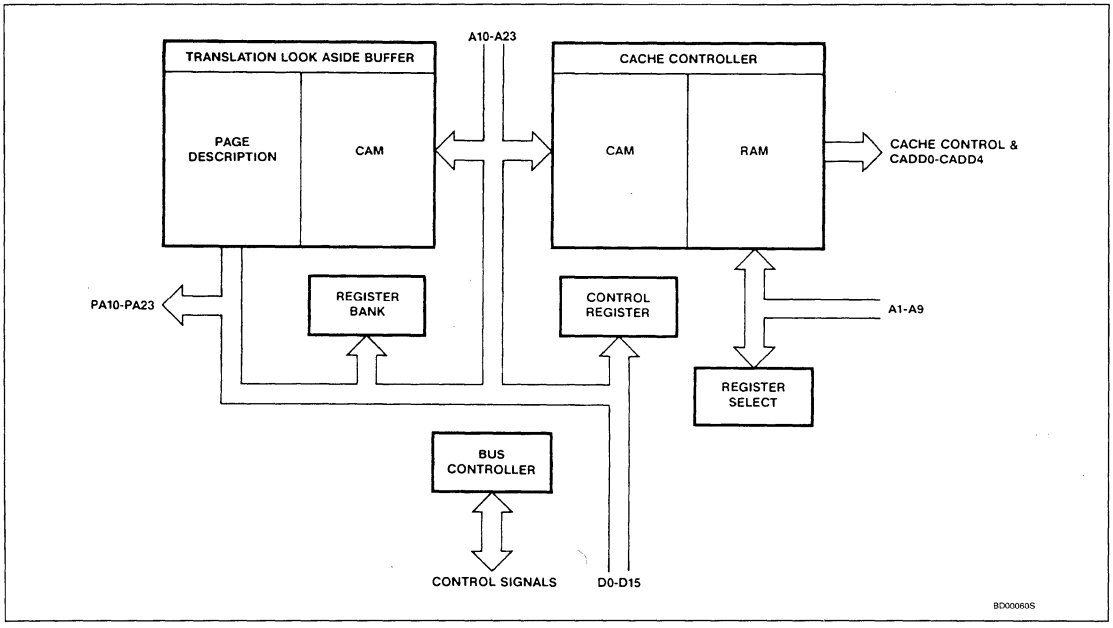
PGA	Function	PGA	Function	PLCC	Function	PLCC	Function
A-1	A3	F-1	V_{CC}	1	CADD3	43	MASN
A-2	A5	F-2	D15	2	CADD4	44	V_{SS}
A-3	A7	F-3	D14	3	A9	45	DTACKINN
A-4	A8	F-8	ASN	4	A8	46	CCEN
A-5	CADD3	F-9	BERRINN	5	A7	47	LACLR
A-6	CADD2	F-10	V_{CC}	6	A6	48	DTACKOUTN
A-7	CADD0	G-1	D13	7	A5	49	PA10
A-8	A12	G-2	D11	8	A4	50	PA11
A-9	PA23	G-3	D9	9	A3	51	PA12
A-10	A15	G-8	PA19	10	A2	52	PA13
B-1	FC2	G-9	PA20	11	A1	53	PA14
B-2	A1	G-10	CLOCK	12	FC0	54	PA15
B-3	A2	H-1	D12	13	FC1	55	PA16
B-4	A6	H-2	D7	14	FC2	56	PA17
B-5	CADD4	H-3	D5	15	FC3	57	PA18
B-6	CADD1	H-4	D1	16	RESETN	58	PA19
B-7	A10	H-5	MISS	17	CSN	59	PA20
B-8	PA22	H-6	DTACKINN	18	R/WN	60	CLOCK
B-9	A13	H-7	PA11	19	NC	61	BERRINN
B-10	A17	H-8	PA15	20	UDSN	62	ASN
C-1	FC3	H-9	PA16	21	LDSN	63	V_{CC}
C-2	FC1	H-10	PA18	22	V_{CC}	64	A23
C-3	FC0	J-1	D10	23	D15	65	A22
C-4	A4	J-2	D6	24	D14	66	A21
C-5	A9	J-3	D4	25	D13	67	A20
C-6	V_{SS}	J-4	D0	26	D12	68	A19
C-7	A11	J-5	BERROUTN	27	D11	69	A18
C-8	PA21	J-6	V_{SS}	28	D10	70	A17
C-9	A14	J-7	DTACKOUTN	29	D9	71	A16
C-10	A19	J-8	PA13	30	D8	72	A15
D-1	R/WN	J-9	PA14	31	D7	73	A14
D-2	CSN	J-10	PA17	32	D6	74	A13
D-3	RESETN	K-1	D8	33	D5	75	PA21
D-8	A16	K-2	D3	34	D4	76	PA22
D-9	A18	K-3	D2	35	D3	77	PA23
D-10	A20	K-4	LOCAL	36	D2	78	A12
E-1	LDSN	K-5	CWEN	37	D1	79	A11
E-2	NC	K-6	MASN	38	D0	80	A10
E-3	UDSN	K-7	CCEN	39	LOCAL	81	CADD0
E-8	A21	K-8	LACLR	40	BERROUTN	82	CADD1
E-9	A22	K-9	PA10	41	MISS	83	V_{SS}
E-10	A23	K-10	PA12	42	CWEN	84	CADD2

Basic Memory Access Controller (BMAC) SCC68905

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C
Ceramic PGA	SCC68905CCP84
Plastic LCC	SCC68905CCA84

BLOCK DIAGRAM



The BMAC provides the basis for a total memory management solution. The BMAC combines a memory management unit (MMU) and a cache controller in the same package. The MMU contains locations for 32 descriptors which are loaded on demand. Also, the BMAC contains hardware support to define the region context and boundaries. The cache controller provides the matching and control logic for a virtual cache which reduces memory access times by at least one clock cycle. With the BMAC, a cache hit will result in a no wait state access for even the fastest microprocessors.

The MMU and the cache are programmer configurable. The BMAC allows the system programmer to choose segments, pages or paged segments and the cache size is configurable in both width and depth.

The BMAC is functionally compatible with the MAC (SCC68910); updating the MMU is accomplished by software on the BMAC and by hardware on the MAC.

FUNCTIONAL DESCRIPTION

In most real modular systems, an 8MHz 68010 with an MMU needs between two and four wait states when it accesses memory. Speed and memory management pose problems for the system designer even for the 8MHz 68010 and it will be further aggravated when the higher speed 68010s and 68020s become available. This problem will be slightly reduced when faster DRAMs are available; but it is clear that a 16MHz 680xx and MMU will need at least 4 wait states when accessing memory unless that memory is not on a bus. The BMAC reduces this problem by functioning as a cache controller and a memory management unit. It performs access control of the memory hierarchy, translation of virtual to real addresses, supports context changes, and protects against unauthorized access during memory processing.

The operating system is responsible for allocating memory and access rights. The memory hierarchy can consist of three intermediate access store elements; a cache, a local memory, and the system memory. The cache memory element contains a copy of scattered sections of the local and system memories and is transparent to the software. The cache is a small, fast memory that allows the processor to see the cache access time instead of the usual system memory access time for 90% of its accesses and reduces the bus usage by 85%. The BMAC implements these functions dynamically.

Local memories can be attached to each processor in a multiprocessor system. The BMAC provides facilities to control the movement of code and data in and out of these local memories. This feature is transparent to the user software.

A demand paged virtual memory system can be implemented with one or more BMACs. The BMAC makes a multiprocessor system practical by reducing bus utilization. The BMAC has a special provision that allows multiple processing or processors to operate even if they share common data. If an 85% reduction in bus usage is not adequate, a local memory can then be attached to each processor which will reduce it further.

PIN DESCRIPTION

Signal names ending in 'N' are active low. All other signals are active high. In this data sheet, signals are discussed using the terms 'active' and 'inactive' or 'asserted' and 'negated' independent of whether the signal is active in the high (logic one) state or the low (logic 0) state. Refer to the individual pin descriptions for the definition of the active level of each signal.

MNEMONIC	PIN NO.		TYPE	DESCRIPTION
	PLCC	PGA		
A23 – 10	64 – 74, 78 – 80	E-10 – E-8 D-10,C-10, D-9,B-10, D-8,A-10, C-9,B-9, A-8,C-7, B-7	I	**Address Lines:** Active high. High order virtual address lines issued by the processor.
A9 – 1	3 – 11	C-5,A-4, A-3,B-4, A-2,C-4, A-1,B-3, B-2	I	**Address Lines:** Active high. These lines are the page address or when CSN is asserted, the BMAC register address. These are I/O lines on the MAC.
PA23 – 10	77 – 75, 59 – 49	A-9,B-8, C-8,G-9, G-8,H-10, J-10,H-9, H-8,J-9, J-8,K-10, H-7,K-9	O	**Physical Address:** Active high, three statable. Some or all of pins PA12-10 may always be tri-state on the BMAC, depending on the page size programmed in the PDR. This is done to maintain campatibility with the MAC.
CADD4 – 0	2, 1, 84, 82, 81	B-5,A-5, A-6,B-6, A-7	O	**Cache Address:** Active high. High order cache address lines.
D15 – 0	23 – 38	F-2,F-3, G-1,H-1, G-2,J-1 G-3,K-1, H-2,J-2, H-3,J-3, K-2,K-3, H-4,J-4	I/O	**Data Lines:** Active high, three-statable. These lines are used to transfer data between the processor and the BMAC when the CSN line is asserted.
FC2 – 0	14 – 12	B-I,C-2,C-3	I	**Function Codes:** Active high. These lines are issued by the processor.
FC3	15	C-1	I	**Function Code:** Active high. Issued externally to differentiate I/O cycles from processor cycles.
ASN	62	F-8	I	**Address Strobe:** Active low. Issued by processor.
MASN	43	K-6	O	**Memory Address Strobe:** Active low. To memory issued by BMAC.
UDSN	20	E-3	I	**Upper Data Strobe:** Active low, three-statable. This signal is an I/O line on the MAC. This signal line requires an external tri-state buffer since the 680xx will not make this line high impedant when the BMAC is accessing memory.
LDSN	21	E-1	I	**Lower Data Strobe:** Active low, three-statable. This signal is an I/O line on the MAC. This signal line requires an external tri-state buffer since the 680xx will not make this line high impedant when the BMAC is accessing memory.
DTACKINN	45	H-6	I	**Data Transfer Acknowledge In:** Active low. Data acknowledge input from memory.
DTACKOUTN	48	J-7	O	**Data Transfer Acknowledge Out:** Active low, open-drain. Data acknowledge output to processor.
R/WN	18	D-1	I	**Read/Write:** Active high for read, low for write. This signal is an I/O line on the MAC. This signal line requires an external tri-state buffer since the 680xx will not make this line high impedant when the BMAC is accessing memory.
CSN	17	D-2	I	**Chip Select:** Active low.
CCEN	46	K-7	O	**Cache Chip Enable:** Active low. This signal has to be qualified externally by the data strobe signals.
CWEN	42	K-5	O	**Cache Write Enable:** Active low.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	DESCRIPTION
	PLCC	PGA		
LOCAL	39	K-4	O	**Local:** Active high, three-statable. Local memory enable.
BERRINN	61	F-9	I	**Bus Error In:** Active low. Bus error signal, input from bus. During a memory access, DTACKINN and BERRINN cannot both be asserted unless BERRINN is asserted two clock cycles before DTACKINN.
BERROUTN	40	J-5	O	**Bus Error Out:** Active low, open-drain. Bus error signal, output to processor.
CLOCK	60	G-10	I	**Clock:** Active high. Clock input.
RESETN	16	D-3	I	**Reset:** Active low. When asserted this signal will reset the BMAC.
MISS	41	H-5	O	**MISS:** Active high. When cache is enabled, it provides a cache read miss signal. It enables an external counter to be incremented on a block fetch during sequential mode.
LACLR	47	K-8	O	**Latch Clear:** Active high. This line is pulsed low to clear the counter for a block fetch.
V_{CC}	22,63	F-1,F-10	I	+5volt ±5% power input.
V_{SS}	83,44	C-6,J-6	I	Power ground return.

MEMORY STRUCTURE

The BMAC uses a memory which is divided into variable length segments, which in turn can be divided into fixed length pages. Even if pages are not used, the BMAC measures the length of the segment in multiples of 1, 2, 4, or 8k bytes. As any segment can be any number of pages in length, the system can work if desired with only two segments - program and data. Protection is provided on a segment basis or page basis. Segments, or pages, can have the following protection attributes:

1. Supervisor permission
2. Read permission
3. Write permission
4. Execute permission

These attributes are recognized and protected dynamically by the hardware. If a user attempts to access a segment without permission, a bus error (BERR) signal is generated.

Aside from protection attributes, the BMAC supports three additional attributes which are used to control the memory hierarchy.

1. Non-cacheable. This attribute forces associated segments or pages to never be cached. It forces segments or pages which are shared between processes to have only one copy which is always up to date.
2. Local. If the system has local memory then this attribute is used to force the associated page into the memory local to the processor and saves bus accesses.
3. Contiguous. This attribute, if set, indicates a segment that has no page table associated with it. Such segments are to be loaded and relocated in memory as a whole.

CACHE

A cache contains a copy of portions of program and data which are in the memory. Its associated principle works because most words of memory are accessed more than once within a relatively short time interval. The BMAC automatically stores each word read by the processor in the cache and, if the processor tries to read that word a second time, the BMAC provides the word from the cache instead of accessing the memory. When the processor writes a word, the BMAC implements a write-through policy, i.e., the word is written into both the cache and the memory. If a segment is marked as local, then this refers to the local memory and not the system memory. If the bus is not available to the processor, the BMAC allows the processor to proceed, and if the bus is granted before the next 'cache miss' or before the next write, the processor need not be aware that there was a bus access latency time.

Under certain conditions, e.g., when two processors are sharing access to a common segment of memory interleaved in time, it may not be desirable to cache the data. These segments can be marked as non-cacheable.

Sector-Associative Caches

Sector-associative caches are attempts to temper the idea of an associative cache with a dose of reality. Associative caches have an identifier attached to each cache word. The identifier is a content addressable memory (CAM) which holds the high order bits of the address. Basically, a sector-associative cache adds one assumption to the idea of associative caches. The assumption is that there is a working set of memory locations which in fact is reasonably small and tend to cluster.

In a sector-associative cache, the cache is divided into sub-caches. A sub-cache can contain a whole sector. Note that as far as the TLB is concerned, main memory could be logically partitioned into pages and segments, while as far as the cache is concerned, main memory could be logically partitioned into fixed size sectors. Since sectors are naturally large, filling a cache one sector at a time is not practical, therefore, sectors are partitioned into blocks made up of one, two, four or eight words. Thus, cache transfers imply block transfers to a particular sub-cache. Also note that not every block in a sector is necessarily in the cache.

An address, n bits in length, presented to the cache can be divided into three fields:

S bits (sector) – high order address bits which define the sector

B bits (block) – used to access the sub-cache (the algorithm will be given later)

O bits (offset) – used only if the block size is greater than the word size. The block size is the width of the data path between the cache memory and the memory. It can be wider than the processor's data bus.

ADDRESS FORMAT

Sector (S Bits)	Block (B Bits)	Offset (O Bits)

To determine if a sector is in the cache, the controller divides any virtual address it receives into sector, block, and offset fields. If the sector field matches the contents of one of the CAM positions, then the next step is to see if the particular block is present. Thus, a

presence bit per cache block is needed. When a block is read into the cache for the first time the presence bit is set. Subsequently, when the block is accessed, the cache controller checks to see if the sector has been mapped into a sub-cache and that the presence bit is set. If both are affirmative, the word is accessed out of the cache. Note that the path to memory can be wider than the path to the processor. The O offset bits are not used in the cache controller but are used externally to select the word from the block. If a block is to be read into the cache using multiple reads, this is controlled by the BMAC.

The basic algorithm to see if a word is present in the cache is (see figure 1):

1. Compare the sector field against the CAM. If there is a match go on. If not, a miss has been recorded and go to step 3.
2. Each sub-cache has a 2^B x 1 RAM associated with it in order to see if the desired block (word) is present, i.e., each block has a presence bit associated with it. Use B to access this RAM. If the presence bit is set, then the required data is present and the O offset bits can be used to access it. If the presence bit is not set, then a memory access or memory accesses load the data into the cache setting the relevant RAM bit.
3. This step is taken only if the address space denoted by the sector has no sub-cache allocated to it. Therefore, a sub-cache needs to be allocated. The oldest allocated sub-cache is freed, i.e., a sector sub-space is de-allocated and the needed sector is allocated to the sub-cache. All the presence bits of the sub-cache are reset to zero and step 2 is implemented. In this case a new mapping has to be made between the sub-caches and the addresses. Note that steps 1 and 2 can be taken in parallel, with the output of step 1 being used to validate the output of step 2.

Note that this replacement algorithm is essentially a FIFO mechanism. Simulations have shown that, with a working set the size of the BMAC's, there is no real performance difference between a FIFO algorithm and other replacement algorithms.

VIRTUAL MEMORY

The view of virtual memory presented here is supported by a mixture of hardware and software. It may well appear to be too complex for simple implementations, or not complex enough for high security systems. In the former case, the system can be simplified. In the latter case, the hardware protection can be extended by software to increase the degree by subtlet or range of the protection

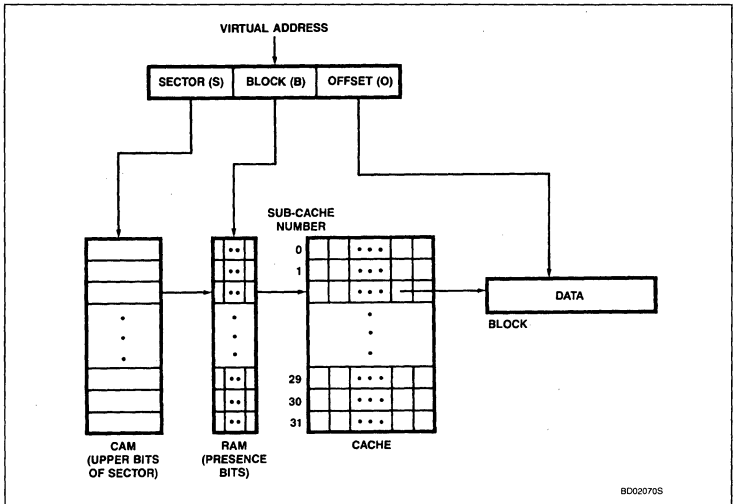


Figure 1. Addressing the Cache

mechanisms. These software extensions are protected by hardware so that they cannot be circumvented, i.e., a virtual address can be split into three portions.

1. Segment number
2. Page number
3. Offset within a page

This split is supported by hardware. The split between 1 and 2 is a convention of the software established at SYSGEN time, and the BMAC can be programmed accordingly. The choice of the split between 2 and 3 is more restricted and is also handled by the BMAC.

Partitioning of the Address Space

The virtual address space, as seen by the processor and the software, can be partitioned into up to four different regions each of which has its own segment table (first-level table). This partitioning need not occur, i.e., the address space can be left as one region. These regions which are configured by software are provided for a fast and protected operating system.

Virtual Address Space Regions

Four regions are defined sequentially starting from the bottom end of the virtual address space. The upper limit of each region is specified in turn. This implicitly defines the lower limit of the next region. Thus if only one region is to be specified, then its upper limit is given as FFFFFF. The four regions are all equivalent.

Note that the virtual address space can be defined by the address only. The function codes are then used for protection. The concept of address spaces in the 68451 MMU is replaced here by the concepts of the regions. The latter are used in a similar but less restricted fashion than the former to enable different processes each to have its own unique virtual address space. Naturally portions of this address space can be shared with other processes.

Each region on the MAC has its own segment table. Associated with each segment table is a region descriptor. The format for the region descriptor is in figure 2 and its use is shown in figure 3.

Note that the region descriptor refers to the segment table as its first-level table. The segment table defines which areas of the virtual address space are accessible to the current process. The contents of a region are flushed from the cache by writing into the appropriate flush region register. This can only be done in the supervisor mode. Region definition is done via the RDR register.

Use of the Regions in a Simple System

A simple system is defined as having only one region which is accessible by all function codes. Thus protection is performed by the protection bits associated with each segment or page. This system is configured by writing FFFFFF into all region definition registers, thus defining the upper boundary of region 0 (R0) to be FFFFFF and the lower boundary of R0 to be 000000. The system can be used as if it had only one region which stretches uniformly from 0 to FFFFFF.

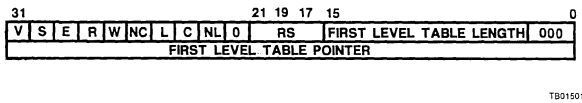


Figure 2. Region Descriptor

Full Use of the Regions

This section gives an example of how all four regions can be used in the implementation of an efficient operating system which protects users from one another and from itself. Region R0 is used only by the privileged part of the operating system and contains the part of the kernel which is privileged. Thus, all segments in this region are marked with an indication that supervisor permission is needed to access them. This part of the operating system should be as small as possible, because it has access to all of memory.

The next region, R1, is used by the supervisor. This is the part of the operating system that provides the non-privileged functions. This code is nevertheless 'trusted code' and can have access to resources that are not available to user programs. On a system call, the handling routine in R0, not only switches the call to the relevant routine in R1 but also disables the privileged state and enables R1.

There is often a large body of utilities in a system which is used by most programs. If the data which they access is protected, then there is no reason why these utilities cannot automatically be made available to all users in an execute only mode. Thus, R2 is used as a utility region available to all processes and always enabled.

The last region, R3, is the user region. This region contains the process specific code and data. The mapping of this virtual address space into an associated physical address differs from process to process. This remapping is performed by changing registers in the BMAC.

Segment Tables

At any instance, there are 1 to 4 valid segment tables (denoted as first-level tables in the region descriptor) depending on how many regions have been defined and enabled. The segment table consists of descriptors, each with three fields.

Page table pointer (next table pointer) – This 32-bit field contains the pointer to the page table which defines the current segment. Page tables must be aligned on long-word boundaries in memory.

Page table length (next table length) – This 22-bit field contains the page number of the last page in the segment.

Segment protection – This 10-bit field contains the bits which define the status of the segment. Each use of the segment is protected differently depending on the permissions granted to each process. These bits are:

1. Valid bit (V). This bit is used to indicate whether the segment is valid or not.
2. Supervisor permission (S). If this bit is set, supervisor permission is necessary to access this segment.
3. Access permission bits (R, W, E). If these bits are set, corresponding access is available for the segment: read access (R), write access (W), execute access (E).
4. The following three bits define the mapping of the segment:
   a. Non-Cacheable (NC). If this bit is set, the associated segment will not be brought into the cache memory. For example, if a memory is used concurrently by the processors or memory addresses are used for I/O, this segment should be made non-cacheable.
   b. Local (L). If this bit is set, pages of the associated segment may belong to the local memory attached to the processor. It enables the use of local memory identification (LI) field.
   c. Contiguous (C). If this bit is set, the associated segment does not have a page table. It is segment-only and should be mapped into memory as a whole. Here the base address is given in units of 1K bytes.
5. Non-leaf (NL). NL is not used by the BMAC and must be set to zero.
6. Offset bit (O). This bit is used to denote whether pages of a segment start at the top (higher address, O = 1) or bottom (lower address, O = 0) of the segment's virtual address space.

For a contiguous segment configuration, additional bits are defined in the second word. These bits are defined for the MAC and should be set to 000000 for the BMAC to maintain compatibility.

Figure 4 shows the format of the segment descriptors in the segment table for a paged (non-contiguous) segment and figure 5 shows the format of the contiguous segment descriptors.

Page Tables

The page table (or next-level table) is split into two fields totalling four bytes which are always used by the hardware. They contain:

1. The 18-bits pointer to the physical address of the beginning of the page given in units of 1K bytes.
2. Three page-history bits.
   – Present (P): page is present in memory.
   – Used (U): page is used since this bit is last reset.
   – Dirty (D): page is written to since this bit is last reset.
3. Two or six reserved bits (Rs). These bits may not be used by software for any purpose.
4. Local memory identification (LI). The BMAC supports systems with up to 15 local memories. If a segment is marked local, those bits indicate in which local memory the page is resident. This will be checked against the contents of the processor identification register. This check should be done by the BERR routine which loads the descriptor into the SCC68905. If LI = 1111, then the page is considered to reside in system memory.
5. A non-cacheable (NC) attribute bit. If this bit is set, the associated segment will not be brought into the cache memory. For example, if a memory is used concurrently by the processors or memory addresses are used for I/O, this segment should be made non-cacheable.
6. Page protection bits:
   – Supervisor permission (S) needed to access this page
   – Three access permission bits. Read access (R), write access (W), execute access (E). The three access permissions bits allow the access mode if set.

Figure 6 shows the format of the page descriptors in the page table in memory.

Residence of System Tables

It may be desirable to place most of the system tables mentioned above in virtual memory without necessarily having them in physical memory all the time.

Memory Protection and Sharing

The BMAC provides mechanisms for implementing systems with varying degrees of protection. The simplest protected system would only separate user code from operating system code. Protection is basically provided only by giving a process access to the resources it needs. These resources include memory and I/O. Access to I/O is accomplished by giving a process access to the necessary I/O management routines. This access can be execute-only and the entrance to these routines can be controlled via hardware in the normal system call manner or by putting a software 'gateway' as the access

point. This gateway is essentially an execute-only segment and/or page which can be marked so as to cause a trap when entered. Thus, a trusted program can be used to do the I/O access or to provide extra entry points to the calling program. For example, if a file is opened then this technique can add segments to the calling process's virtual address space, where each segment performs an I/O function. The process will only have access to these routines.

This process will only be effective if a process is forbidden to access its segment and/or page descriptors. This is achieved by not putting the segment and/or page tables belonging to a process in a segment and/or page directly accessible to the process. This segment and/or page is accessible to the operating system.

Sharing of code and data can be accomplished by merely putting a segment and/or page descriptor pointing to the same page table in more than one process's segment and/or page table. Note that these segment and/or pages can have different virtual addresses for the different process. Message passing can then be easily and efficiently implemented by asking the operating system to 'add' a descriptor to another process's address space.

Sub-Setting Facilities for Simple Systems

The BMAC can be used as an MMU either with or without using the cache controller function. If a simple system is desired, a subset of the BMAC's capabilities can be used. The following are some examples of possible system configurations:

1. A purely paged system
2. A minimally protected system. This would have one region with two segments for code and data, with no paging within each segment.
3. A minimally protected stack-based system. This would have one more segment for each stack used to be used to easily allocate stack space as it is needed, while still controlling unlimited growth.

These examples are provided to emphasize that all the facilities need not always be used.

REGISTERS

The registers on the BMAC are used for three purposes.

1. By the processor to inform the BMAC of the hardware system configuration.
2. To force the BMAC to invalidate the cache descriptor entries.
3. By the BMAC to inform the processor of its status.

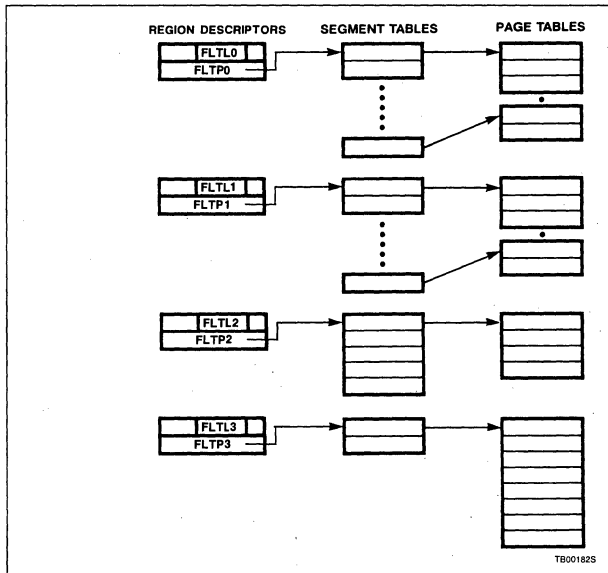


Figure 3. Table Structure

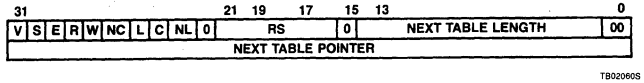


Figure 4. Segment Descriptor for a Paged Segment (C = 0)

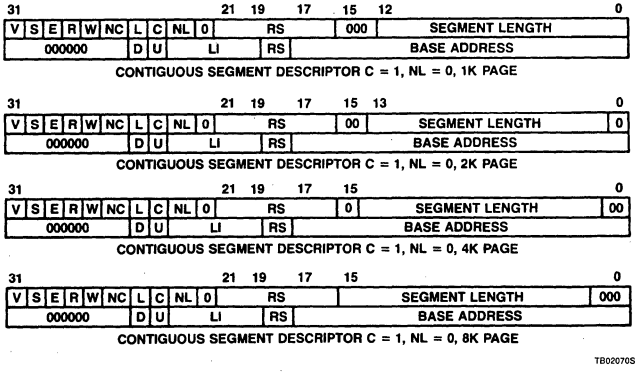


Figure 5. Segment Descriptors For A Contiguous Segment

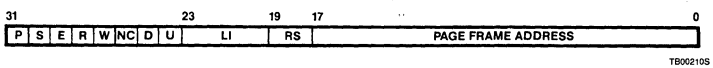


Figure 6. Page Descriptor For A Paged Segment

The BMAC will issue a BERR if chip select is asserted without using a supervisor function code. Registers on the BMAC can be accessed by the processor when the chip select input is active and address line A9 = 0. The register addresses are then decoded from address inputs A6–1.

Page Definition Register (PDR)

Only bits 0 – 1 of this register are used. These bits indicate the size of the page. Note that depending on the page size, some physical address pins of the BMAC will always be in tri-state mode. The pins that are tri-stated are listed below:

Bits 1 0	Page Size (Bytes)	Tri-stated Lines
0 0	1024	–
0 1	2048	PA10
1 0	4096	PA10-11
1 1	8192	PA10-12

Size of Segment Register (SSR)

Only bits 0 – 3 are used. This register, which is reset to zero on a BMAC reset, defines the part of the address that selects a segment. The contents of the SSR is the encoded number of address bits (starting with A23) that are part of the segment number. The segment size can vary between 16k and 8M bytes in size.

If less than five address bits are used for the segment number, the split of the address space into regions is no longer supported. In those cases the whole address space will be treated as region 0.

Size of Cache Register (SCR)

Only bits 0 – 3 are used. This register specifies the cache organization. The meaning of the bits which are all reset to zero on a BMAC reset are:

Bits 3 – 0	Block Size	Cache Depth
0000	16 bits	512
0001	32 bits	512
0010	64 bits	512
0011	128 bits	512
0100	16 bits	1024
0101	32 bits	1024
0110	64 bits	1024
0111	128 bits	1024
1000	16 bits	2048
1001	32 bits	2048
1010	64 bits	2048
1011	128 bits	2048
11xx	Reserved	

Note that bits 2, 3 define the cache depth and bits 0, 1 define the cache block size.

Region Definition Register (RDR)

This register defines the boundaries of regions 0, 1 and 2 explicitly and region 3 implicitly. The register is split up into three fields which define the five most significant bits of the address for each region. These bits are then extended to a full address with 19 1's in the least significant bit positions. Bits 0 – 4 define the highest address in region 0. Region 1 is defined from the top of region 0 to the address specified by bits 5 – 9 of the RDR, extended with 19 1's. Similarly, region 2 is defined by the top of region 1 at its low end and bits 10 – 14 of RDR at its high end. Region 3 is defined from the top of region 2 to FFFFFF at its top end. Bit 15 is not used.

Processor Identification Register (PIR)

Only bits 0 – 3 are used. The number encoded in the four bits of this register identifies one of the 15 processors. Code 1111 in this register is reserved.

BMAC Mode Register (MMR)

Only bits 0 – 1 are used. On reset, the BMAC resets this register to zero. It has to be set if the BMAC is to be used. When the BMAC is disabled, the cache is effectively not present and all addresses are passed through the BMAC without any change. When the BMAC is enabled, the addresses are translated in the chosen manner and the cache is used in the configuration given in the SCR. If bit 0 is set, it indicates that the cache memory is enabled and if bit 1 is set, it enables the BMAC.

Note that loading this register will enable the BMAC. This register should only be loaded when all the other registers have been loaded.

MMR Bit 1	MMR Bit 0	FUNCTION
0	0	BMAC disabled. Physical address out equals virtual address in.
0	1	BMAC enabled for cache operation only.
1	0	BMAC enabled for MMU operation only.
1	1	BMAC enabled for MMU operation and cache operation.

Memory Fetch Register (MFR)

Only bits 0 – 1 are used. This register indicates how many memory accesses are needed when the BMAC loads a cache block.

Bit 1	Bit 0	No. Memory Accesses
0	0	1
0	1	2
1	0	4
1	1	8

The number of memory accesses also corresponds to the block size.

Flush Region Registers (FRR0 – 3)

The BMAC will respond with a DTACK when any of these registers is addressed. The FRR registers are 16 bits wide. When the FRRs are loaded, all the mappings in the BMAC in the relevant region are reset, i.e., the cache is flushed and all addresses in the region and all page descriptors in the BMAC in the region's address space are invalidated.

MAC Communication Area Pointer Register (MCAPRU, MCAPRL)

Logically this register is 32 bits; it can be accessed by the upper and lower word. The BMAC will respond with a DTACK when this register is addressed. On the MAC, this register contains the physical address of the MAC communication area in memory.

Error Code Register (ECR)

This register contains the reason for the bus error. The error code is set when BERROUT is asserted by the BMAC and when register DESCR1 is loaded. The meaning of the bits is:

bit 0, 1 Contain the region number where the error occurred.

bits 2 – 6 5-bit error code. The following error causes are encoded for the BMAC. All other codes are reserved for future use.

EC 2	3	4	5	6	
0	0	0	0	0	No error
0	0	0	0	1	Page not in required local memory
0	0	0	1	0	Access permission violation
0	1	0	0	0	Illegal BMAC access
0	1	1	1	1	BERR on global bus

An access permission violation occurs when the current bus cycle does not have the access permissions required for the segment or page that it attempts to access. The access permissions of a page are S (supervisor permission needed), E (execute access), R (read access) and W (write access). These four bits are stored for each page in the page translation look-aside buffer (TLB) on the BMAC and they represent the exclusive OR of the corresponding bits in the segment and the page tables.

The current bus cycle is characterized by the signals FC3 or (processor/I/O cycle), FC2 (supervisor/user mode), FC1 (program/data access) and R/W (read/write). Access per-

Table 1. ACCESS RIGHTS

FC3 0: I/O 1: PROC	FC2 USER SUPERV	FC1 DATA PROGRAM	FC0	R/W WRITE READ	S	E	R	W
1	0	0	X	0	0	X	X	1
1	0	0	X	1	0	X	1	X
1	0	1	X	0	N	N	N	N
1	0	1	X	1	0	1	X	X
1	1	0	X	0	X	X	X	1
1	1	0	X	1	X	X	1	X
1	1	1	0	0	N	N	N	N
1	1	1	0	1	X	1	X	X
1	1	1	1	X	X	X	X	X
0	X	X	X	X	X	X	X	X

X = do not care, N = violation, illegal input combination.

missions will not be checked during an I/O cycle (FC3 = 0). Table 1 shows the required access rights (S, E, R, W) for all possible non CPU-space bus cycles. All other combinations will result in an access permission violation.

bit 7 — Valid only when bit 15 is set. If ECR(15) = 1, this bit, if set, indicates a page descriptor miss; if not set, it means the dirty bit is not set on a write.

bits 8 – 10 — Not used, reserved.

bits 11 – 14 — S, E, R and W access permission bits of the currently accessed page. This field is valid only when an access permission violation is reported.

bit 15 — If set, it indicates a TLB miss; to be handled by the BMAC TLB loading software routine. If not set, an access error as specified by the error code has occurred.

Descriptor Address Register (DAR)

Only bits 4–0 are used. This 5-bit register contains the address of the next slot to be used in the TLB. This register must be updated before the next descriptor is loaded into the TLB (see Memory Mapping Unit).

Descriptor Registers (DESCR0 – 2)

These registers are used to load descriptors. The BMAC requires the most significant 16 bits of the first long word of the segment descriptor to be written to register DESCR0, that the most significant 16 bits of the page descriptor be written to register DESCR1, and that the least 16 significant bits of the page descriptor be written to register DESCR2. Registers DESCR0, DESCR1, and DESCR2 must be written in that order. It is recommended that the user use the MOVEM instruction to transfer the data to the registers.

Register Summary

Tables 2 and 3 describe the register map and bit map formats.

BMAC OPERATION

The BMAC can be thought of as two cooperating units; the cache controller section and the memory mapping unit (MMU). Between them, all of the BMAC functions are provided.

The Cache Controller

The cache controller manages a sector-associative cache. It has a working set of 32 sub-caches. These sub-caches are defined not only by the address, but also implicitly by the address space. The BMAC recognizes four address spaces defined by the regions. The region definitions are independent of their use in the system. System designers can use them for whatever purpose is required. However, the same virtual address cannot be used in different regions. The distinction between the regions, in the cache controller, is made purely to allow the cache entries belonging to the separate address spaces to be invalidated for a region as a whole. If a system's dynamic working set is favorable, this will allow interrupt service routines, for instance, to remain in the cache while the processor continues a process or even changes processes.

On a reset, the cache is marked as empty and all 32 sub-caches are available to all four regions. All sub-caches which are in use by a region are automatically marked as empty whenever that region's flush region register is loaded. Accesses with a CPU address space function code 111 are never cached; they are passed through to the physical address bus. Upon a CPU space access (FC2 – 0 = 111 and ASN are asserted), the BMAC will pass the virtual address on to the physical address pins and assert MAS. When the DTACKIN (or BERRIN) becomes active, the BMAC will assert DTACKOUT (or BERROUT). The cycle will be finished when ASN is negated by the processor.

A word can only get into the cache if it has satisfied the permission criteria of the associated descriptors. Thus, when a word is read and it is found in the cache, it is valid if the processor has the necessary permissions. A copy of these permission rights is kept with the associated sub-cache. If a write is performed, then this is only written into the cache if the associated descriptor has write permission. If it is so desired, the cache can be automatically disabled on reset. It can be enabled by writing into the MAC mode register.

Table 2. BMAC REGISTER MAP

ADDRESS[1] 6 5 4 3 2 1	ACRONYM	REGISTER NAME	MODE	AFFECTED BY RESET	NOTES
0 0 0 0 0 0	PDR	Page definition register	R/W	Yes	
0 0 0 0 0 1	SSR	Size of segment register	R/W	Yes	
0 0 0 0 1 0	SCR	Size of cache register	R/W	Yes	
0 0 0 0 1 1	Reserved				
0 0 0 1 0 0	ECR	Error code register	R	Yes	
0 0 0 1 0 1	PIR	Processor indentification register	R/W	Yes	
0 0 0 1 1 0	MMR	MAC mode register	R/W	Yes	
0 0 0 1 1 1	MFR	Memory fetch register	R/W	Yes	
0 0 1 0 0 0	Reserved				
0 0 1 0 0 1	Reserved				
0 0 1 0 1 0	DESCR0	Descriptor register 0	W	No	
0 0 1 0 1 1	Reserved				
0 0 1 1 0 0	DESCR1	Descriptor register 1	W	No	
0 0 1 1 0 1	DESCR2	Descriptor register 2	W	No	
0 0 1 1 1 0	DAR	Descriptor address register	R/W	Yes	
0 0 1 1 1 1	Reserved				
0 1 0 0 0 0	Reserved				
0 1 0 0 0 1	FRR0	Flush region register 0	R/W	No	
0 1 0 0 1 0	Reserved				
0 1 0 0 1 1	FRR1	Flush region register 1	R/W	No	
0 1 0 1 0 0	Reserved				
0 1 0 1 0 1	FRR2	Flush region register 2	R/W	No	
0 1 0 1 1 0	Reserved				
0 1 0 1 1 1	FRR3	Flush region register 3	R/W	No	
0 1 1 0 0 0	Reserved				
0 1 1 0 0 1	RDR	Region definition register	R/W	Yes	
0 1 1 0 1 0	Reserved				
0 1 1 0 1 1	Reserved				
0 1 1 1 0 0	Reserved				
0 1 1 1 0 1	Reserved				
0 1 1 1 1 0	Reserved				
0 1 1 1 1 1	Reserved				
1 0 0 0 0 0	Reserved				
0 1 0 0 0 1	Reserved				
1 0 0 0 1 0	MCARU	MAC communication area pointer upper	R/W	No	2
1 0 0 0 1 1	MCARL	MAC communication area pointer lower	R/W	No	2
1 0 0 1 0 0	Reserved				
•					
•					
•					
1 1 1 1 1 1					

NOTES:
1. All registers are accessible as 16 word bits only. Attempt to access these registers as one 8-bit byte will result in an illegal access error.
2. These registers maintain compatibility with the MAC. Access to these registers in the BMAC will be responded to only with DTACK. No other actions will take place.

If a read miss occurs, the BMAC will fetch enough words to fill the associated cache block. It does so with the aid of an external counter (74F193), and is connected as shown in figure 7 with associated timing in figure 8.

The miss signal enable (MISS) is normally low and LACLR is also low so that the signals A3–1 pass through the 74F193. If a block is to be loaded, LACLR is raised high for one clock period so that the counter is cleared. MISS is also raised high for the duration of the load. Next, MASN is asserted and the first word is accessed. After each DTACKINN, the MAC will dis-assert MASN for a 1-1/2 clock period and reissue MASN as many times as specified in the MFR register. As the high going edge of MASN will increment the counter, the correct word address within the block will be produced. At the end of the block load, the BMAC lowers MISS so that the correct low order address is fed to the cache before asserting DTACKOUTN to the processor. After MISS gets dis-asserted, CCEN goes low so that the word or byte initially desired be read from the cache.

Note that if the SCC68172 BUSCON is used in the system and sequential accesses are necessary, then LACLR can be used to set an external 74F74 to indicate that a sequential access is being made. MISS going low at the end of the sequence will reset the flip-flop.

Write Through

When a write is accomplished, the information is written to memory. Information is also written to the cache only when a sub-cache is already allocated for this data and the relevant block presence bit is marked as valid.

Read/Write Overlap

On a write, the BMAC will strobe the address and data into external latches and start a memory write cycle. Before doing so, it will check if it has the required descriptor; if not, it will assert BERR. If it is not already set, the

Table 3. BMAC BIT REGISTER MAP[1]

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PDR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	pg size	
SSR	*	*	*	*	*	*	*	*	*	*	*	*	segment size			
SCR	*	*	*	*	*	*	*	*	*	*	*	*	cache size			
ECR	T	S	E	R	W	*	*	*	P/D	EC4	EC3	EC2	EC1	EC0	RN1	RN0
PIR	*	*	*	*	*	*	*	*	*	*	*	*	processor no.			
MMR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	mmu	cach
MFR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	count no.	
DESCR0	V	S	E	R	W	NC	NL	C	NL	O	*	*	*	*	*	*
DESCR1 (C = 0)	P	PS	PE	PR	PW	NC	D	U	LI3	LI2	LI1	LI0	*	*	*	*
DESCR1 (C = 1)	*	*	*	*	*	*	D	U	LI3	LI2	LI1	LI0	*	*	*	*
DESCR2 (PA)	*	*	23	22	21	20	19	18	17	16	15	14	13	12	11	10
DAR	*	*	*	*	*	*	*	*	*	*	*	TLB address				
FRR0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FRR1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FRR2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FRR3	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
RDR	*	region 2 top msbs					region 1 top msbs					region 0 top msbs				
STLR0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STLR1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STLR2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STLR3	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
MCAPRU	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
MCAPRL	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

NOTE:
* Indicates null bits. All null bits will be read as zero.

BMAC will also set the dirty bit in the descriptor within the translation look-aside buffer on the BMAC and assert BERR to the processor. If the required descriptor is in the BMAC, then it will allow the processor to proceed by asserting the DTACK signal while the memory transfer is taking place. The processor will proceed unless it requests another memory transfer (either a cache read miss or a write) before the first write is complete.

Size of the Cache

The size of the cache is not infinitely variable, but can be expanded in both width and depth. The width can be expanded in units of 16 bits. The block can be configured to be 1, 2, 4, or 8 units wide, i.e., 16, 32, 64 or 128 bits wide. This width, the block size, is independent of the width of the data path to the processor. The number of memory accesses needed to fill a block is given by the memory fetch register.

The length of the cache can also vary and, again, the lengths to which it can be extended are limited. These limits are 512, 1024, or 2048 blocks deep. In terms of bytes the cache can then vary from a minimum of 1K bytes (512 blocks of 2 bytes) to 32K bytes (2K blocks of 16 bytes).

Although the actual number is application dependent, in general, a cache of 2048 blocks of four words, i.e., 2Kx64, will result in the processor seeing a cache access time for at least 95% of the memory accesses. For example, if a cache hit or an overlappped write occur fast enough not to cause any wait states in the processor and a memory access results in four wait states, then the average number of wait states per memory access is 0.2. These figures are realistic for a 16MHz 68010/20 coupled to a slow, 250ns access memory.

Memory Mapping Unit

The memory mapping unit (MMU) performs paging control between three levels of memory. These three levels are:

1. Backing store (Disk)
2. System memory (RAM)
3. Local memory (RAM)

Local memory is not always present. In a multiprocessor situation, it may be desirable in order to reduce bus traffic, to attach a local memory to each or some of the processors. If so, the BMAC is capable of providing signals to the processor if the segment is marked as belonging to local memory. This page fault trap will be generated if the accessed page is not present in the local memory. Thus the BMAC cannot only control paging between the system memory and the backing store, but also between local memories and the system memory or backing store.

The memory mapping unit section of the BMAC contains descriptors for 32 pages which are loaded on demand by the BMAC. As in the cache, each page is identified only by its logical address. In addition, the MMU uses four region descriptors. Each of these descriptors defines the current context of its respective region. The region definition register is used to define region boundaries. The page descriptors are in a fully associative

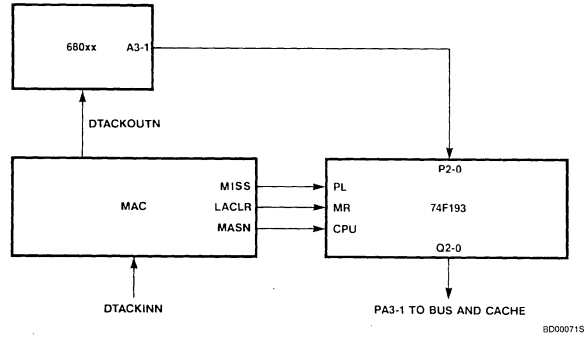


Figure 7. Cache Memory Block Transfer Circuitry

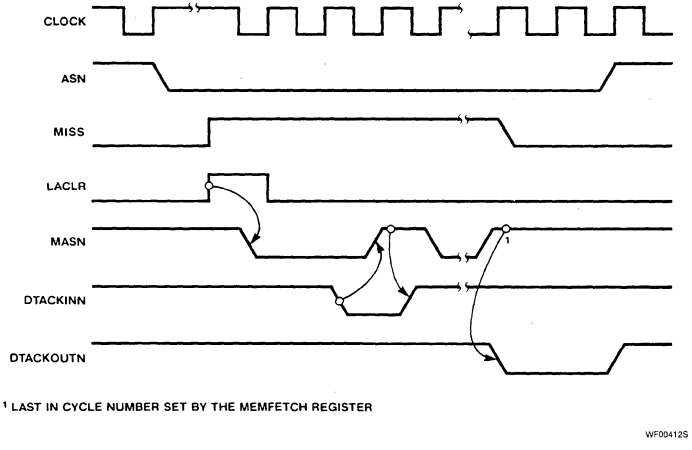


Figure 8. Memory Fetch Cycle Timing

TLB on the BMAC. Each descriptor consists of the following fields:

1. An 18-bit physical address
2. Four permission bits
   - Supervisor permission needed (S)
   - Access permission bits:
   - Read access (R)
   - Write access (W)
   - Execute access (E)
3. Page management bits
   - Non-cacheable (NC)
   - Local (L)
4. A dirty bit (D), i.e., the page has been written into.

The processor must write the descriptor to be used into the descriptor registers. This is done by loading the first word of the relevant segment table descriptor into register DESCR0, and the two words of the relevant page table descriptor into register DESCR1 and register DESCR2, respectively. While the three registers DESCR0 – 2 are loaded, the BMAC will check whether a page that is marked local is present in the required local memory. This is done only when the local bit in the segment table is set, by comparing the LI field in the page table with the contents of the PIR register. When the L bit is not set, or the LI field equals 1111, this check is disabled. The result of this check is stored in the ECR register. When the result is negative, i.e., the page is not in the required local memory, the BMAC will not load the page descriptor into its cache.

There is space for a cache of 32 descriptors on the BMAC. Before a new descriptor is loaded, the corresponding TLB slot number must be loaded into the DAR.

After the BMAC is reset, the DAR points to slot 0. The first several slots should be used for the descriptors which contain the system stack, BERR handling code, and data. These 'fixed' descriptors need to be resident before the virtual address system is enabled. It should be noted that regions containing these 'fixed' descriptors must not be flushed.

Translation Look-Aside Buffer Miss
If a word is read and it is not in the cache, the MMU section is also accessed to find its physical address. If the relevant page descriptor is not in the TLB, then the BMAC will assert BERR.

Handling of the Dirty Bit
If any permitted write is made to a page and the dirty bit in the page descriptor in the TLB is not set, then it is set in the TLB and a BERR is generated.

Handling of Illegal Accesses to Memory
If an access is made to a memory location for which the required permission bits are not set, then the BERR is set. The reason is stored in the BMAC, in the error code register. If an access to a non-resident page is made, the same procedure will be followed. This will be done whether the page at fault was the desired page, or a page containing either the segment table or the page table.

Error Reporting
When an error or a page fault is detected by the BMAC during its operation, an error-report containing a description of the problem is stored in the ECR by the BMAC. The BMAC then generates a BERR to the processor.

Cache Only Mode

In cache only mode, the BMAC does no address translation or permission testing. Therefore, the physical address output is not useful since it has no relation to the input logical address. Regions can be used to provide a basis for selective flushing of the cache, where the applied address can either be real or virtual (depending on the application). Accesses with a CPU address space function code (FC2 – 0 = 111) are never cached; virtual addresses are passed through to the physical address bus.

THE BMAC'S PLACE IN THE SYSTEM

It is possible to have one or more BMACs in the system. A system with one processor with an attached BMAC is shown in figure 9. The BMAC handles all the processor requests to memory and completely controls the cache. This is a fairly straight forward solution, as all the DMA requests use real addresses. While this is fairly straight forward for the hardware, the software has to arrange the blocks that have to be DMA'ed into memory to fit into consecutive real addresses, unless the DMA units can handle a scattered real memory.

Virtual I/O Addresses

This system is shown in figure 10. It is much easier for the software to handle, but the address bus onto which the DMA peripherals hang is the bus between the processor and the BMAC (note that all relevant I/O descriptors must be in the BMAC before needed). This is fine for a system with limited modularity, but it is difficult for a system which needs the traditional modularity, e.g., on the card level of a minicomputer. If that type of system is needed, then the configuration given in figure 11 can be used, i.e., with a separate BMAC dedicated to I/O. Here all descriptors are loaded before the start of an I/O action.

I/O units using a BMAC for address translation, must drive the FC3 signal low. The BMAC will then skip the check on access violations. FC2 – 0 can be used to identify each of up to seven I/O units connected to the MAC (note that FC2 – 0 = 111 is reserved and may not be used for I/O identification).

Multiple MACs Attached to Multiple Processors

This can be accomplished in two ways depending on the system needs. A typical system without local memory is shown in figure 12. This sort of system will allow up to four 68010s to work without seriously disturbing one another. If they work in a MIMD (multi-instruction stream, multi-data stream) mode, i.e., each working on its own process but only one running each section of the operating system at a time. Then typically four 68010s should give the performance of about 3.2 68010s, each running in their own system. This assumes that the processors are memory bound and not I/O bound. Note that although figure 10 shows processors each with their own MAC, these processors could just as easily be I/O units with DMA properties.

Systems with Local Memory

A typical system is shown in figure 13. Systems like this can be used if there is a way to partition the application. This can be done in many ways depending on the complexity of the system and the software. Some applications can be partitioned statically, e.g., a process control system where the tasks can be pre-assigned to a processor. In this case, the local memory can be used for code and process specific data, e.g., stacks.

Of course, applications that are reassigned dynamically can also be made using this system arrangement and the BMAC provides facilities to page in and out of local memories. When the page descriptor associated with a local page or segment (i.e., L bit is set) is loaded into the BMAC, an error status is stored in the ECR register if the LI field does not match the BMAC's PIR register contents. This check is overridden if the LI field is 1111.

The BMAC allows up to fifteen 680xxs, each having its own local memory to cooperate together. In particular, with the requisite system software, it is possible to write application programs that could run unchanged in either a single processor system or a multiprocessor with or without local memory.

The Use of Multiple BMACs Attached To One Processor

In some special cases, there may be a need to attach more than one BMAC to a processor. Normally, this should not be necessary as the cache hit rate should be high enough (greater than 90%) and the TLB hit rate should be better than 99.5%. However, if this is not enough or if a fast response is wanted for interrupts then multiple BMACs can be used. The BMAC will keep all of its output lines in a high impedance state when ASN is high. Therefore, multiple BMACs can be used by gating ASN with another address line(s). Note this address line should not be part of the offset within a page as that will reduce the effect of adding more than one BMAC.

If multiple BMACs are attached to one processor, then all the BMACs have to have identical region descriptor definitions.

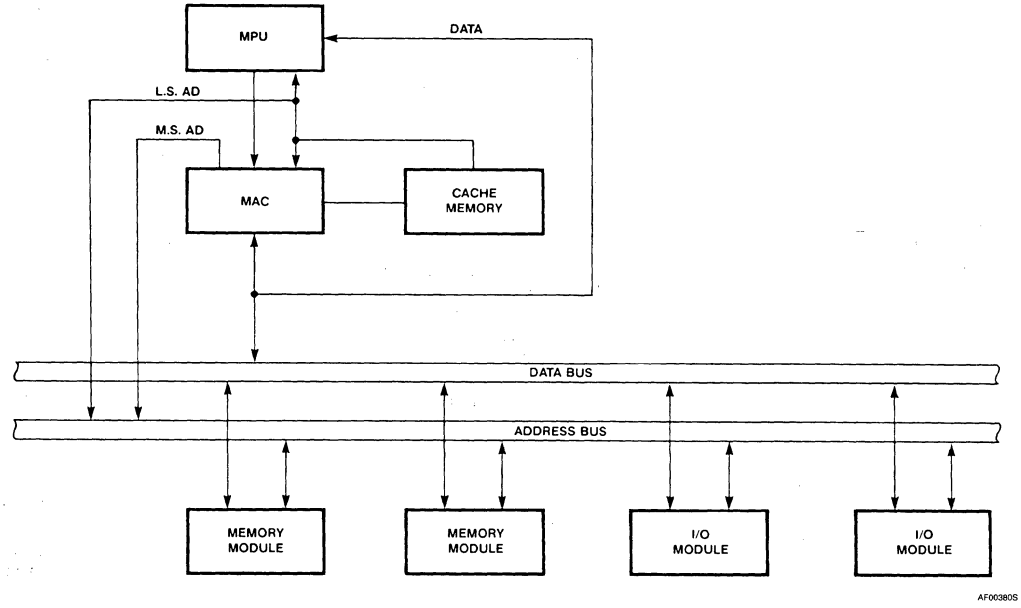


Figure 9. Processor With BMAC, Real I/O

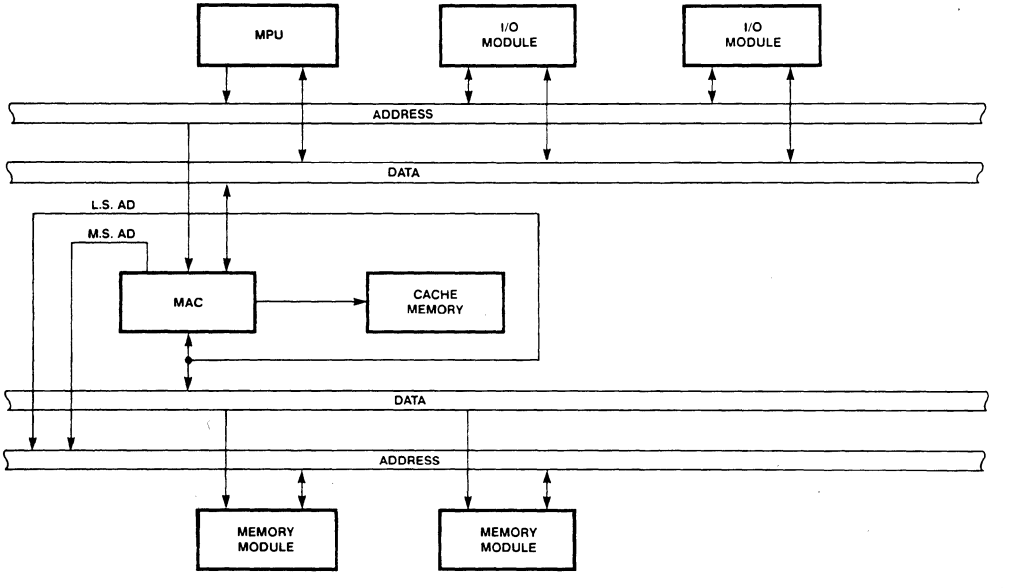


Figure 10. Virtual I/O

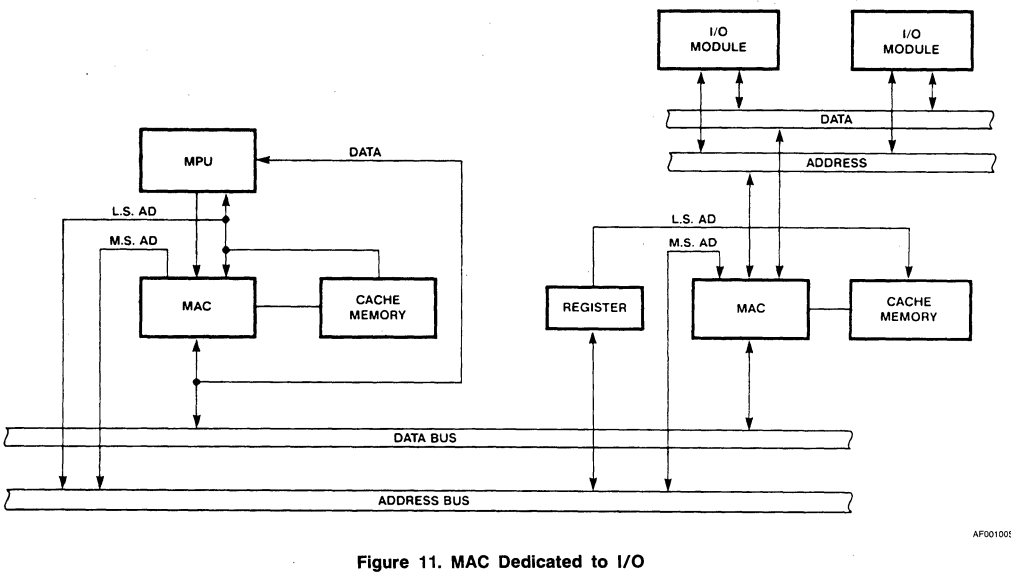


Figure 11. MAC Dedicated to I/O

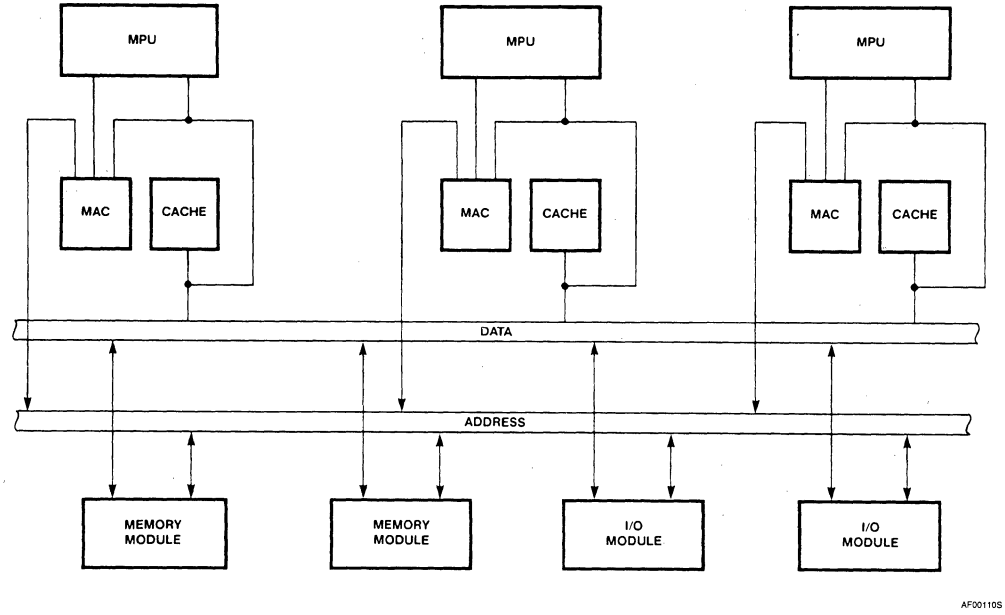


Figure 12. Multiple Processors, No Local Memory

PROGRAMMING THE BMAC

On power-up all the registers on the BMAC are reset to zero. This implies that there are no caches attached and all addresses are real, i.e., the BMAC does no translation whatsoever. The BMAC will also be reset by the RESET signal.

Initializing the System

The following registers have to be set.

1. The page definition register
2. The size of segment register
3. The region definition register
4. The BMAC communication area pointer register
5. The size of cache register
6. The memory fetch register
7. The BMAC mode register

The order of setting is irrelevant, except that as the BMAC mode register activates the system it should be set last. In addition the 'fixed' descriptors should be loaded.

Programming on a Context Change

As the BMAC needs only to be given a process context, it does not have to be reloaded on a process change. The only actions necessary are to invalidate the old context. This is done by writing to the relevant flush region register FRRn. This action only invalidates the cache and TLB so as not to confuse the old context with the new context.

Programming for Change of System State

Care should be taken when this is done. In particular, if there are multiple units accessing memory, i.e., processors or I/O units, they should be in a known state before this is done and should be suitably informed when the system state has changed. In other words, it is assumed that the system is normally only the kernel of the operating system and is normally fixed in memory.

This possibility for reconfiguration is only provided for safety and to allow the system to be reconfigured on a module failure.

Programming for I/O

Two philosophies are possible:

1. Real I/O
2. Virtual I/O

In real I/O, all I/O is done in real address space and the BMAC is not involved with the I/O operation. In virtual I/O, system I/O is mapped through one or more BMACs. If BMACs are shared between DMA control units then, seven function codes 0xxx should be shared, one to each of up to seven peripherals. Note that function code 0000 is reserved for the processor and cannot be used for I/O devices. Also note that FC3 is low for I/O transfers. The required descriptors should be in the BMAC. If the associated peripheral controller does not contain a deep FIFO, then required pages and tables should be locked in memory.

If required descriptors are not initially in the BMAC, then care must be taken to ensure their presence before they are needed.

TIMING

Memory accesses initiated by the BMAC will use the same timing as the 68010, i.e., three cycles for read and four for write. The internal timing of the BMAC will be such that when a series of reads have to be done, assuming zero wait states from the memory, then a read will be issued every four clock cycles. Some exceptions to this requirement may be made.

The CWEN signal will have a positive set-up and hold time with respect to CCEN.

ADDRESS FORMATS AND MEMORY ORGANIZATION SUMMARY

Table 4 summarizes the address and data path widths for the BMAC.

The logical address is composed of three fields: segment number, page number and offset within the page. The sizes of these fields are programmable in the BMAC via the

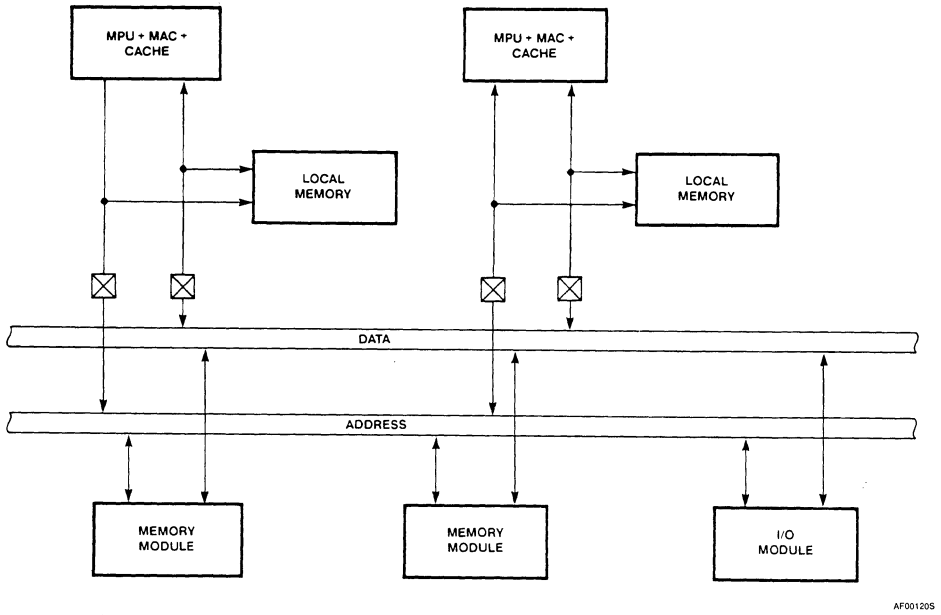


Figure 13. System With Local Memory

page definition register. The possible address compositions are shown in table 5.

BMAC BERR HANDLER PROCESSING

Refer to figure 14 which covers all of the possible BERR handling features. The BERR handler routine first saves any required 68010 registers. The contents of the error code register are then read. When bit 15 = 0, an access/permission/true BERR has occurred. Processing then branches to entry E1, where bits 2 – 6 of the ECR are checked for the type of violation. Access violations are further identified via bits 11 – 14 which give the TLB descriptor values for permissions S,E,R, and W. A jump table contains the entry points for the various violations. These violations are listed in table 6.

Table 4. ADDRESS AND DATA PATH WIDTH

VERSION	NUMBER OF BITS		
	Logical ADDR	Physical ADDR	Data
84-Pin	24	24*	16

*If no local memories are used, the physical address may consist of 25 bits.

24-BIT LOGICAL ADDRESS

23	13	12	10	9	0
S/P -	S/P	P/0 - - - -	P/0	0 - - - - - - -	0

The main processing of the BMAC BERR handler occurs when ECR bit 15=1, indicating a descriptor fault. The routine then accesses the 68010 exception stack to obtain three words; the fault address (high/low) and the special status word (see table 7). Note that the routine may have to move this information from the stack in three 16-bit word moves, foregoing the more efficient long-word moves to avoid possible addressing (boundary) problems. The fault address words provide the virtual address of the BERR. These words must be saved in a 68010 data register. This data represents the segment number, page number, and offset within the page of the virtual address. The exact locations of these portions are determined by the user during software set-up.

Bit 8 of the special status word, figure 15, indicates if a read/write was being performed when the BERR occurred. Bits 0 – 1 of the

Table 5. MEMORY ORGANIZATION

SEGMENT SIZE (BYTES)	NUMBER OF SEGMENTS	MAX NUMBER OF PAGES PER SEGMENT			
		Page Size 1K Bytes	Page Size 2K Bytes	Page Size 4K Bytes	Page Size 8K Bytes
8M	2	8K	4K	2K	1K
4M	4	4K	2K	1K	512
2M	8	2K	1K	512	256
1M	16	1K	512	256	128
512K	32	512	256	128	64
256K	64	256	128	64	32
128K	128	128	64	32	16
64K	256	64	32	16	8
32K	512	32	16	8	4
16K	1K	16	8	4	2

Table 6. BMAC BERR HANDLER SUBROUTINES FOR VIOLATIONS (E1 ENTRY POINTS)

BITS 6–2	BITS 14–11	SUBROUTINE ENTRY POINT
00000	NA	No error
10000	NA	Page not in required local memory
01000	S,E,R,W	Access permission violation: (Supervisor, Execute, Read, Write)
00010	NA	Illegal BMAC access
11110	NA	BERR on local bus

Table 7. EXCEPTION STACK ORDER FOR 68010 (BUS AND ADDRESS ERROR)

15 0

Program counter (high)	
Program counter (low)	
1000	Vector offset
Special status word	
Fault address (high)	
Fault address (low)	
Unused, reserved	
Data output buffer	
Unused, reserved	
Data input buffer	
Unused, reserved	
Instruction input buffer	
Internal information	

ECR contain the region number where the error occurred. This region number is used to select the appropriate region descriptor from which are selected, in turn, the segment table pointer, (first-level table pointer, FLTP) and the segment table length (first-level table length, FLTL). Next, the routine will shift out the segment number from the virtual address.

Bit 7 of the ECR now determines the major processing decision of the routine. If bit 7 = 1, a page descriptor miss has occurred. A series of segment and page checks need to be made, and then the three BMAC descriptor registers will be loaded by the routine. When bit 7 = 0, the dirty bit has not been set on a write. The series of checking and descriptor loading will be bypassed, as they already have been performed. Table 8 gives the possible error processing done by the routine. Both branches refer to the segment descriptors and the page descriptors.

When bit 7 = 1, the segment number (which has been shifted out of the virtual address by the routine) is compared to the number of segments currently valid (segment table length register). If the segment number is greater than the FLTL, a 'segment no. out-of-range' error is in effect, and the routine will branch to the E2 error entry point (see table 6). Otherwise, the segment descriptor is accessed to test the various protection bits. The segment descriptor address will be built by the routine as follows: FLTL + segment number * 8.

If the segment is invalid (bit 31 = 1), processing branches to error entry point E3.

The offset (bit 22) bit denotes whether the pages of a segment start at the top (higher address, O = 1) or bottom (lower address, O not = 1) of the segment's virtual address space. The routine will use the page number (shifted out of the virtual address of the exception stack) and the page length (from the segment descriptor words). If O = 1, the routine will check for the page length being greater than the page number. If O is not = 1, the error checked will be for the page number being greater than the page length. Both error conditions will force a branch to error entry point E4.

Next, the routine will test the contiguous bit (bit 24) in the segment descriptor. If C = 1, the segment is contiguous, i.e., has no page table. The routine will then skip over the page checking code to go on to loading the segment data into the descriptor registers. If C is not = 1, then the routine will perform page processing as follows. It will shift out the page number from the virtual address and get the page length from the segment descriptor. Then, the routine will build the page descriptor address thus: base address from segment descriptor + page number * 4.

Next the page present bit (bit 31) will be tested. If P = 0, the page is not present in memory, and the routine will branch to error entry point E5 to process the error. Otherwise, the routine will proceed with remaining page processing. The access type (from special status word) will then be tested. If a write, the dirty bit and the used bit in the page descriptor will be set. If the BERR interrupted a read access, only the used bit will be set by the routine. Next, the routine will load the three descriptor registers, DESCR0–DESCR2, as follows: DESCR0 = first word of segment descriptor, DESCR1 = first word of page descriptor, and DESCR2 = second word of page descriptor. Two important rules must be followed here:

1. The registers must be loaded in the order of DESCR0, DESCR1, DESCR2.
2. The registers must be loaded atomically. Thus, the 68010 MOVEM instruction must be used.

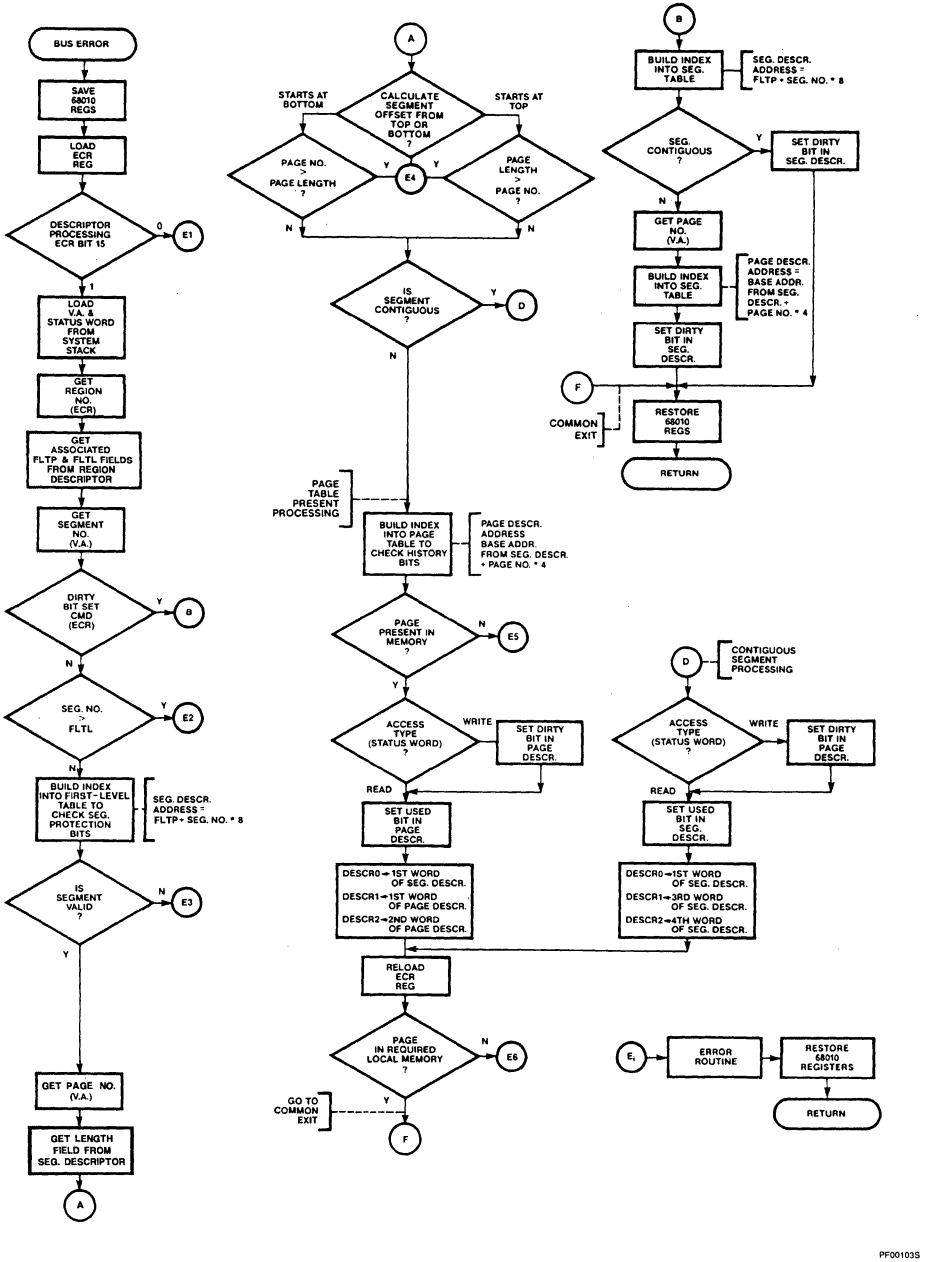
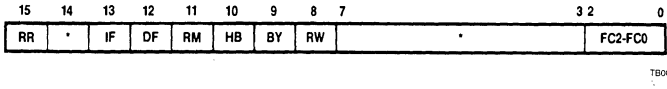


Figure 14. Bus Error (BERR) Handling



RR – Re-run flag; 0 = processor re-run (default), 1 = software re-run.
IF – Instruction fetch to the instruction input buffer.
DF – Data fetch to the data input buffer.
RM – Read-modify-write cycle.
HB – High byte transfer from the data output buffer or to the data input buffer.
BY – Byte transfer flag; HB selects the high or low byte of the transfer register. If BY is clear, the transfer is a word.
RW – Read/write flag; 0 = write, 1 = read.
FC – The function code used during the faulted access.
* These bits are reserved for future use and will be zero when written by the 68010.

Figure 15. Special Status Word Format

Table 8. ERROR ENTRY POINTS

ENTRY	SUBROUTINE ENTRY
E1	Access/other errors - 'good' violations (table 5)
E2	Segment number out of range i.e. greater than number of segments currently valid
E3	Segment not valid
E4	Page number out of range i.e., is in invalid address space of the segment
E5	Page is not present in memory
E6	Page is not in required local memory

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltage	−0.3 to +7.0	V
Input voltage[2]	−0.3 to +7.0	V
Operating temperature range[3]	0 to +70	°C
Storage temperature	55 to +150	°C

The routine will then reload the contents of the ECR. This is required as loading DESCR0 – DESCR2 causes the BMAC to check whether a page that is marked local is present in memory and to set bits 2 – 6 accordingly (bits 2 – 6 = 00001). If the bit has been set, the routine will branch to error entry point E6. Else, the routine will restore its 68010 registers and return.

In the case where the segment is contiguous (C = 1), the routine will set the dirty bit and used bits in the segment descriptor for a write access; the used bit will be set if the BERR occurred during a read access (access type being obtained from the special status word). The routine will then perform activities similar to those taken for loading page descriptors, except for the following differences. DESCR0 will be loaded with the first word of the segment descriptor, DESCR1 will be loaded with the third word of the segment descriptor, and DESCR2 will be loaded with the fourth word of the segment descriptor. The remaining processing is identical to that for paged segments.

The remaining paragraphs describe actions that will be performed when a dirty bit is not set on a write (ECR bit 7 = 0). The routine will build the segment descriptor address thus: FLTP + Segment number * 8. Next, if the segment is contiguous (C = 1, bit 24 of the segment descriptor), the routine will set the dirty bit (bit 25) in the segment descriptor, restore the 68010 registers and return. If the segment has page tables, the page number will be shifted out of the virtual address and will be used to build the page descriptor address, thus: base address from segment descriptor + page number * 4. The routine will then set the dirty bit (bit 25) in the page descriptor, restore the 68010 registers and return.

DC ELECTRICAL CHARACTERISTICS $V_{CC} = 5V \pm 5\%$, $V_{SS} = 0V$; $T_A = 0°C$ to $+70°C$[4,5]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IH}	Input high voltage		2.0	V_{CC}	V
V_{IL}	Input low voltage		GND – 0.75	0.8	V
I_{IN}	Input leakage current	5.25V		20	μA
I_{TSI}	Three-state (off state) input current	2.4V/0.4V		20	μA
V_{OH}	Output high voltage	$I_{OH} = -400\mu A$	2.4		V
V_{OL}	Output low voltage	$I_{OL} = 5.3mA$		0.5	V
P_D	Power dissipation			1.5	W
C_{IN}	Capacitance	$V_{in} = 0V$, $T_A = 25°C$, $f_o = 16MHz$		10	pF

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5]

NO.	FIGURE	PARAMETER	TENTATIVE LIMITS		UNIT
			Min	Max	
0		Clock period	80		μs
1	16	Delay of CS from AS		5	ns
2	16, 19, 20, 24	Synchronous output delay from clock edge		40	ns
3	16, 18, 19	Output delay from negation of AS/DS		80	ns
4	17	Read register access time from assertion of CS	185		ns
5	17, 18, 19	Data bus float time from negation of AS/DS	10	100	ns
6	18	DTACKOUT delay from assertion of AS		45	ns
7	18	CCE delay from assertion of AS		65	ns
8	18	Set-up time of CADDO-4 before CCE	0		ns
9	18	Hold time of CADDO-4 after AS/DS	10		ns
10	18	Cache memory access time		55	ns
11	19	PA set-up before assertion of MAS	5		ns
11A	19	PA delay from assertion of VA		100	ns
12	19	PA hold time after negation of MAS	10		ns
13	19	DTACKIN delay from assertion of MAS	2 clk per		
14	19	DTACKIN delay from negation of MAS		1 clk per	
15	19	Asynchronous DTACKOUT delay from assertion of DTACKIN		40	ns
16	19	DTACKIN set-up time before falling edge of clock	15		ns
17	19	Data set-up time before assertion of DTACKIN	−15		ns
18	20	Asynchronous assertion of LACLR and MISS from ASN		55	ns
19	21	DTACKOUT delay from falling edge of clock (for write overlap only)		60	ns

NOTES:

1. Stresses above those listed under absolute maximum ratings may cause permanent damages to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltage larger than the rated maximum.
3. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8 and 2.0V as appropriate.

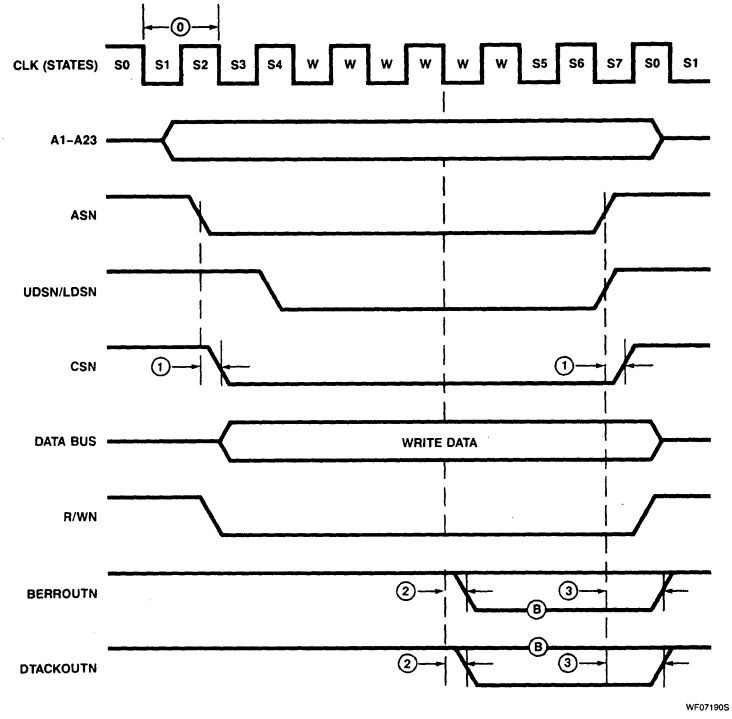


Figure 16. Write Register

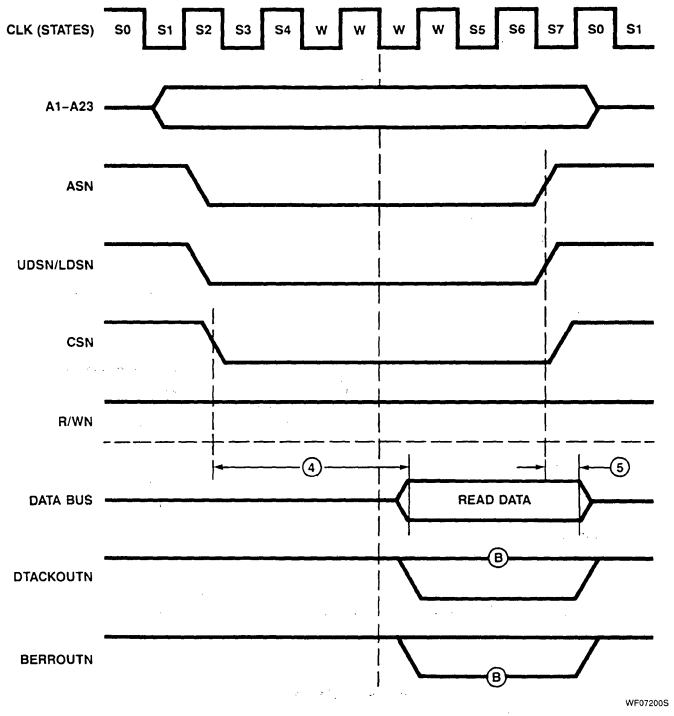


Figure 17. Read Register

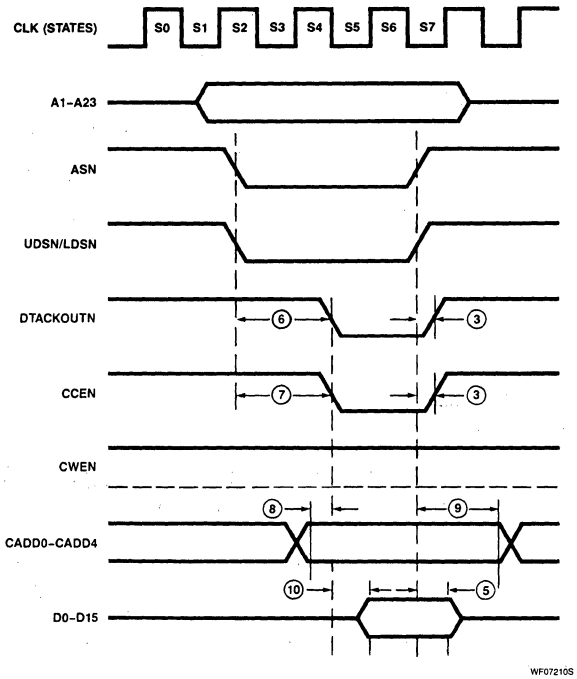
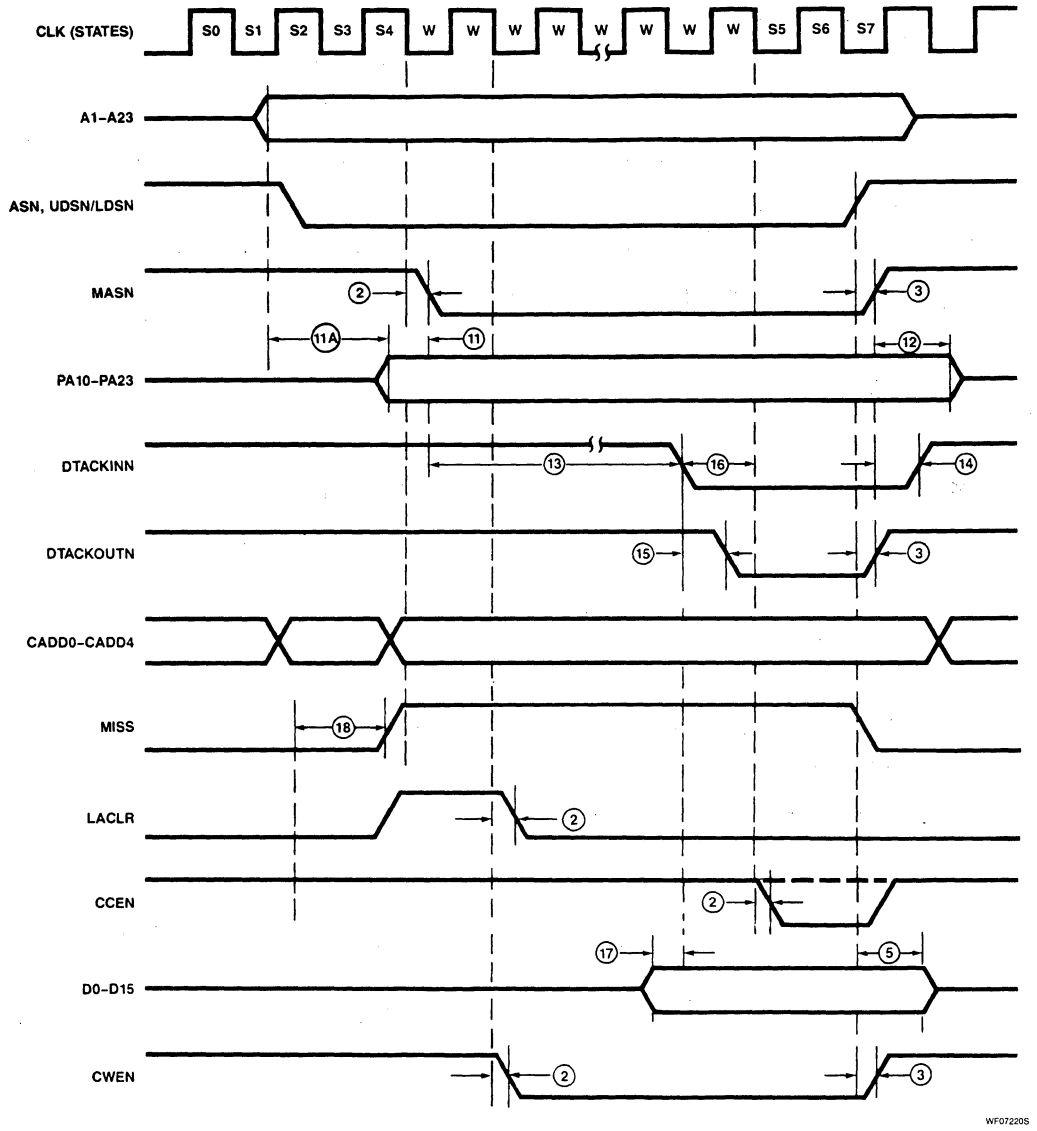


Figure 18. Read Hit



NOTE:
CCEN is not asserted when operation is non-cachable.

Figure 19. Read MISS or Non-Cachable Read

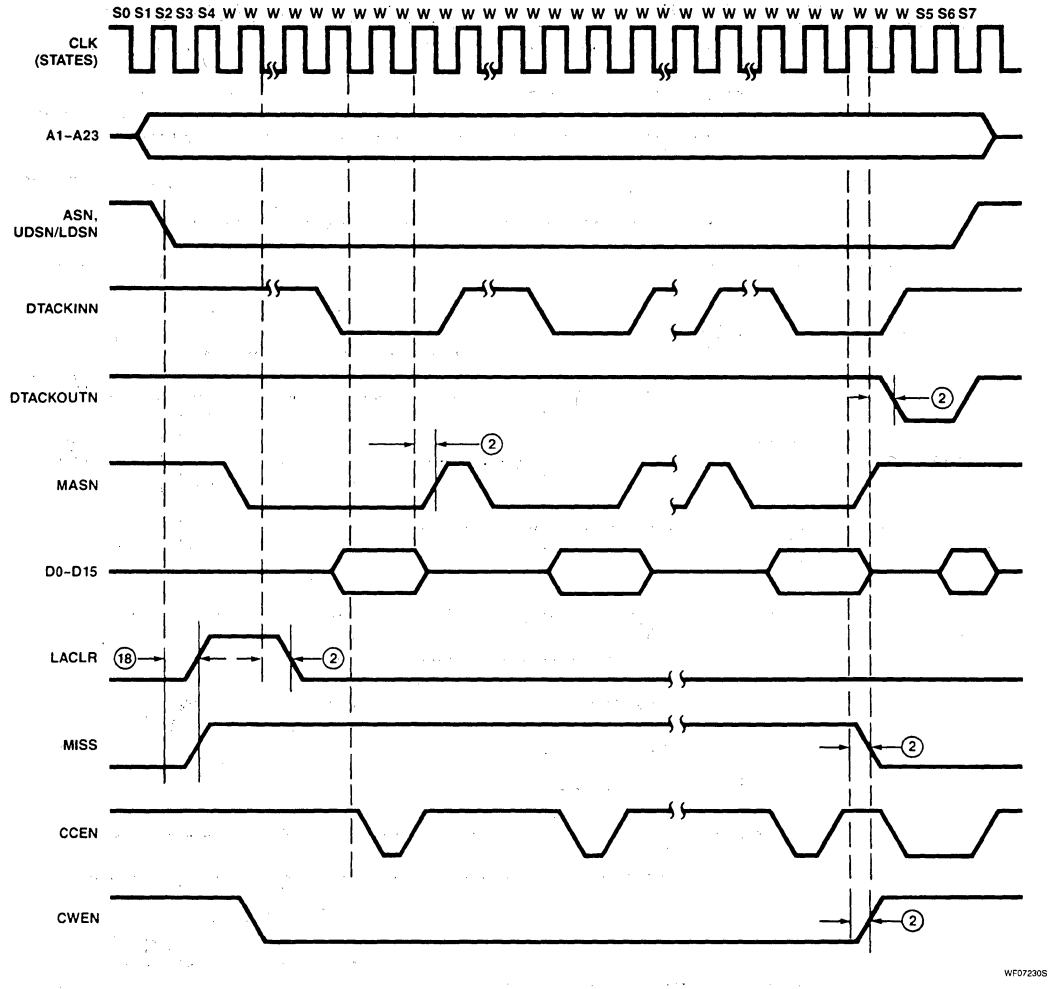
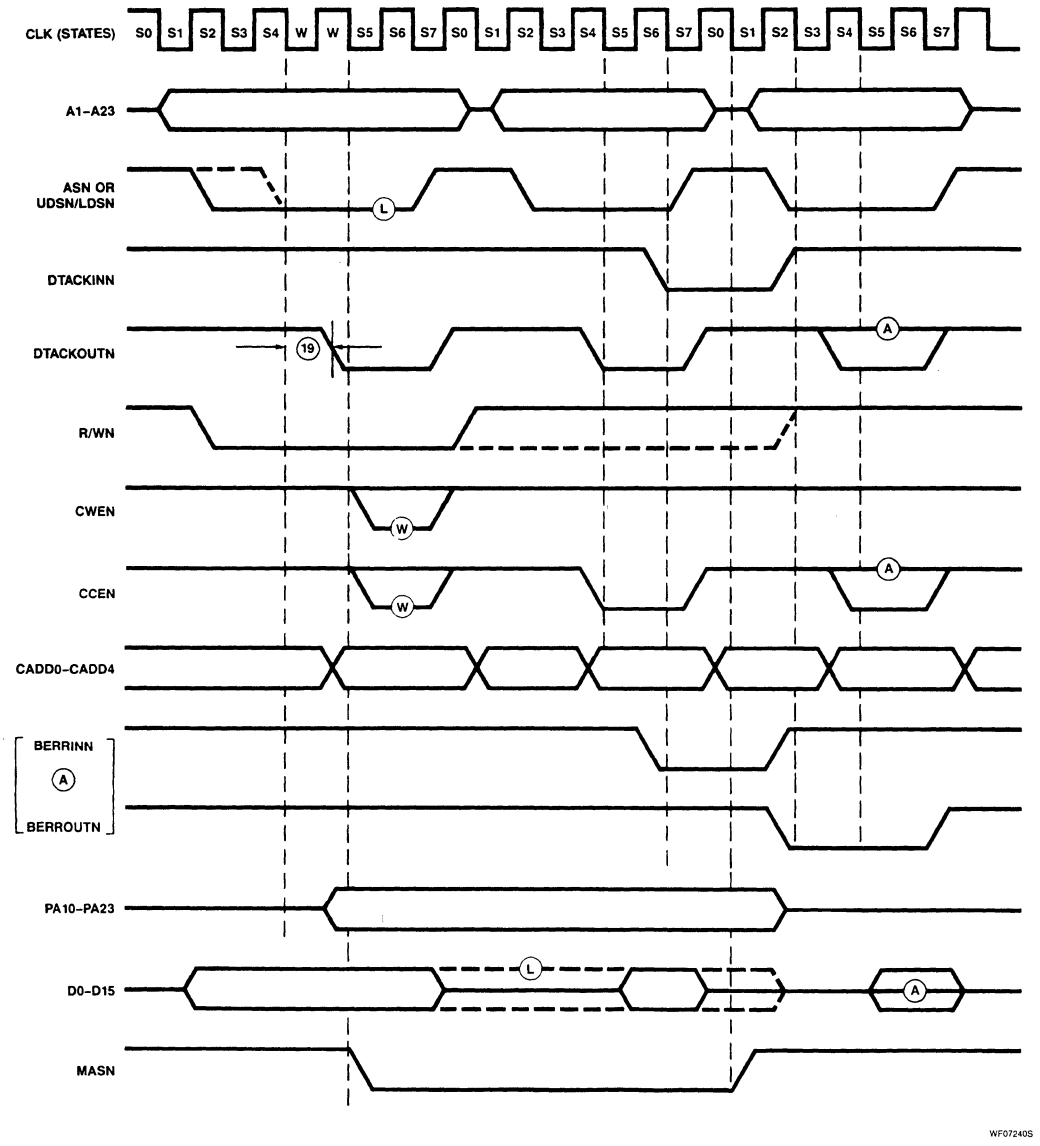


Figure 20. Read Cycle with Sequential Mode



NOTES:
BERR and DTACK are mutually exclusive.
Case (A) is associated with BERR only.

Figure 21. Write Overlap, Write followed by Read Hit

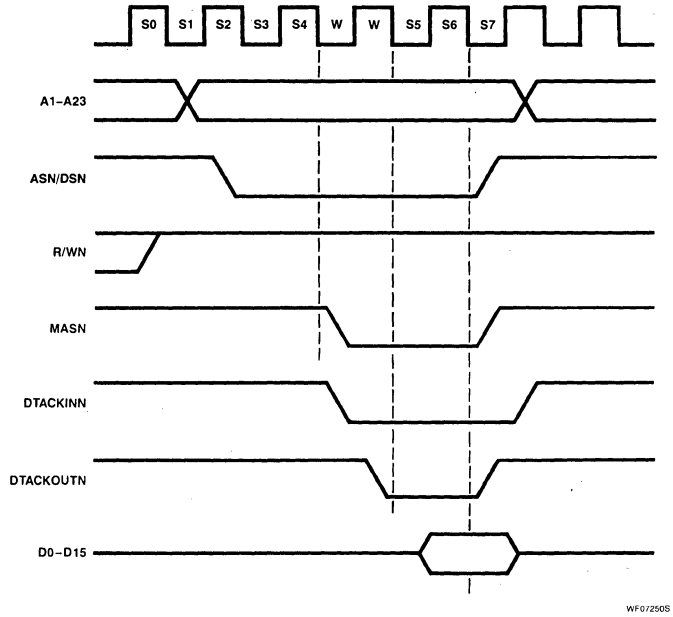


Figure 22. Memory Read; Cache Disabled

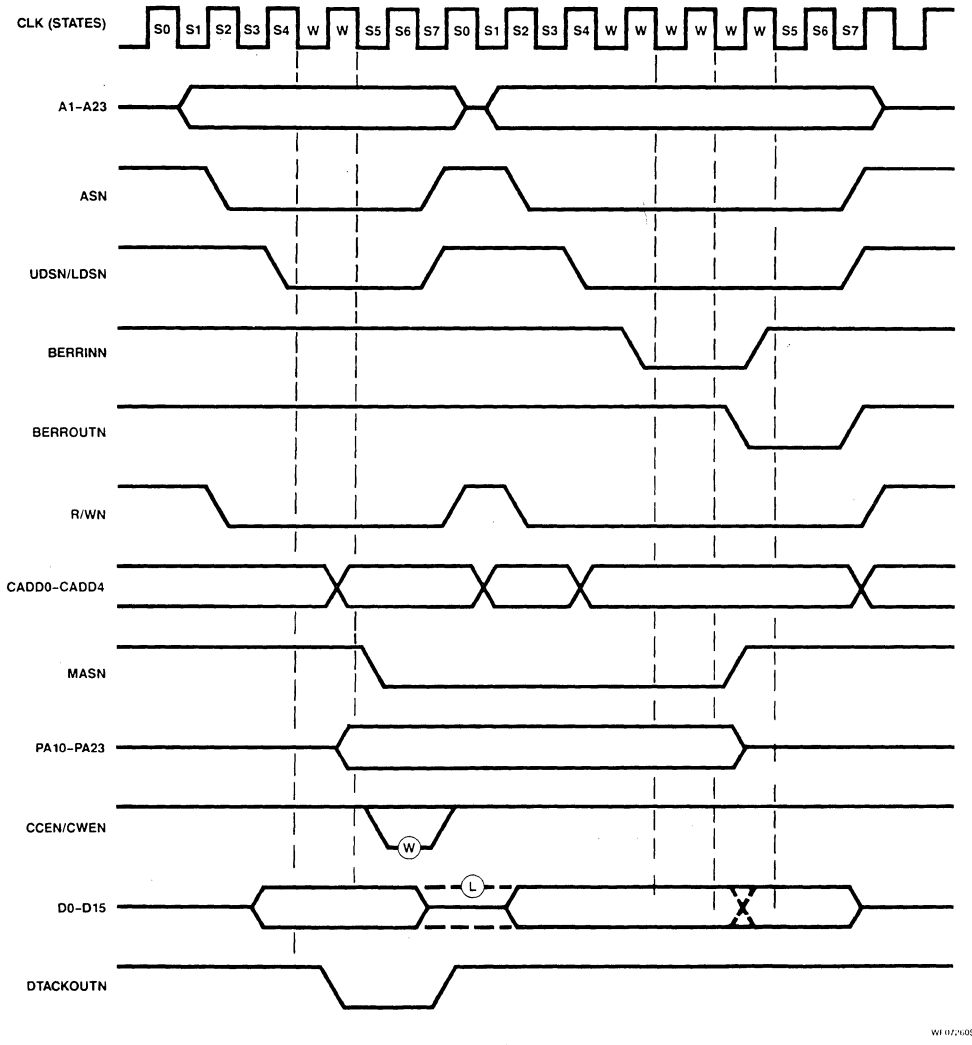


Figure 23. Write Overlap, Write with BERR followed by Write

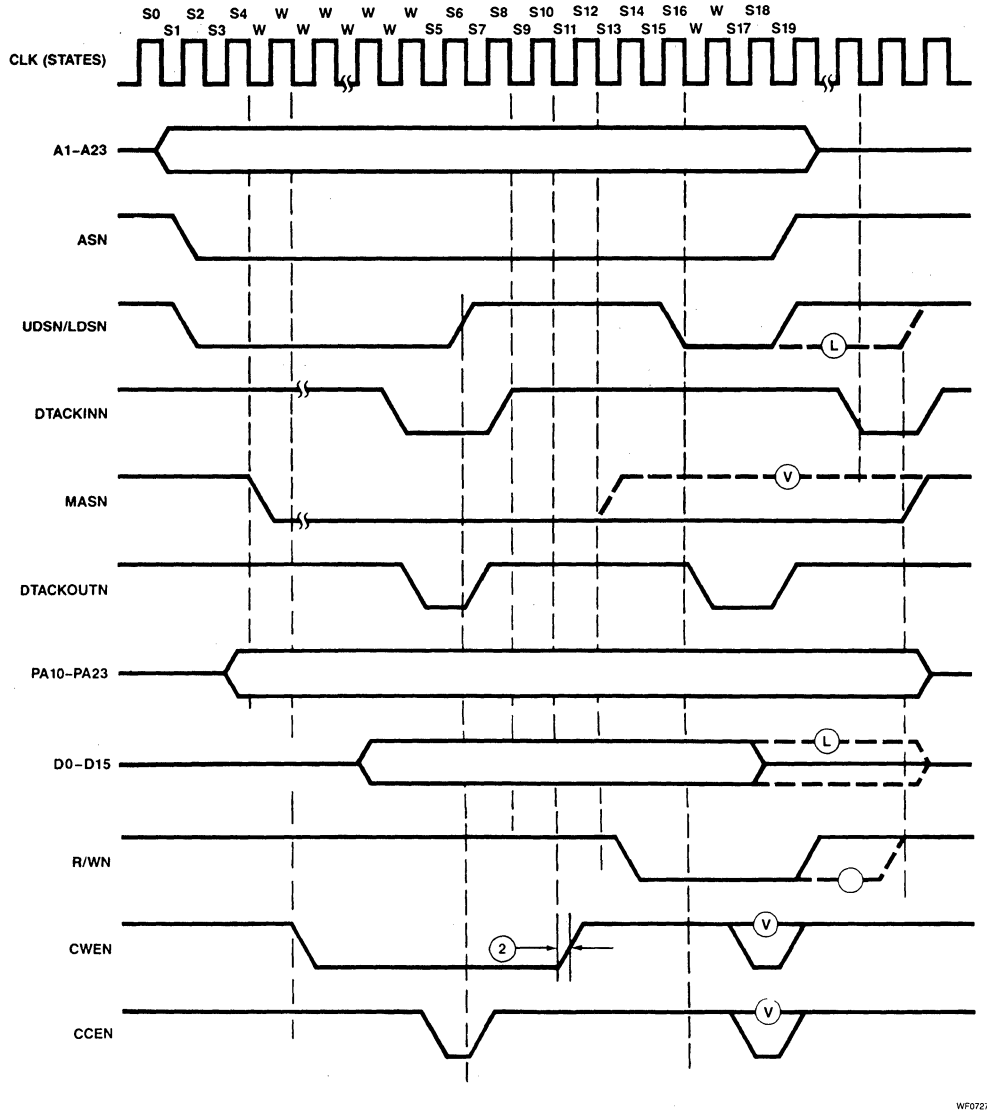


Figure 24. Read Modify Write with Read MISS

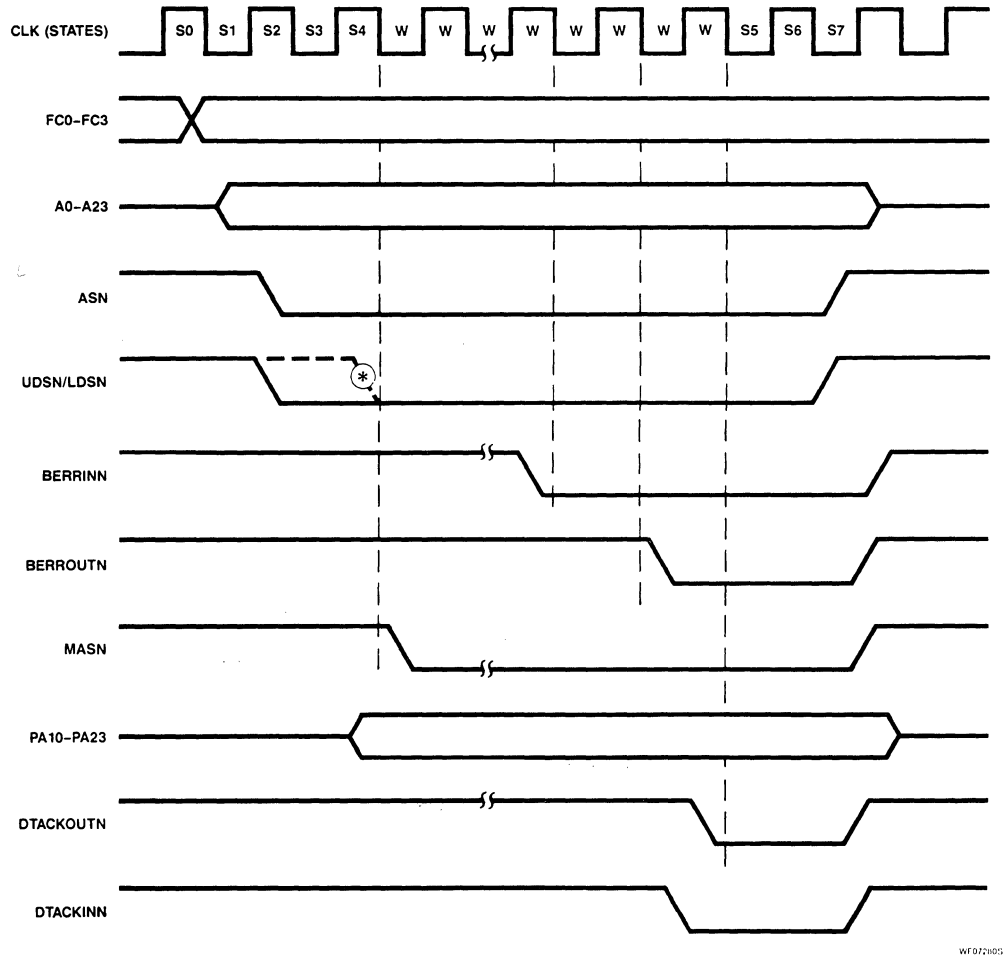


Figure 25. CPU Space Cycle or MAC Not Enabled

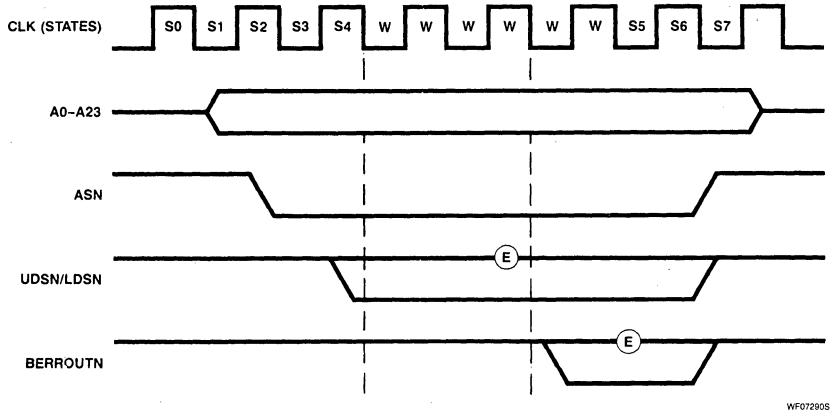


Figure 26. Write with Access Violation in Translation Look Aside Buffer or Dirty Bit Not Set or Translation Look Aside Buffer MISS

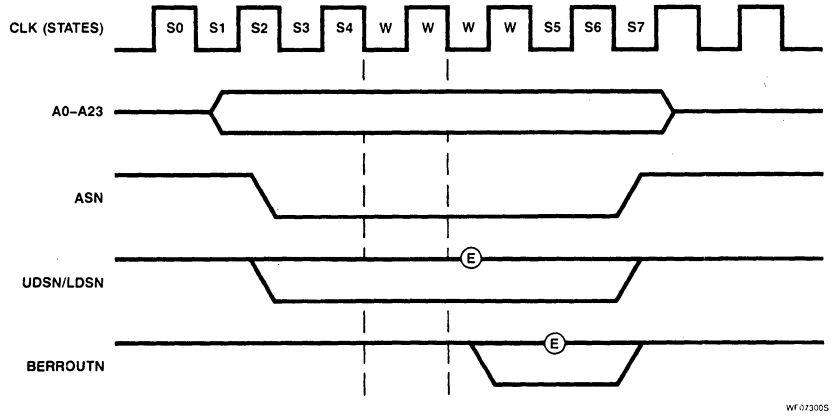


Figure 27. Translation Look Aside Buffer MISS or Read with Access Violation

SCC68906
Basic Memory Access Controller (BMAC)

Objective Specification

Microprocessor Products

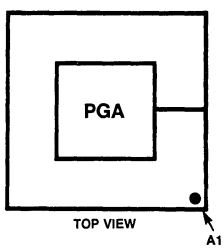
DESCRIPTION

The Signetics SCC68906 Basic Memory Access Controller (BMAC) is designed for the S68000 family (68000, 68020 and 68012) of microprocessors. It has a 32-bit logical address and a 28-bit physical address space. The BMAC mediates all accesses between the processor and system memory by controlling the memory hierarchy, translating virtual addresses into real addresses, supporting context switches and providing protection against unauthorized memory access. The operating system allocates the system memory and the BMAC implements these functions dynamically. The BMAC also provides support for a local memory. The SCC68906 is constructed using Signetics CMOS VLSI technology.

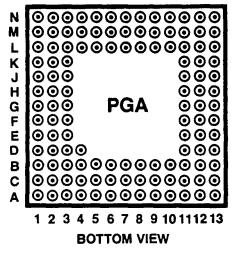
FEATURES

- **Mediates memory reference between processor and memory**
- **Functions as a demand-driven MMU and a cache controller**
- **No wait states on cache hits**
- **MMU action occurs in parallel with cache action**
- **Little mutual bus interference with up to four processors**
- **Ensures correct data in required local memory**
- **Variable length segments with paging, variable length contiguous segments, or paged only**
- **Allows partitioning the virtual address space into one to four regions**
- **Selective flushing by regions**
- **Provides four protection types at the page or segment level**
- **Maintains four history bits**
- **Configurable cache block size of 2, 4, 8, or 16 bytes wide**
- **Configurable cache depth of 512, 1024, or 2048 blocks**
- **Configurable page size of 1k, 2k, 4k, or 8k bytes**
- **Can use sequential accesses to fill cache if available on the system**

PIN CONFIGURATION



CD05350S



CD05340S

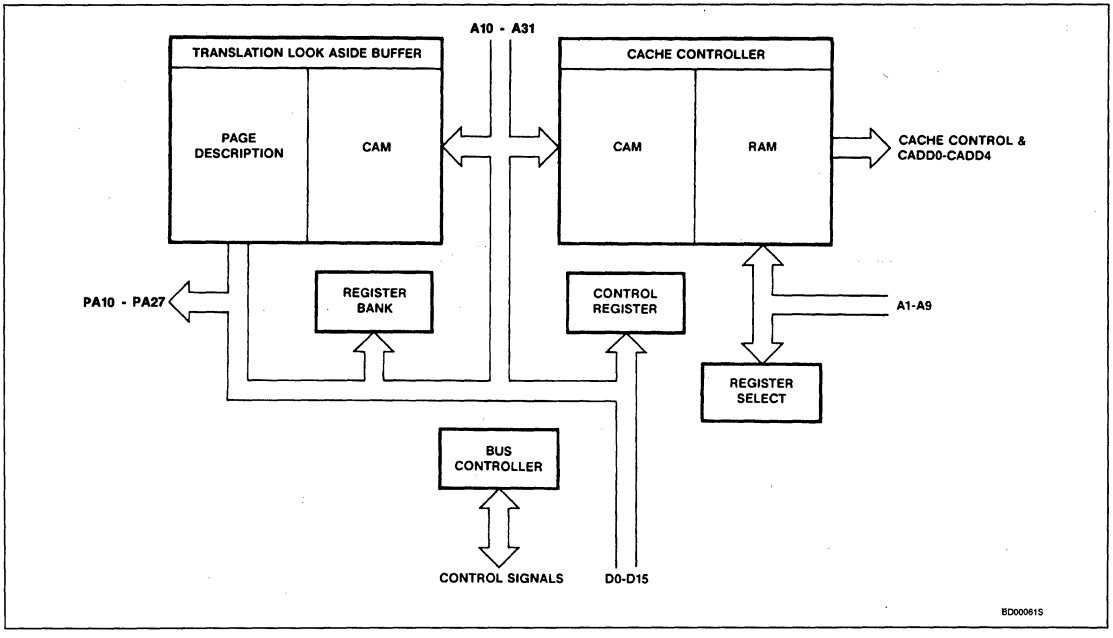
PGA	Function	PGA	Function	PGA	Function	PGA	Function
A-1	NC	D-5	NC	G-8	NC	K-11	NC
A-2	A3	D-6	NC	G-9	NC	K-12	PA16
A-3	A4	D-7	NC	G-10	NC	K-13	PA18
A-4	A6	D-8	NC	G-11	A23	L-1	NC
A-5	A7	D-9	NC	G-12	V_{CC}	L-2	NC
A-6	A9	D-10	NC	G-13	A22	L-3	NC
A-7	CADD4	D-11	NC	H-1	D13	L-4	D5
A-8	CADD2	D-12	A14	H-2	D12	L-5	D1
A-9	CADD1	D-13	A16	H-3	D11	L-6	BERROUTN
A-10	CADD0	E-1	NC	H-4	NC	L-7	MASN
A-11	A11	E-2	R/WN	H-5	NC	L-8	CCEN
A-12	PA23	E-3	RESETN	H-6	NC	L-9	PA10
A-13	NC	E-4	NC	H-7	NC	L-10	PA14
B-1	NC	E-5	NC	H-8	NC	L-11	NC
B-2	NC	E-6	NC	H-9	NC	L-12	NC
B-3	A1	E-7	NC	H-10	NC	L-13	PA15
B-4	A24	E-8	NC	H-11	CLOCK	M-1	NC
B-5	A25	E-9	NC	H-12	BERRINN	M-2	NC
B-6	A26	E-10	NC	H-13	ASN	M-3	D4
B-7	CADD3	E-11	A15	J-1	D10	M-4	D2
B-8	A28	E-12	A17	J-2	D9	M-5	PA24
B-9	A29	E-13	A18	J-3	D7	M-6	MISS
B-10	A30	F-1	V_{CC}	J-4	NC	M-7	CWEN
B-11	A12	F-2	LDSN	J-5	NC	M-8	PA26
B-12	PA22	F-3	UDSN	J-6	NC	M-9	DTACKOUTN
B-13	NC	F-4	NC	J-7	NC	M-10	PA11
C-1	FC2	F-5	NC	J-8	NC	M-11	NC
C-2	FC1	F-6	NC	J-9	NC	M-12	NC
C-3	NC	F-7	NC	J-10	NC	M-13	NC
C-4	A2	F-8	NC	J-11	PA17	N-1	NC
C-5	A5	F-9	NC	J-12	PA19	N-2	NC
C-6	A8	F-10	NC	J-13	PA20	N-3	D3
C-7	A27	F-11	A19	K-1	D8	N-4	D0
C-8	V_{SS}	F-12	A20	K-2	D6	N-5	LOCAL
C-9	A10	F-13	A21	K-3	NC	N-6	PA25
C-10	A31	G-1	D14	K-4	NC	N-7	V_{SS}
C-11	PA21	G-2	NC	K-5	NC	N-8	DTACKINN
C-12	NC	G-3	D15	K-6	NC	N-9	LACLR
C-13	A13	G-4	NC	K-7	NC	N-10	PA27
D-1	CSN	G-5	NC	K-8	NC	N-11	PA12
D-2	FC3	G-6	NC	K-9	NC	N-12	PA13
D-3	FC0	G-7	NC	K-10	NC	N-13	NC
D-4	NC						

Basic Memory Access Controller (BMAC) SCC68906

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C
Ceramic PGA	SCC68906CCP120

BLOCK DIAGRAM



The BMAC provides the basis for a total memory management solution. The BMAC combines a memory management unit (MMU) and a cache controller in the same package. The MMU contains locations for 32 descriptors which are loaded on demand. Also, the BMAC contains hardware support to define the region context and boundaries. The cache controller provides the matching and control logic for a virtual cache which reduces memory access times by at least one clock cycle. With the BMAC, a cache hit will result in a no wait state access for even the fastest microprocessors.

The MMU and the cache are programmer configurable. The BMAC allows the system programmer to choose segments, pages or paged segments and the cache size is configurable in both width and depth.

The BMAC is functionally compatible with the MAC (SCC68920); updating the MMU is accomplished by software on the BMAC and by hardware on the MAC.

FUNCTIONAL DESCRIPTION

In most real modular systems, a 16MHz 68020 with a MMU needs sev eral wait states when it accesses memory. The BMAC reduces this problem by functioning as a cache controller and a memory management unit. It performs access control of the memory hierarchy, translation of virtual to real addresses, supports context changes, and protects against unauthorized access during memory processing.

The operating system is responsible for allocating memory and access rights. The memory hierarchy can consist of three intermediate access store elements; a cache, a local memory, and the system memory. The cache memory element contains a copy of scattered sections of the local and system memories and is transparent to the software. The cache is a small, fast memory that allows the processor to see the cache access time instead of the usual system memory access time for 90% of its accesses and reduces the bus usage by 85%. The BMAC implements these functions dynamically.

Local memories can be attached to each processor in a multiprocessor system. The BMAC provides facilities to control the movement of code and data in and out of these local memories. This feature is transparent to the user software.

A demand paged virtual memory system can be implemented with one or more BMACs. The BMAC makes a multiprocessor system practical by reducing bus utilization. The BMAC has a special provision that allows multiple processing or processors to operate even if they share common data. If an 85% reduction in bus usage is not adequate, a local memory can then be attached to each processor which will reduce it further.

PIN DESCRIPTION

Signal names ending in 'N' are active low. All other signals are active high. In this data sheet, signals are discussed using the terms 'active' and 'inactive' or 'asserted' and 'negated' independent of whether the signal is active in the high (logic one) state or the low (logic 0) state. Refer to the individual pin descriptions for the definition of the active level of each signal.

MNEMONIC	PIN NO. PGA	TYPE	DESCRIPTION
A31 – 10	C-10, B-10, B-9, B-8, C-7, B-6, B-5, B-4, G-11, G-13, F-13, F-12, F-11, E-13, E-12, D-13, E-11, D-12, C-13, B-11, A-11, C-9	I	**Address Lines:** Active high. High order virtual address lines issued by the processor.
A9 – 1	A-6, C-6, A-5, A-4, C-5, A-3, A-2, C-4, B-3	I	**Address Lines:** Active high. These lines are the page address or when CSN is asserted, the BMAC register address. These are I/O lines on the MAC.
PA27 – 10	N-10, M-8, N-6, M-5, A-12, B-12, C-11, J-13, J-12, K-13, J-11, K-12, L-13, L-10, N-12, N-11, M-10, L-9	O	**Physical Address:** Active high, three statable. Some or all of pins PA12-10 may always be tri-state on the BMAC, depending on the page size programmed in the PDR. This is done to maintain campatibility with the MAC.
CADD4 – 0	A-7, A-8, A-9, A-10	O	**Cache Address:** Active high. High order cache address lines.
D15 – 0	G-3, G-1, H-1, H-2, H-3, J-1, J-2, K-1, J-3, K-2, L-4, M-3, N-3, M-4, L-5, N-4	I/O	**Data Lines:** Active high, three-statable. These lines are used to transfer data between the processor and the BMAC when the CSN line is asserted.
FC2 – 0	C-1, C-2, D-3	I	**Function Codes:** Active high. These lines are issued by the processor.
FC3	D-2	I	**Function Code:** Active high. Issued externally to differentiate I/O cycles from processor cycles.
ASN	H-13	I	**Address Strobe:** Active low. Issued by processor.
MASN	L-7	O	**Memory Address Strobe:** Active low. To memory issued by BMAC.
UDSN	F-3	I	**Upper Data Strobe:** Active low, three-statable. This signal is an I/O line on the MAC. This signal line requires an external tri-state buffer since the 680xx will not make this line high impedant when the BMAC is accessing memory.
LDSN	F-2	I	**Lower Data Strobe:** Active low, three-statable. This signal is an I/O line on the MAC. This signal line requires an external tri-state buffer since the 680xx will not make this line high impedant when the BMAC is accessing memory.
DTACKINN	N-8	I	**Data Transfer Acknowledge In:** Active low. Data acknowledge input from memory.
DTACKOUTN	M-9	O	**Data Transfer Acknowledge Out:** Active low, open-drain. Data acknowledge output to processor.
R/WN	E-2	I	**Read/Write:** Active high for read, low for write. This signal is an I/O line on the MAC. This signal line requires an external tri-state buffer since the 680xx will not make this line high impedant when the BMAC is accessing memory.
CSN	D-1	I	**Chip Select:** Active low.
CCEN	L-8	O	**Cache Chip Enable:** Active low. This signal has to be qualified externally by the data strobe signals.
CWEN	M-7	O	**Cache Write Enable:** Active low.
LOCAL	N-5	O	**Local:** Active high, three-statable. Local memory enable.
BERRINN	H-12	I	**Bus Error In:** Active low. Bus error signal, input from bus. During a memory access, DTACKINN and BERRINN cannot both be asserted unless BERRINN is asserted two clock cycles before DTACKINN.
BERROUTN	L-6	O	**Bus Error Out:** Active low, open-drain. Bus error signal, output to processor.
CLOCK	H-11	I	**Clock:** Active high. Clock input.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. PGA	TYPE	DESCRIPTION
RESETN	E-3	I	**Reset:** Active low. When asserted this signal will reset the BMAC.
MISS	M-6	O	**MISS:** Active high. When cache is enabled, it provides a cache read miss signal. It enables an external counter to be incremented on a block fetch during sequential mode.
LACLR	N-9	O	**Latch Clear:** Active high. This line is pulsed low to clear the counter for a block fetch.
V_{CC}	G-12, F1	1	+5volt ±5% power input.
V_{SS}	N-7, C-8	I	Power ground return.

MEMORY STRUCTURE

The BMAC uses a memory which is divided into variable length segments, which in turn can be divided into fixed length pages. Even if pages are not used, the BMAC measures the length of the segment in multiples of 1, 2, 4, or 8k bytes. As any segment can be any number of pages in length, the system can work if desired with only two segments - program and data. Protection is provided on a segment basis or page basis. Segments or pages, can have the following protection attributes:

1. Supervisor permission
2. Read permission
3. Write permission
4. Execute permission

These attributes are recognized and protected dynamically by the hardware. If a user attempts to access a segment without permission, a bus error (BERR) signal is generated.

Asido from protection attributes, the BMAC supports three additional attributes which are used to control the memory hierarchy.

1. Non-cacheable. This attribute forces associated segments or pages to never be cached. It forces segments or pages which are shared between processes to have only one copy which is always up to date.
2. Local. If the system has local memory then this attribute is used to force the associated page into the memory local to the processor and saves bus accesses.
3. Contiguous. This attribute, if set, indicates a segment that has no page table associated with it. Such segments are to be loaded and relocated in memory as a whole.

CACHE

A cache contains a copy of portions of program and data which are in the memory. Its associated principle works because most words of memory are accessed more than once within a relatively short time interval. The BMAC automatically stores each word read by the processor in the cache and, if the processor tries to read that word a second time, the BMAC provides the word from the cache instead of accessing the memory. When the processor writes a word, the BMAC implements a write-through policy, i.e., the word is written into both the cache and the memory. If a segment is marked as local, then this refers to the local memory and not the system memory. If the bus is not available to the processor, the BMAC allows the processor to proceed, and if the bus is granted before the next 'cache miss' or before the next write, the processor need not be aware that there was a bus access latency time.

Under certain conditions, e.g., when two processors are sharing access to a common segment of memory interleaved in time, it may not be desirable to cache the data. These segments can be marked as non-cacheable.

Sector-Associative Caches

Sector-associative caches are attempts to temper the idea of an associative cache with a dose of reality. Associative caches have an identifier attached to each cache word. The identifier is a content addressable memory (CAM) which holds the high order bits of the address. Basically, a sector-associative cache adds one assumption to the idea of associative caches. The assumption is that there is a working set of memory locations which in fact is reasonably small and tend to cluster.

In a sector-associative cache, the cache is divided into sub-caches. A sub-cache can contain a whole sector. Note that as far as the TLB is concerned, main memory could be logically partitioned into pages and segments, while as far as the cache is concerned, main memory could be logically partitioned into fixed size sectors. Since sectors are naturally large, filling a cache one sector at a time is not practical, therefore, sectors are partitioned into blocks made up of one, two, four or eight words. Thus, cache transfers imply block transfers to a particular sub-cache. Also note that not every block in a sector is necessarily in the cache.

An address, n bits in length presented to the cache, can be divided into three fields:

S bits (sector) – high order address bits which define the sector

B bits (block) – used to access the sub-cache (the algorithm will be given later)

O bits (offset) – used only if the block size is greater than the word size. The block size is the width of the data path between the cache memory and the memory. It can be wider than the processor's data bus.

ADDRESS FORMAT

Sector (S Bits)	Block (B Bits)	Offset (O Bits)

To determine if a sector is in the cache, the controller divides any virtual address it receives into sector, block, and offset fields. If the sector field matches the contents of one of the CAM positions, then the next step is to see if the particular block is present. Thus, a presence bit per cache block is needed. When a block is read into the cache for the first time the presence bit is set. Subsequently, when the block is accessed, the cache controller checks to see if the sector has been mapped into a sub-cache and that the presence bit is set. If both are affirmative, the word is accessed out of the cache. Note that the path to memory can be wider than the path to the processor. The O offset bits are not used in the cache controller but are used externally to select the word from the block. If a block is to be read into the cache using multiple reads, this is controlled by the BMAC.

The basic algorithm to see if a word is present in the cache is (see figure 1):

1. Compare the sector field against the CAM. If there is a match go on. If not, a miss has been recorded and go to step 3.
2. Each sub-cache has a 2^B x 1 RAM associated with it in order to see if the desired block (word) is present, i.e., each block has a presence bit associated with it. Use B to access this RAM. If the presence bit is set, then the required data is present and the O offset bits can be

used to access it. If the presence bit is not set, then a memory access or memory accesses load the data into the cache setting the relevant RAM bit.

3. This step is taken only if the address space denoted by the sector has no sub-cache allocated to it. Therefore, a sub-cache needs to be allocated. The oldest allocated sub-cache is freed, i.e., a sector sub-space is de-allocated and the needed sector is allocated to the sub-cache. All the presence bits of the sub-cache are reset to zero and step 2 is implemented. In this case a new mapping has to be made between the sub-caches and the addresses. Note that steps 1 and 2 can be taken in parallel, with the output of step 1 being used to validate the output of step 2.

Note that this replacement algorithm is essentially a FIFO mechanism. Simulations have shown that, with a working set the size of the BMAC's, there is no real performance difference between a FIFO algorithm and other replacement algorithms.

VIRTUAL MEMORY

The view of virtual memory presented here is supported by a mixture of hardware and software. It may well appear to be too complex for simple implementations, or not complex enough for high security systems. In the former case, the system can be simplified. In the latter case, the hardware protection can be extended by software to increase the degree by subtlet or range of the protection mechanisms. These software extensions are protected by hardware so that they cannot be circumvented, i.e., a virtual address can be split into three portions.

1. Segment number
2. Page number
3. Offset within a page

This split is supported by hardware. The split between 1 and 2 is a convention of the software established at SYSGEN time, and the BMAC can be programmed accordingly. The choice of the split between 2 and 3 is more restricted and is also handled by the BMAC.

Partitioning of the Address Space

The virtual address space, as seen by the processor and the software, can be partitioned into up to four different regions each of which has its own segment table (first-level table). This partitioning need not occur, i.e., the address space can be left as one region. These regions which are configured by software are provided for a fast and protected operating system.

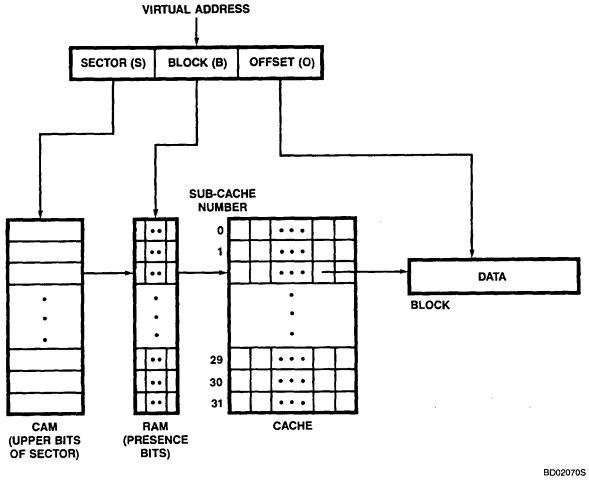


Figure 1. Addressing the Cache

Virtual Address Space Regions

Four regions are defined sequentially starting from the bottom end of the virtual address space. The upper limit of each region is specified in turn. This implicitly defines the lower limit of the next region. Thus if only one region is to be specified, then its upper limit is given as FFFFFFFF. The four regions are all equivalent.

Note that the virtual address space can be defined by the address only. The function codes are then used for protection. The concept of address spaces in the 68451 MMU is replaced here by the concepts of the regions. The latter are used in a similar but less restricted fashion than the former to enable different processes each to have its own unique virtual address space. Naturally portions of this address space can be shared with other processes.

Each region on the MAC has its own segment table. Associated with each segment table is a region descriptor. The format for the region descriptor is in figure 2 and its use is shown in figure 3.

Note that the region descriptor refers to the segment table as its first-level table. The segment table defines which areas of the virtual address space are accessible to the current process. The contents of a region are flushed from the cache by writing into the appropriate flush region register. This can only be done in the supervisor mode. Region definition is done via the RDR register.

Use of the Regions in a Simple System

A simple system is defined as having only one region which is accessible by all function codes. Thus protection is performed by the protection bits associated with each segment or page. This system is configured by writing FFFFFFFF into all region definition registers, thus defining the upper boundary of region 0 (R0) to be FFFFFFFF and the lower boundary of R0 to be 00000000. The system can be used as if it had only one region which stretches uniformly from 0 to FFFFFFFF.

Full Use of the Regions

This section gives an example of how all four regions can be used in the implementation of an efficient operating system which protects users from one another and from itself. Region R0 is used only by the privileged part of the operating system and contains the part of the kernel which is privileged. Thus, all segments in this region are marked with an indication that supervisor permission is needed to access them. This part of the operating system should be as small as possible, because it has access to all of memory.

The next region, R1, is used by the supervisor. This is the part of the operating system that provides the non-privileged functions. This code is nevertheless 'trusted code' and can have access to resources that are not available to user programs. On a system call, the handling routine in R0, not only switches the call to the relevant routine in R1 but also disables the privileged state and enables R1.

There is often a large body of utilities in a system which is used by most programs. If

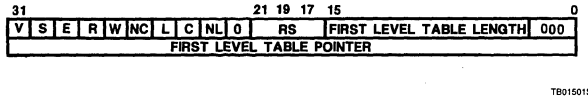


Figure 2. Region Descriptor

the data which they access is protected, then there is no reason why these utilities cannot automatically be made available to all users in an execute only mode. Thus, R2 is used as a utility region available to all processes and always enabled.

The last region, R3, is the user region. This region contains the process specific code and data. The mapping of this virtual address space into an associated physical address differs from process to process. This remapping is performed by changing registers in the BMAC.

Segment Tables

At any instance, there are 1 to 4 valid segment tables (denoted as first-level tables in the region descriptor) depending on how many regions have been defined and enabled. The segment table consists of descriptors, each with three fields.

Page table pointer (next table pointer) – This 32-bit field contains the pointer to the page table which defines the current segment. Page tables must be aligned on long-word boundaries in memory.

Page table length (next table length) – This 22-bit field contains the page number of the last page in the segment.

Segment protection – This 10-bit field contains the bits which define the status of the segment. Each use of the segment is protected differently depending on the permissions granted to each process. These bits are:

1. Valid bit (V). This bit is used to indicate whether the segment is valid or not.
2. Supervisor permission (S). If this bit is set, supervisor permission is necessary to access this segment.
3. Access permission bits (R, W, E). If these bits are set, corresponding access is available for the segment: read access (R), write access (W), execute access (E).
4. The following three bits define the mapping of the segment:
   a. Non-Cacheable (NC). If this bit is set, the associated segment will not be brought into the cache memory. For example, if a memory is used concurrently by the processors or memory addresses are used for I/O, this segment should be made non-cacheable.
   b. Local (L). If this bit is set, pages of the associated segment may belong to the local memory attached to the processor. It enables the use of local memory identification (LI) field.
   c. Contiguous (C). If this bit is set, the associated segment does not have a page table. It is segment-only and should be mapped into memory as a whole. Here the base address is given in units of 1K bytes.
5. Non-leaf (NL). NL is not used by the BMAC and must be set to zero.
6. Offset bit (O). This bit is used to denote whether pages of a segment start at the top (higher address, O = 1) or bottom (lower address, O = 0) of the segment's virtual address space.

For a contiguous segment configuration, additional bits are defined in the second word. These bits are defined for the MAC and should be set to 000000 for the BMAC to maintain compatibility.

Figure 4 shows the format of the segment descriptors in the segment table for a paged (non-contiguous) segment and figure 5 shows the format of the contiguous segment descriptors.

Page Tables

The page table (or next-level table) is split into two fields totalling four bytes which are always used by the hardware. They contain:

1. The 18-bit pointer to the physical address of the beginning of the page given in units of 1K bytes.
2. Three page-history bits.
   – Present (P): page is present in memory.
   – Used (U): page is used since this bit is last reset.
   – Dirty (D): page is written to since this bit is last reset.
3. Two reserved bits (Rs). These bits may not be used by software for any purpose.
4. Local memory identification (LI). The BMAC supports systems with up to 15 local memories. If a segment is marked local, those bits indicate in which local memory the page is resident. This will be checked against the contents of the processor identification register. This check should be done by the BERR routine which loads the descriptor into the SCC68906. If LI = 1111, then the page is considered to reside in system memory.
5. A non-cacheable (NC) attribute bit. If this bit is set, the associated segment will not be brought into the cache memory. For example, if a memory is used concurrently by the processors or memory addresses are used for I/O, this segment should be made non-cacheable.
6. Page protection bits:
   – Supervisor permission (S) needed to access this page
   – Three access permission bits. Read access (R), write access (W), execute access (E). The three access permissions bits allow the access mode if set.

Figure 6 shows the format of the page descriptors in the page table in memory.

Residence of System Tables

It may be desirable to place most of the system tables mentioned above in virtual memory without necessarily having them in physical memory all the time.

Memory Protection and Sharing

The BMAC provides mechanisms for implementing systems with varying degrees of protection. The simplest protected system would only separate user code from operating system code. Protection is basically provided only by giving a process access to the resources it needs. These resources include memory and I/O. Access to I/O is accomplished by giving a process access to the necessary I/O management routines. This access can be execute-only and the entrance to these routines can be controlled via hardware in the normal system call manner or by putting a software 'gateway' as the access point. This gateway is essentially an execute-only segment and/or page which can be marked so as to cause a trap when entered. Thus, a trusted program can be used to do the I/O access or to provide extra entry points to the calling program. For example, if a file is opened then this technique can add segments to the calling process's virtual address space, where each segment performs an I/O function. The process will only have access to these routines.

This process will only be effective if a process is forbidden to access its segment and/or page descriptors. This is achieved by not putting the segment and/or page tables belonging to a process in a segment and/or page directly accessible to the process. This segment and/or page is accessible to the operating system.

Sharing of code and data can be accomplished by merely putting a segment and/or page descriptor pointing to the same page table in more than one process's segment and/or page table. Note that these segment and/or pages can have different virtual addresses for the different process. Message passing can then be easily and efficiently implemented by asking the operating system to 'add' a descriptor to another process's address space.

2

Sub-Setting Facilities for Simple Systems

The BMAC can be used as an MMU either with or without using the cache controller function. If a simple system is desired, a subset of the BMAC's capabilities can be used. The following are some examples of possible system configurations:

1. A purely paged system
2. A minimally protected system. This would have one region with two segments for code and data, with no paging within each segment.
3. A minimally protected stack-based system. This would have one more segment for each stack used to be used to easily allocate stack space as it is needed, while still controlling unlimited growth.

These examples are provided to emphasize that all the facilities need not always be used.

REGISTERS

The registers on the BMAC are used for three purposes.

1. By the processor to inform the BMAC of the hardware system configuration.
2. To force the BMAC to invalidate the cache descriptor entries.
3. By the BMAC to inform the processor of its status.

The BMAC will issue a BERR if chip select is asserted without using a supervisor function code. Registers on the BMAC can be accessed by the processor when the chip select input is active and address line A9 = 0. The register addresses are then decoded from address inputs A6–1.

Page Definition Register (PDR)

Only bits 0 – 1 of this register are used. These bits indicate the size of the page. Note that depending on the page size, some physical address pins of the BMAC will always be in tri-state mode. The pins that are tri-stated are listed below:

Bits 1 0	Page Size (Bytes)	Tri-stated Lines
0 0	1024	–
0 1	2048	PA10
1 0	4096	PA10-11
1 1	8192	PA10-12

Size of Segment Register (SSR)

Only bits 0 – 3 are used. This register, which is reset to zero on a BMAC reset, defines the part of the address that selects a segment. The contents of the SSR is the encoded number of address bits (starting with A31) that are part of the segment number. The segment size can vary between 512k and 64M bytes in size.

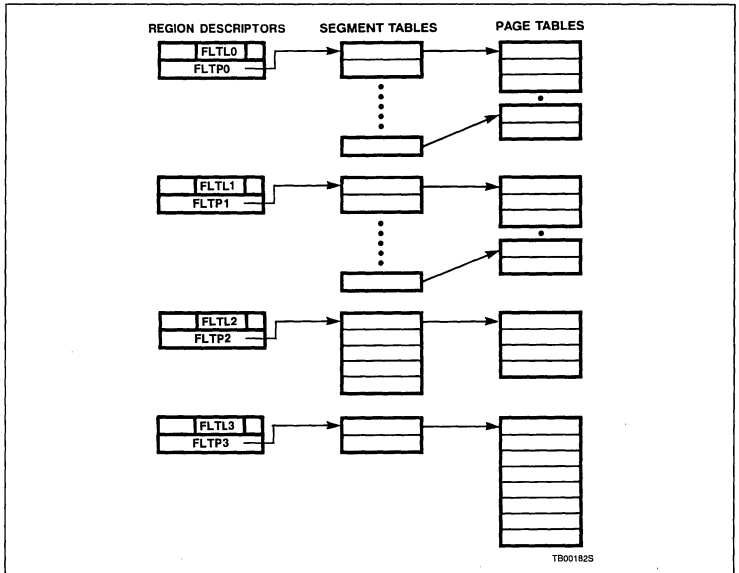


Figure 3. Table Structure

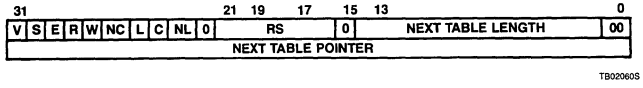


Figure 4. Segment Descriptor for a Paged Segment (C = 0)

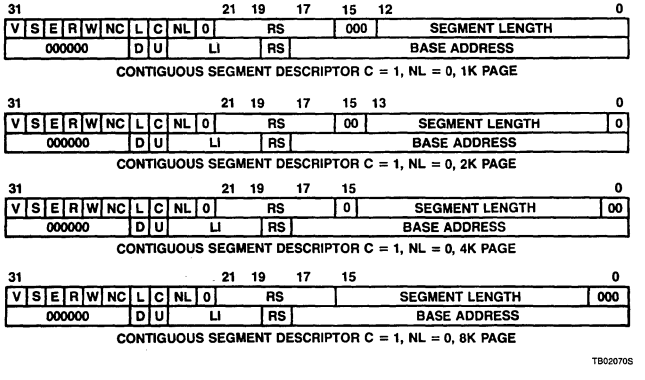


Figure 5. Segment Descriptors For A Contiguous Segment

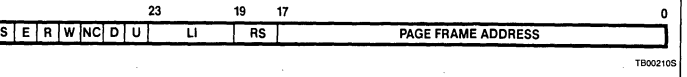


Figure 6. Page Descriptor For A Paged Segment

If less than five address bits are used for the segment number, the split of the address space into regions is no longer supported. In those cases the whole address space will be treated as region 0.

Size of Cache Register (SCR)

Only bits 0 – 3 are used. This register specifies the cache organization. The meaning of the bits which are all reset to zero on a BMAC reset are:

Bits 3 – 0	Block Size	Cache Depth
0000	16 bits	512
0001	32 bits	512
0010	64 bits	512
0011	128 bits	512
0100	16 bits	1024
0101	32 bits	1024
0110	64 bits	1024
0111	128 bits	1024
1000	16 bits	2048
1001	32 bits	2048
1010	64 bits	2048
1011	128 bits	2048
11xx	Reserved	

Note that bits 2, 3 define the cache depth and bits 0, 1 define the cache block size.

Region Definition Register (RDR)

This register defines the boundaries of regions 0, 1 and 2 explicitly and region 3 implicitly. The register is split up into three fields which define the five most significant bits of the address for each region. These bits are then extended to a full address with 26 1's in the least significant bit positions. Bits 0 – 4 define the highest address in region 0. Region 1 is defined from the top of region 0 to the address specified by bits 5 – 9 of the RDR, extended with 26 1's. Similarly, region 2 is defined by the top of region 1 at its low end and bits 10 – 14 of RDR at its high end. Region 3 is defined from the top of region 2 to FFFFFFFF at its top end. Bit 15 is not used.

Processor Identification Register (PIR)

Only bits 0 – 3 are used. The number encoded in the four bits of this register identifies one of the 15 processors. Code 1111 in this register is reserved.

BMAC Mode Register (MMR)

Only bits 0 – 1 are used. On reset, the BMAC resets this register to zero. It has to be set if the BMAC is to be used. When the BMAC is disabled, the cache is effectively not present and all addresses are passed through the BMAC without any change. When the BMAC is enabled, the addresses are translated in the chosen manner and the cache is used in the configuration given in the SCR. If bit 0 is set, it indicates that the cache memory is enabled and if bit 1 is set, it enables the BMAC.

Note that loading this register will enable the BMAC. This register should only be loaded when all the other registers have been loaded.

MMR Bit 1	MMR Bit 0	FUNCTION
0	0	BMAC disabled. Physical address out equals virtual address in.
0	1	BMAC enabled for cache operation only.
1	0	BMAC enabled for MMU operation only.
1	1	BMAC enabled for MMU operation and cache operation.

Memory Fetch Register (MFR)

Only bits 0 – 1 are used. This register indicates how many memory accesses are needed when the BMAC loads a cache block.

Bit 1	Bit 0	No. Memory Accesses
0	0	1
0	1	2
1	0	4
1	1	8

The number of memory accesses also corresponds to the block size.

Flush Region Registers (FRR0 – 3)

The BMAC will respond with a DTACK when any of these registers is addressed. The FRR registers are 32 bits wide; they are implemented by two 16-bit registers, upper and lower word. When the FFRs are loaded, all the mappings in the BMAC in the relevant region are reset, i.e., the cache is flushed and all addresses in the region and all page descriptors in the BMAC in the region's address space are invalidated.

MAC Communication Area Pointer Register (MCAPRU, MCAPRL)

Logically this register is 32 bits; it can be accessed by the upper and lower word. The BMAC will respond with a DTACK when this register is addressed. On the MAC, this register contains the physical address of the MAC communication area in memory.

Error Code Register (ECR)

This register contains the reason for the bus error. The error code is set when BERROUT is asserted by the BMAC and when register DESCR1 is loaded. The meaning of the bits is:

bit 0, 1 Contain the region number where the error occurred.

bits 2 – 6 5-bit error code. The following error causes are encoded for the BMAC. All other codes are reserved for future use.

	EC				
2	3	4	5	6	
0	0	0	0	0	No error
0	0	0	0	1	Page not in required local memory
0	0	0	1	0	Access permission violation
0	1	0	0	0	Illegal BMAC access
0	1	1	1	1	BERR on global bus

An access permission violation occurs when the current bus cycle does not have the access permissions required for the segment or page that it attempts to access. The access permissions of a page are S (supervisor permission needed), E (execute access), R (read access) and W (write access). These four bits are stored for each page in the page translation look-aside buffer (TLB) on the BMAC and they represent the exclusive OR of the corresponding bits in the segment and the page tables.

The current bus cycle is characterized by the signals FC3 or (processor/I/O cycle), FC2 (supervisor/user mode), FC1 (program/data access) and R/W (read/write). Access permissions will not be checked during an I/O cycle (FC3 = 0). Table 1 shows the required access rights (S, E, R, W) for all possible non CPU-space bus cycles. All other combinations will result in an access permission violation.

bit 7 Valid only when bit 15 is set. If ECR(15) = 1, this bit, if set, indicates a page descriptor miss; if not set, it means the dirty bit is not set on a write.

bits 8 – 10 Not used, reserved.

bits 11 – 14 S, E, R and W access permission bits of the currently accessed page. This field is valid only when an access permission violation is reported.

bit 15 If set, it indicates a TLB miss; to be handled by the BMAC TLB loading software routine. If not set, an access error as specified by the error code has occurred.

Descriptor Address Register (DAR)

Only bits 4–0 are used. This 5-bit register contains the address of the next slot to be

Table 1. ACCESS RIGHTS

FC3 0: I/O 1: PROC	FC2 USER SUPERV	FC1 DATA PROGRAM	FC0	R/W WRITE READ	S	E	R	W
1	0	0	X	0	0	X	X	1
1	0	0	X	1	0	X	1	X
1	0	1	X	0	N	N	N	N
1	0	1	X	1	0	1	X	X
1	1	0	X	0	X	X	X	1
1	1	0	X	1	X	X	1	X
1	1	1	0	0	N	N	N	N
1	1	1	0	1	X	1	X	X
1	1	1	1	X	X	X	X	X
0	X	X	X	X	X	X	X	X

X = do not care, N = violation, illegal input combination.

used in the TLB. This register must be updated before the next descriptor is loaded into the TLB (see Memory Mapping Unit).

Descriptor Registers (DESCR0 – 2)

These registers are used to load descriptors. The BMAC requires the most significant 16 bits of the first long word of the segment descriptor to be written to register DESCR0, that the most significant 16 bits of the page descriptor be written to register DESCR1, and that the least 16 significant bits of the page descriptor be written to register DESCR2. Registers DESCR0, DESCR1, and DESCR2 must be written in that order. It is recommended that the user use the MOVEM instruction to transfer the data to the registers.

Register Summary

Tables 2 and 3 describe the register map and bit map formats.

BMAC OPERATION

The BMAC can be thought of as two cooperating units; the cache controller section and the memory mapping unit (MMU). Between them, all of the BMAC functions are provided.

The Cache Controller

The cache controller manages a sector-associative cache. It has a working set of 32 sub-caches. These sub-caches are defined not only by the address, but also implicitly by the address space. The BMAC recognizes four address spaces defined by the regions. The region definitions are independent of their use in the system. System designers can use them for whatever purpose is required. However, the same virtual address cannot be used in different regions. The distinction between the regions, in the cache controller, is made purely to allow the cache entries belonging to the separate address spaces to be invalidated for a region as a whole. If a system's dynamic working set is favorable, this will allow interrupt service routines, for instance, to remain in the cache while the processor continues a process or even changes processes.

On a reset, the cache is marked as empty and all 32 sub-caches are available to all four regions. All sub-caches which are in use by a region are automatically marked as empty whenever that region's flush region register is loaded. Accesses with a CPU address space function code 111 are never cached; they are passed through to the physical address bus. Upon a CPU space access (FC2 – 0 = 111 and ASN are asserted), the BMAC will pass the virtual address on to the physical address pins and assert MAS. When the DTACKIN (or BERRIN) becomes active, the BMAC will assert DTACKOUT (or BERROUT). The cycle will be finished when ASN is negated by the processor.

A word can only get into the cache if it has satisfied the permission criteria of the associated descriptors. Thus, when a word is read and it is found in the cache, it is valid if the processor has the necessary permissions. A copy of these permission rights is kept with the associated sub-cache. If a write is performed, then this is only written into the cache if the associated descriptor has write permission. If it is so desired, the cache can be automatically disabled on reset. It can be enabled by writing into the MAC mode register.

Table 2. BMAC REGISTER MAP

ADDRESS[1] 6 5 4 3 2 1	ACRONYM	REGISTER NAME	MODE	AFFECTED BY RESET	NOTES
0 0 0 0 0 0	PDR	Page definition register	R/W	Yes	
0 0 0 0 0 1	SSR	Size of segment register	R/W	Yes	
0 0 0 0 1 0	SCR	Size of cache register	R/W	Yes	
0 0 0 0 1 1	Reserved				
0 0 0 1 0 0	ECR	Error code register	R	Yes	
0 0 0 1 0 1	PIR	Processor indentification register	R/W	Yes	
0 0 0 1 1 0	MMR	MAC mode register	R/W	Yes	
0 0 0 1 1 1	MFR	Memory fetch register	R/W	Yes	
0 0 1 0 0 0	Reserved				
0 0 1 0 0 1	Reserved				
0 0 1 0 1 0	DESCR0	Descriptor register 0	W	No	
0 0 1 0 1 1	Reserved				
0 0 1 1 0 0	DESCR1	Descriptor register 1	W	No	
0 0 1 1 0 1	DESCR2	Descriptor register 2	W	No	
0 0 1 1 1 0	DAR	Descriptor address register	R/W	Yes	
0 0 1 1 1 1	Reserved				
0 1 0 0 0 0	Reserved				
0 1 0 0 0 1	FRR0	Flush region register 0	R/W	No	
0 1 0 0 1 0	Reserved				
0 1 0 0 1 1	FRR1	Flush region register 1	R/W	No	
0 1 0 1 0 0	Reserved				
0 1 0 1 0 1	FRR2	Flush region register 2	R/W	No	
0 1 0 1 1 0	Reserved				
0 1 0 1 1 1	FRR3	Flush region register 3	R/W	No	
0 1 1 0 0 0	Reserved				
0 1 1 0 0 1	RDR	Region definition register	R/W	Yes	
0 1 1 0 1 0	Reserved				
0 1 1 0 1 1	Reserved				
0 1 1 1 0 0	Reserved				
0 1 1 1 0 1	Reserved				
0 1 1 1 1 0	Reserved				
0 1 1 1 1 1	Reserved				
1 0 0 0 0 0	Reserved				
0 1 0 0 0 1	Reserved				
1 0 0 0 1 0	MCARU	MAC communication area pointer upper	R/W	No	2
1 0 0 0 1 1	MCARL	MAC communication area pointer lower	R/W	No	2
1 0 0 1 0 0	Reserved				
•					
•					
•					
1 1 1 1 1 1					

NOTES:
1. All registers are accessible as 16 word bits only. Attempt to access these registers as one 8-bit byte will result in an illegal access error.
2. These registers maintain compatibility with the MAC. Access to these registers in the BMAC will be responded to only with DTACK. No other actions will take place.

If a read miss occurs, the BMAC will fetch enough words to fill the associated cache block. It does so with the aid of an external counter (74F193), and is connected as shown in figure 7 with associated timing in figure 8.

The miss signal (MISS) is normally low and LACLR is also lo w so that the signals A3–1 pass through the 74F193. If a block is to be loaded, LACLR is raised high for one clock period so that the counter is cleared. MISS is also raised high for the duration of the load. Next, MASN is asserted and the first word is accessed. After each DTACKINN, the MAC will dis-assert MASN for a 1-1/2 clock period and reissue MASN as many times as specified in the MFR register. As the high going edge of MASN will increment the counter, the correct word address within the block will be produced. At the end of the block load, the BMAC lowers MISS so that the correct low order address is fed to the cache before asserting DTACKOUTN to the processor. After MISS gets disasserted, CCEN goes low so that the word or byte initially desired be read from the cache.

Note that if the SCC68172 BUSCON is used in the system and sequential accesses are necessary, then LACLR can be used to set an external 74F74 to indicate that a sequential access is being made. MISS going low at the end of the sequence will reset the flip-flop.

Write Through

When a write is accomplished, the information is written to memory. Information is also written to the cache only when a sub-cache is already allocated for this data and the relevant block presence bit is marked as valid.

Read/Write Overlap

On a write, the BMAC will strobe the address and data into external latches and start a memory write cycle. Before doing so, it will check if it has the required descriptor; if not, it will assert BERR. If it is not already set, the

Table 3. BMAC BIT REGISTER MAP[1]

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PDR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	pg size	
SSR	*	*	*	*	*	*	*	*	*	*	*	*	segment size			
SCR	*	*	*	*	*	*	*	*	*	*	*	*	cache size			
ECR	T	S	E	R	W	*	*	*	P/D	EC4	EC3	EC2	EC1	EC0	RN1	RN0
PIR	*	*	*	*	*	*	*	*	*	*	*	*	processor no.			
MMR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	MMU	cache
MFR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	count no.	
DESCR0	V	S	E	R	W	NC	NL	C	NL	O	*	*	*	*	*	*
DESCR1 (C = 0)	P	PS	PE	PR	PW	NC	D	U	LI3	LI2	LI1	LI0	*	*	*	*
DESCR1 (C = 1)	*	*	*	*	*	*	D	U	LI3	LI2	LI1	LI0	*	*	*	*
DESCR2 (PA)	*	*	23	22	21	20	19	18	17	16	15	14	13	12	11	10
DAR	*	*	*	*	*	*	*	*	*	*	*	TLB address				
FRR0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FRR1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FRR2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
FRR3	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
RDR	*	region 2 top msbs					region 1 top msbs					region 0 top msbs				
MCAPRU	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
MCAPRL	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

NOTE:
* Indicates null bits. All null bits will be read as zero.

BMAC will also set the dirty bit in the descriptor within the translation look-aside buffer on the BMAC and assert BERR to the processor. If the required descriptor is in the BMAC, then it will allow the processor to proceed by asserting the DTACK signal while the memory transfer is taking place. The processor will proceed unless it requests another memory transfer (either a cache read miss or a write) before the first write is complete.

Size of the Cache

The size of the cache is not infinitely variable, but can be expanded in both width and depth. The width can be expanded in units of 16 bits. The block can be configured to be 1, 2, 4, or 8 units wide, i.e., 16, 32, 64 or 128 bits wide. This width, the block size, is independent of the width of the data path to the processor. The number of memory accesses needed to fill a block is given by the memory fetch register.

The length of the cache can also vary and, again, the lengths to which it can be extended are limited. These limits are 512, 1024, or 2048 blocks deep. In terms of bytes the cache can then vary from a minimum of 1K bytes (512 blocks of 2 bytes) to 32K bytes (2K blocks of 16 bytes).

Although the actual number is application dependent, in general, a cache of 2048 blocks of four words, i.e., 2Kx64, will result in the processor seeing a cache access time for at least 95% of the memory accesses. For example, if a cache hit or an overlappped write occur fast enough not to cause any wait states in the processor and a memory access results in four wait states, then the average number of wait states per memory access is 0.2. These figures are realistic for a 16MHz 68010/20 coupled to a slow, 250ns access memory.

Memory Mapping Unit

The memory mapping unit (MMU) performs paging control between three levels of memory. These three levels are:

1. Backing store (Disk)
2. System memory (RAM)
3. Local memory (RAM)

Local memory is not always present. In a multiprocessor situation, it may be desirable in order to reduce bus traffic, to attach a local memory to each or some of the processors. If so, the BMAC is capable of providing signals to the processor if the segment is marked as belonging to local memory. This page fault trap will be generated if the accessed page is not present in the local memory. Thus the BMAC cannot only control paging between the system memory and the backing store, but also between local memories and the system memory or backing store.

The memory mapping unit section of the BMAC contains descriptors for 32 pages which are loaded on demand by the BMAC. As in the cache, each page is identified only by its logical address. In addition, the MMU uses four region descriptors. Each of these descriptors define the current context of its respective region. The region definition register is used to define region boundaries. The page descriptors are in a fully associative TLB on the BMAC. Each descriptor consists of the following fields:

1. An 18-bit physical address
2. Four permission bits
   - Supervisor permission needed (S)
   - Access permission bits:
   - Read access (R)
   - Write access (W)
   - Execute access (E)
3. Page management bits – Non-cacheable (NC)
   - Local (L)
4. A dirty bit (D), i.e., the page has been written into.

The processor must write the descriptor to be used into the descriptor registers. This is done by loading the first word of the relevant

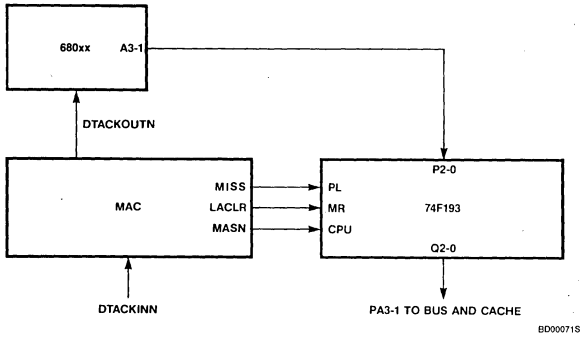


Figure 7. Cache Memory Block Transfer Circuitry

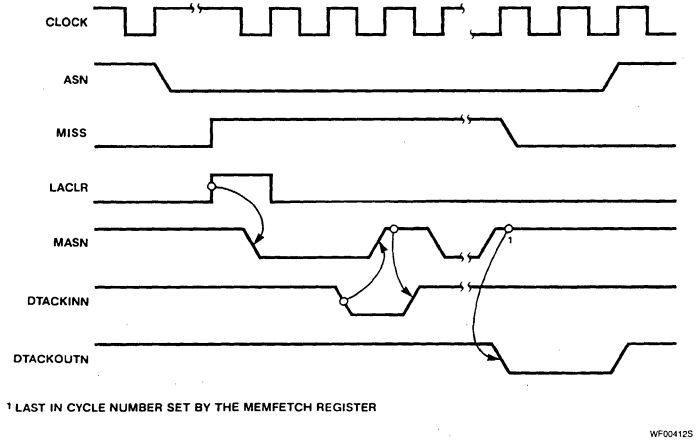


Figure 8. Memory Fetch Cycle Timing

segment table descriptor into register DESCR0, and the two words of the relevant page table descriptor into register DESCR1 and register DESCR2, respectively. While the three registers DESCR0 – 2 are loaded, the BMAC will check whether a page that is marked local is present in the required local memory. This is done only when the local bit in the segment table is set, by comparing the LI field in the page table with the contents of the PIR register. When the L bit is not set, or the LI field equals 1111, this check is disabled. The result of this check is stored in the ECR register. When the result is negative, i.e., the page is not in the required local memory, the BMAC will not load the page descriptor into its cache.

There is space for a cache of 32 descriptors on the BMAC. Before a new descriptor is loaded, the corresponding TLB slot number must be loaded into the DAR.

After the BMAC is reset, the DAR points to slot 0. The first several slots should be used for the descriptors which contain the system stack, BERR handling code, and data. These 'fixed' descriptors need to be resident before the virtual address system is enabled. It should be noted that regions containing these 'fixed' descriptors must not be flushed.

Translation Look-Aside Buffer Miss
If a word is read and it is not in the cache, the MMU section is also accessed to find its physical address. If the relevant page descriptor is not in the TLB, then the BMAC will assert BERR.

Handling of the Dirty Bit
If any permitted write is made to a page and the dirty bit in the page descriptor in the TLB is not set, then it is set in the TLB and a BERR is generated.

Handling of Illegal Accesses to Memory
If an access is made to a memory location for which the required permission bits are not set, then the BERR is set. The reason is stored in the BMAC, in the error code register. If an access to a non-resident page is made, the same procedure will be followed. This will be done whether the page at fault was the desired page, or a page containing either the segment table or the page table.

Error Reporting
When an error or a page fault is detected by the BMAC during its operation, an error-report containing a description of the problem is stored in the ECR by the BMAC. The BMAC then generates a BERR to the processor.

Cache Only Mode

In cache only mode, the BMAC does no address translation or permission testing. Therefore, the physical address output is not useful since it has no relation to the input logical address. Regions can be used to provide a basis for selective flushing of the cache, where the applied address can either be real or virtual (depending on the application). Accesses with a CPU address space function code (FC2 – 0 = 111) are never cached; virtual addresses are passed through to the physical address bus.

THE BMAC'S PLACE IN THE SYSTEM

It is possible to have one or more BMACs in the system. A system with one processor with an attached BMAC is shown in figure 9. The BMAC handles all the processor requests to memory and completely controls the cache. This is a fairly straight forward solution, as all the DMA requests use real addresses. While this is fairly straight forward for the hardware, the software has to arrange the blocks that have to be DMA'ed into memory to fit into consecutive real addresses, unless the DMA units can handle a scattered real memory.

Virtual I/O Addresses

This system is shown in figure 10. It is much easier for the software to handle, but the address bus onto which the DMA peripherals hang is the bus between the processor and the BMAC (note that all relevant I/O descriptors must be in the BMAC before needed). This is fine for a system with limited modularity, but it is difficult for a system which needs the traditional modularity, e.g., on the card level of a minicomputer. If that type of system is needed, then the configuration given in figure 11 can be used, i.e., with a separate BMAC dedicated to I/O. Here all descriptors are loaded before the start of an I/O action.

I/O units using a BMAC for address translation, must drive the FC3 signal low. The BMAC will then skip the check on access

violations. FC2 – 0 can be used to identify each of up to seven I/O units connected to the MAC (note that FC2 – 0 = 111 is reserved and may not be used for I/O identification).

Multiple MACs Attached to Multiple Processors

This can be accomplished in two ways depending on the system needs. A typical system without local memory is shown in figure 12. This sort of system will allow up to four 68020s to work without seriously disturbing one another. If they work in a MIMD (multi-instruction stream, multi-data stream) mode, i.e., each working on its own process but only one running each section of the operating system at a time. Then typically four 68020s should give the performance of about 3.2 68020s, each running in their own system. This assumes that the processors are memory bound and not I/O bound. Note that although figure 10 shows processors each with their own MAC, these processors could just as easily be I/O units with DMA properties.

Systems with Local Memory

A typical system is shown in figure 13. Systems like this can be used if there is a way to partition the application. This can be done in many ways depending on the complexity of the system and the software. Some applications can be partitioned statically, e.g., a process control system where the tasks can be pre-assigned to a processor. In this case, the local memory can be used for code and process specific data, e.g., stacks.

Of course, applications that are reassigned dynamically can also be made using this system arrangement and the BMAC provides facilities to page in and out of local memories. When the page descriptor associated with a local page or segment (i.e., L bit is set) is loaded into the BMAC, an error status is stored in the ECR register if the LI field does not match the BMAC's PIR register contents. This check is overridden if the LI field is 1111.

The BMAC allows up to fifteen 680xxs, each having its own local memory to cooperate together. In particular, with the requisite system software, it is possible to write application programs that could run unchanged in either a single processor system or a multiprocessor with or without local memory.

The Use of Multiple BMACs Attached To One Processor

In some special cases, there may be a need to attach more than one BMAC to a processor. Normally, this should not be necessary as the cache hit rate should be high enough (greater than 90%) and the TLB hit rate should be better than 99.5%. However, if this is not enough or if a fast response is wanted for interrupts then multiple BMACs can be used. The BMAC will keep all of its output lines in a high impedance state when ASN is high. Therefore, multiple BMACs can be used by gating ASN with another address line(s). Note this address line should not be part of the offset within a page as that will reduce the effect of adding more than one BMAC.

If multiple BMACs are attached to one processor, then all the BMACs have to have identical region descriptor definitions.

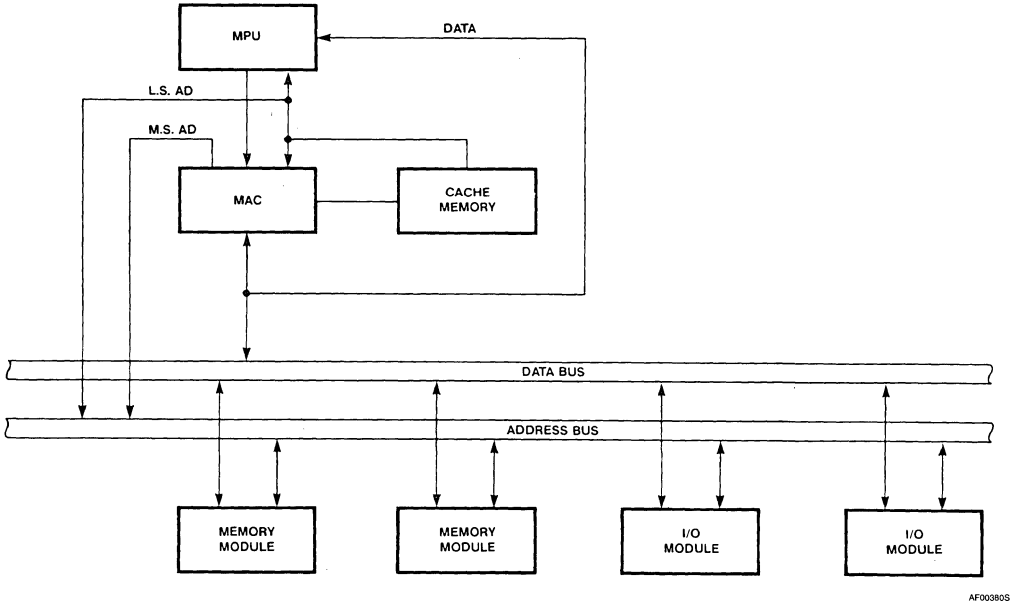


Figure 9. Processor With BMAC, Real I/O

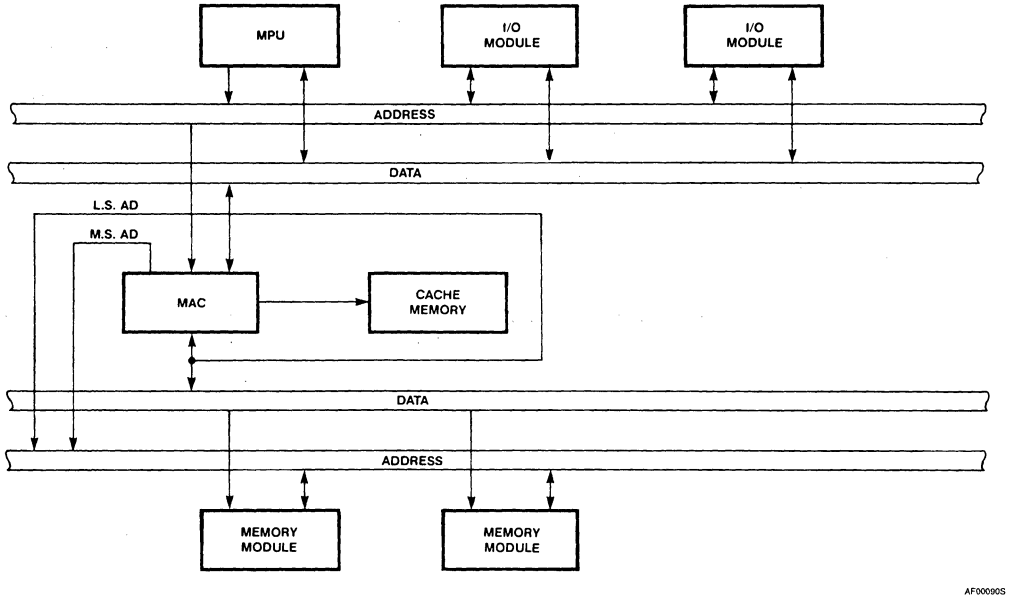


Figure 10. Virtual I/O

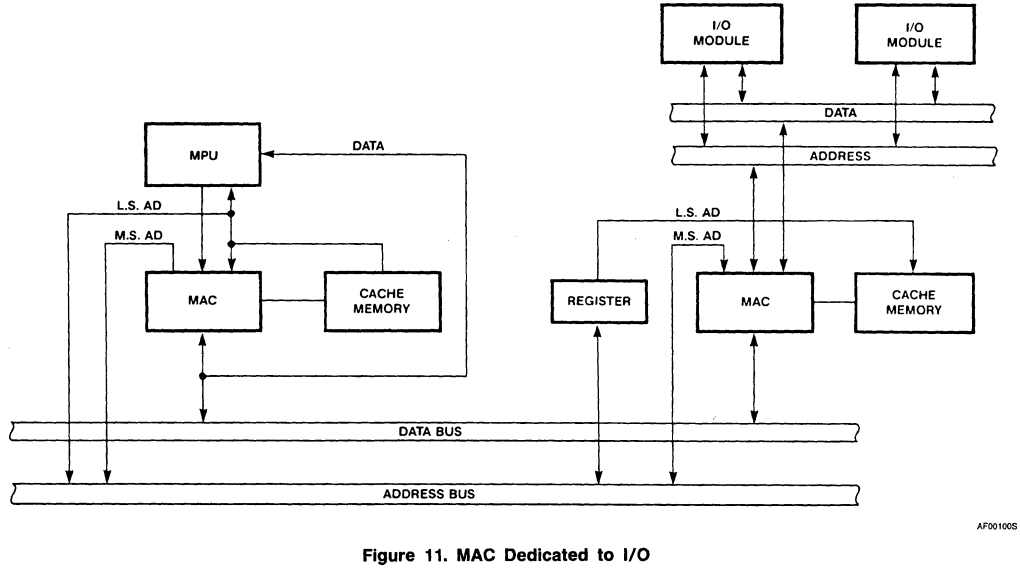


Figure 11. MAC Dedicated to I/O

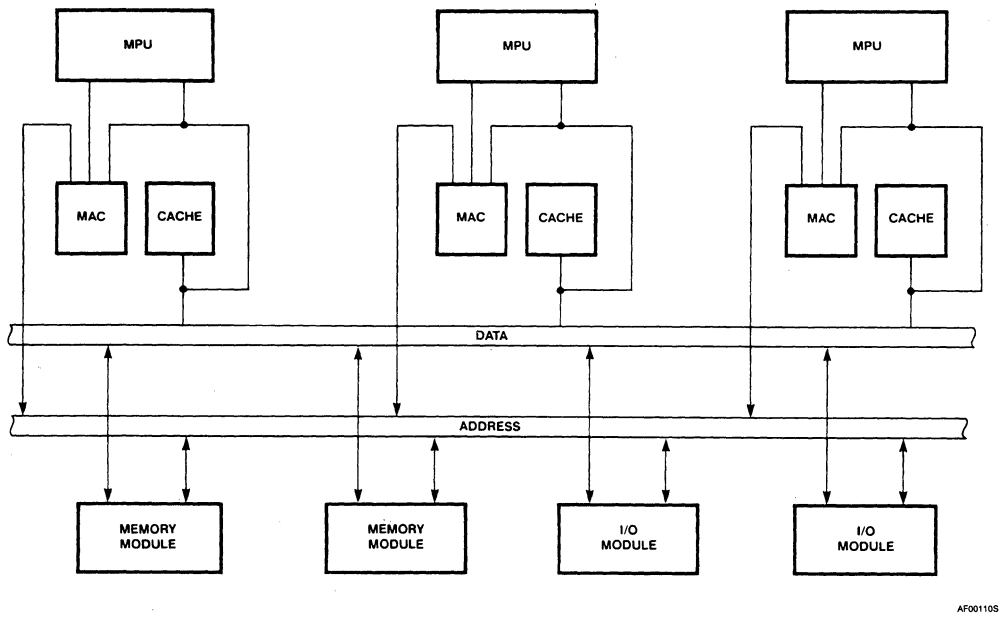


Figure 12. Multiple Processors, No Local Memory

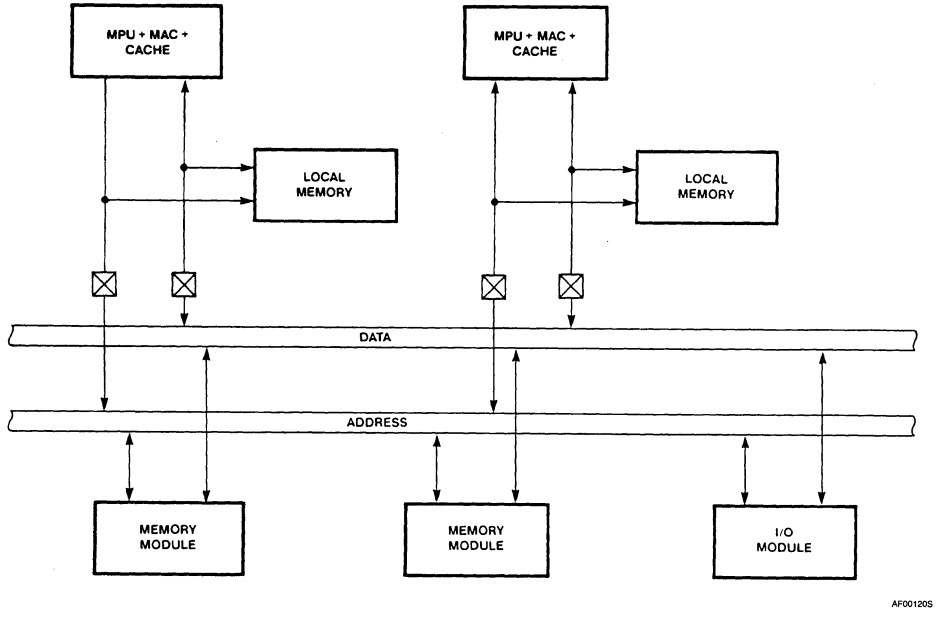


Figure 13. System With Local Memory

PROGRAMMING THE BMAC

On power-up all the registers on the BMAC are reset to zero. This implies that there are no caches attached and all addresses are real, i.e., the BMAC does no translation whatsoever. The BMAC will also be reset by the RESET signal.

Initializing the System

The following registers have to be set.

1. The page definition register
2. The size of segment register
3. The region definition register
4. The BMAC communication area pointer register
5. The size of cache register
6. The memory fetch register
7. The BMAC mode register

The order of setting is irrelevant, except that as the BMAC mode register activates the system it should be set last. In addition the 'fixed' descriptors should be loaded.

Programming on a Context Change

As the BMAC needs only to be given a process context, it does not have to be reloaded on a process change. The only actions necessary are to invalidate the old context. This is done by writing to the relevant flush region register FRRn. This action only invalidates the cache and TLB so as not to confuse the old context with the new context.

Programming for Change of System State

Care should be taken when this is done. In particular, if there are multiple units accessing memory, i.e., processors or I/O units, they should be in a known state before this is done and should be suitably informed when the system state has changed. In other words, it is assumed that the system is normally only the kernel of the operating system and is normally fixed in memory.

This possibility for reconfiguration is only provided for safety and to allow the system to be reconfigured on a module failure.

Programming for I/O

Two philosophies are possible:

1. Real I/O
2. Virtual I/O

In real I/O, all I/O is done in real address space and the BMAC is not involved with the I/O operation. In virtual I/O, system I/O is mapped through one or more BMACs. If BMACs are shared between DMA control units then, seven function codes 0xxx should be shared, one to each of up to seven peripherals. Note that function code 0000 is reserved for the processor and cannot be used for I/O devices. Also note that FC3 is low for I/O transfers. The required descriptors should be in the BMAC. If the associated peripheral controller does not contain a deep FIFO, then required pages and tables should be locked in memory.

If required descriptors are not initially in the BMAC, then care must be taken to ensure their presence before they are needed.

TIMING

Memory accesses initiated by the BMAC will use the same timing as the 68010, i.e., three cycles for read and four for write. The internal timing of the BMAC will be such that when a series of reads have to be done, assuming zero wait states from the memory, then a read will be issued every four clock cycles. Some exceptions to this requirement may be made.

The CWEN signal will have a positive set-up and hold time with respect to CCEN.

ADDRESS FORMATS AND MEMORY ORGANIZATION SUMMARY

Table 4 summarizes the address and data path widths for the BMAC.

The logical address is composed of three fields: segment number, page number and offset within the page. The sizes of these fields are programmable in the BMAC via the page definition register. The possible address compositions are shown in table 5.

BMAC BERR HANDLER PROCESSING

Refer to figure 14 which covers all of the possible BERR handling features. The BERR handler routine first saves any required 68020 registers. The contents of the error code register are then read. When bit 15 = 0, an access/permission/true BERR has occurred. Processing then branches to entry E1, where bits 2 – 6 of the ECR are checked for the type of violation. Access violations are further identified via bits 11 – 14 which give the TLB descriptor values for permissions S,E,R, and W. A jump table contains the entry points for the various violations. These violations are listed in table 6.

The main processing of the BMAC BERR handler occurs when ECR bit 15=1, indicating a descriptor fault. The routine then accesses the 68020 exception stack to obtain three words; the fault address (high/low) and the special status word (see table 7). Note that the routine may have to move this information from the stack in three 16-bit word moves, foregoing the more efficient long-word moves to avoid possible addressing (boundary) problems. The fault address words provide the virtual address of the BERR. These words must be saved in a 68020 data register. This data represents the segment number, page number, and offset within the page of the virtual address. The exact locations of these portions are determined by the user during software set-up.

Table 4. ADDRESS AND DATA PATH WIDTH

VERSION	NUMBER OF BITS		
	Logical ADDR	Physical ADDR	Data
120-Pin	32	28	16

24-BIT LOGICAL ADDRESS

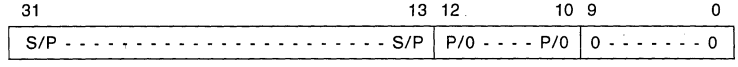


Table 5. MEMORY ORGANIZATION

SEGMENT SIZE (BYTES)	NUMBER OF SEGMENTS	MAX NUMBER OF PAGES PER SEGMENT			
		Page Size 1K Bytes	Page Size 2K Bytes	Page Size 4K Bytes	Page Size 8K Bytes
64M	64	—	—	—	8K
32M	128	—	—	8K	4K
16M	256	—	8K	4K	2K
8M	512	8K	4K	2K	1K
4M	256	4K	2K	1K	512
2M	2K	2K	1K	512	256
1M	4K	1K	512	256	128
512K	8K	512	256	128	64

Table 6. BMAC BERR HANDLER SUBROUTINES FOR VIOLATIONS (E1 ENTRY POINTS)

BITS 6–2	BITS 14–11	SUBROUTINE ENTRY POINT
00000	NA	No error
10000	NA	Page not in required local memory
01000	S,E,R,W	Access permission violation: (Supervisor, Execute, Read, Write)
00010	NA	Illegal BMAC access
11110	NA	BERR on local bus

Table 7. EXCEPTION STACK ORDER FOR 68010 (BUS AND ADDRESS ERROR)

15 0

Program counter (high)	
Program counter (low)	
1000	Vector offset
Special status word	
Fault address (high)	
Fault address (low)	
Unused, reserved	
Data output buffer	
Unused, reserved	
Data input buffer	
Unused, reserved	
Instruction input buffer	
Internal information	

Bit 8 of the special status word, figure 15, indicates if a read/write was being performed when the BERR occurred. Bits 0 – 1 of the ECR contain the region number where the error occurred. This region number is used to select the appropriate region descriptor from which are selected, in turn, the segment table pointer, (first-level table pointer, FLTP) and the segment table length (first-level table length, FLTL). Next, the routine will shift out the segment number from the virtual address.

Bit 7 of the ECR now determines the major processing decision of the routine. If bit 7 = 1, a page descriptor miss has occurred. A series of segment and page checks need to be made, and then the three BMAC descriptor registers will be loaded by the routine. When bit 7 = 0, the dirty bit has not been set on a write. The series of checking and descriptor loading will be bypassed, as they already have been performed. Table 8 gives the possible error processing done by the routine. Both branches refer to the segment descriptors and the page descriptors.

When bit 7 = 1, the segment number (which has been shifted out of the virtual address by the routine) is compared to the number of segments currently valid (segment table length register). If the segment number is greater than the FLTL, a 'segment no. out-of-range' error is in effect, and the routine will branch to the E2 error entry point (see table 6). Otherwise, the segment descriptor is accessed to test the various protection bits. The segment descriptor address will be built by the routine as follows: FLTP + segment number * 8.

If the segment is invalid (bit 31 = 1), processing branches to error entry point E3.

The offset (bit 22) bit denotes whether the pages of a segment start at the top (higher address, O = 1) or bottom (lower address, O not = 1) of the segment's virtual address space. The routine will use the page number (shifted out of the virtual address of the exception stack) and the page length (from the segment descriptor words). If O = 1, the routine will check for the page length being greater than the page number. If O is not = 1, the error checked will be for the page number being greater than the page length. Both error conditions will force a branch to error entry point E4.

Next, the routine will test the contiguous bit (bit 24) in the segment descriptor. If C = 1, the segment is contiguous, i.e., has no page table. The routine will then skip over the page checking code to go on to loading the segment data into the descriptor registers. If C is not = 1, then the routine will perform page processing as follows. It will shift out the page number from the virtual address and get the page length from the segment descriptor. Then, the routine will build the page descriptor address thus: base address from segment descriptor + page number * 4.

Next the page present bit (bit 31) will be tested. If P = 0, the page is not present in memory, and the routine will branch to error entry point E5 to process the error. Otherwise, the routine will proceed with remaining page processing. The access type (from special status word) will then be tested. If a write, the dirty bit and the used bit in the page descriptor will be set. If the BERR interrupted a read access, only the used bit will be set by the routine. Next, the routine will load the three descriptor registers, DESCR0–DESCR2, as follows: DESCR0 = first word of segment descriptor, DESCR1 = first word of page descriptor, and DESCR2 = second word of page descriptor. Two important rules must be followed here:

1. The registers must be loaded in the order of DESCR0, DESCR1, DESCR2.
2. The registers must be loaded atomically. Thus, the 68020 MOVEM instruction must be used.

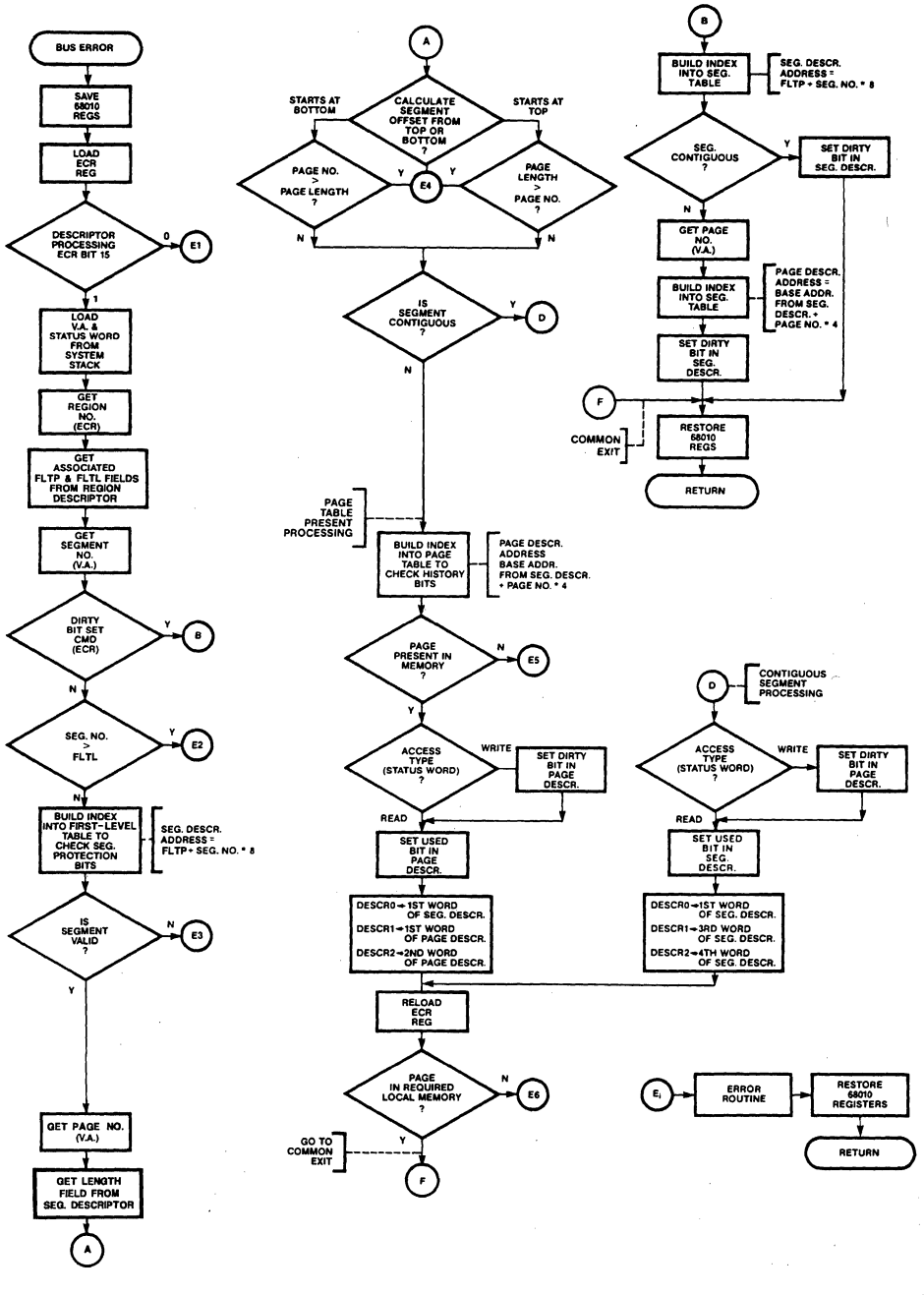
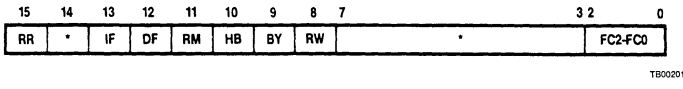


Figure 14. Bus Error (BERR) Handling



RR – Re-run flag; 0 = processor re-run (default), 1 = software re-run.
IF – Instruction fetch to the instruction input buffer.
DF – Data fetch to the data input buffer.
RM – Read-modify-write cycle.
HB – High byte transfer from the data output buffer or to the data input buffer.
BY – Byte transfer flag; HB selects the high or low byte of the transfer register. If BY is clear, the transfer is a word.
RW – Read/write flag; 0 = write, 1 = read.
FC – The function code used during the faulted access.
* These bits are reserved for future use and will be zero when written by the 68020.

Figure 15. Special Status Word Format

Table 8. ERROR ENTRY POINTS

ENTRY	SUBROUTINE ENTRY
E1	Access/other errors - 'good' violations (table 5)
E2	Segment number out of range i.e. greater than number of segments currently valid
E3	Segment not valid
E4	Page number out of range i.e., is in invalid address space of the segment
E5	Page is not present in memory
E6	Page is not in required local memory

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltage	−0.3 to +7.0	V
Input voltage[2]	−0.3 to +7.0	V
Operating temperature range[3]	0 to +70	°C
Storage temperature	55 to +150	°C

The routine will then reload the contents of the ECR. This is required as loading DESCR0 – DESCR2 causes the BMAC to check whether a page that is marked local is present in memory and to set bits 2 – 6 accordingly (bits 2 – 6 = 00001). If the bit has been set, the routine will branch to error entry point E6. Else, the routine will restore its 68020 registers and return.

In the case where the segment is contiguous (C = 1), the routine will set the dirty bit and used bits in the segment descriptor for a write access; the used bit will be set if the BERR occurred during a read access (access type being obtained from the special status word). The routine will then perform activities similar to those taken for loading page descriptors, except for the following differences. DESCR0 will be loaded with the first word of the segment descriptor, DESCR1 will be loaded with the third word of the segment descriptor, and DESCR2 will be loaded with the fourth word of the segment descriptor. The remaining processing is identical to that for paged segments.

The remaining paragraphs describe actions that will be performed when a dirty bit is not set on a write (ECR bit 7 = 0). The routine will build the segment descriptor address thus: FLTP + Segment number * 8. Next, if the segment is contiguous (C = 1, bit 24 of the segment descriptor), the routine will set the dirty bit (bit 25) in the segment descriptor, restore the 68020 registers and return. If the segment has page tables, the page number will be shifted out of the virtual address and will be used to build the page descriptor address, thus: base address from segment descriptor + page number * 4. The routine will then set the dirty bit (bit 25) in the page descriptor, restore the 68020 registers and return.

DC ELECTRICAL CHARACTERISTICS $V_{CC} = 5V \pm 5\%$, $V_{SS} = 0V$; $T_A = 0°C$ to $+70°C$[4,5]

PARAMETER		TEST CONDITIONS	LIMITS Min	LIMITS Max	UNIT
V_{IH}	Input high voltage		2.0	V_{CC}	V
V_{IL}	Input low voltage		GND – 0.75	0.8	V
I_{IN}	Input leakage current	5.25V		20	μA
I_{TSI}	Three-state (off state) input current	2.4V/0.4V		20	μA
V_{OH}	Output high voltage	$I_{OH} = -400\mu A$	2.4		V
V_{OL}	Output low voltage	$I_{OL} = 5.3mA$		0.5	V
P_D	Power dissipation			1.5	W
C_{IN}	Capacitance	$V_{in} = 0V$, $T_A = 25°C$, $f_o = 16MHz$		10	pF

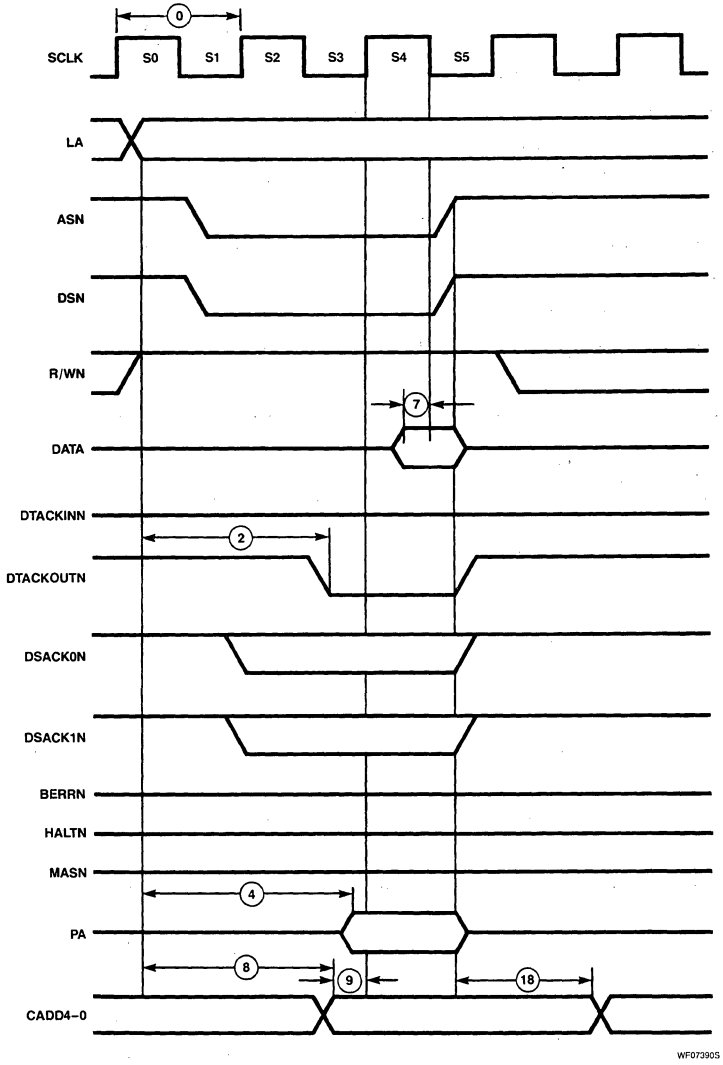
(See notes on next page)

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5]

NO.	FIGURE	PARAMETER	TENTATIVE LIMITS		UNIT
			Min	Max	
0	16	Clock period	80		ns
1	17	CCEN delay after address valid		105	ns
2	16, 17, 25	DTACKOUTN delay after address valid		85	ns
3	17	Cache memory access time		45	ns
4	16, 25	PA delay from address valid		100	ns
5	17	Assertion of BERRN and HALTN from rising edge of clock		14	ns
6	17	BERRN and HALTN set-up time	25		ns
7	16, 17, 25	Data set-up time	10		ns
8	16, 17	CADD0 – 4 valid after address valid		85	ns
9	16, 17	Set-up time of CADD0 – 4 before CCEN	0		ns
10	16, 17	Hold time of CADD0 – 4 after ASN/DSN	10		ns
11	17	DTACKINN delay from assertion of MASN	2 clk periods		
12	17	DTACKINN delay from negation of MASN	1 clk period		
13	17, 25	PA set-up before assertion of MASN	5		ns
14	17, 25	PA hold time after negation of MASN	10		ns
15	25, 15	Data bus float time from negation of ASN/DSN	10	100	ns
16	22, 15	Delay of DSACKsN from DTACKOUTN		11	ns
17	22	DTACKOUTN delay from falling edge of clock		59	ns
18	25	DTACKINN delay from assertion of MASN	0		ns
19	25	MASN delay from clock edge		28	ns
20	25, 27	Synchronous delay from clock edge		40	ns
21	25, 27	Output delay from negation of ASN/DSN		80	ns
22	25	MASN delay from negation of ASN/DSN		94	ns
23	22	Hold time of DSACKsN from edge of clock	10		ns
24	27	CSN delay after address valid		13	ns
25	27	Read register access time from assertion of CSN	185		ns
26	27	BERRN delay from assertion of BERROUTN		15	ns

NOTES:

1. Stresses above those listed under absolute maximum ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other conditions above those indicated in the operation section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltage larger than the rated maximum.
3. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8 and 2.0V as appropriate.



NOTE:
For a 68020 – 68920 interface, the signals followed by an asterisk should be provided externally.

Figure 16. Read Hit

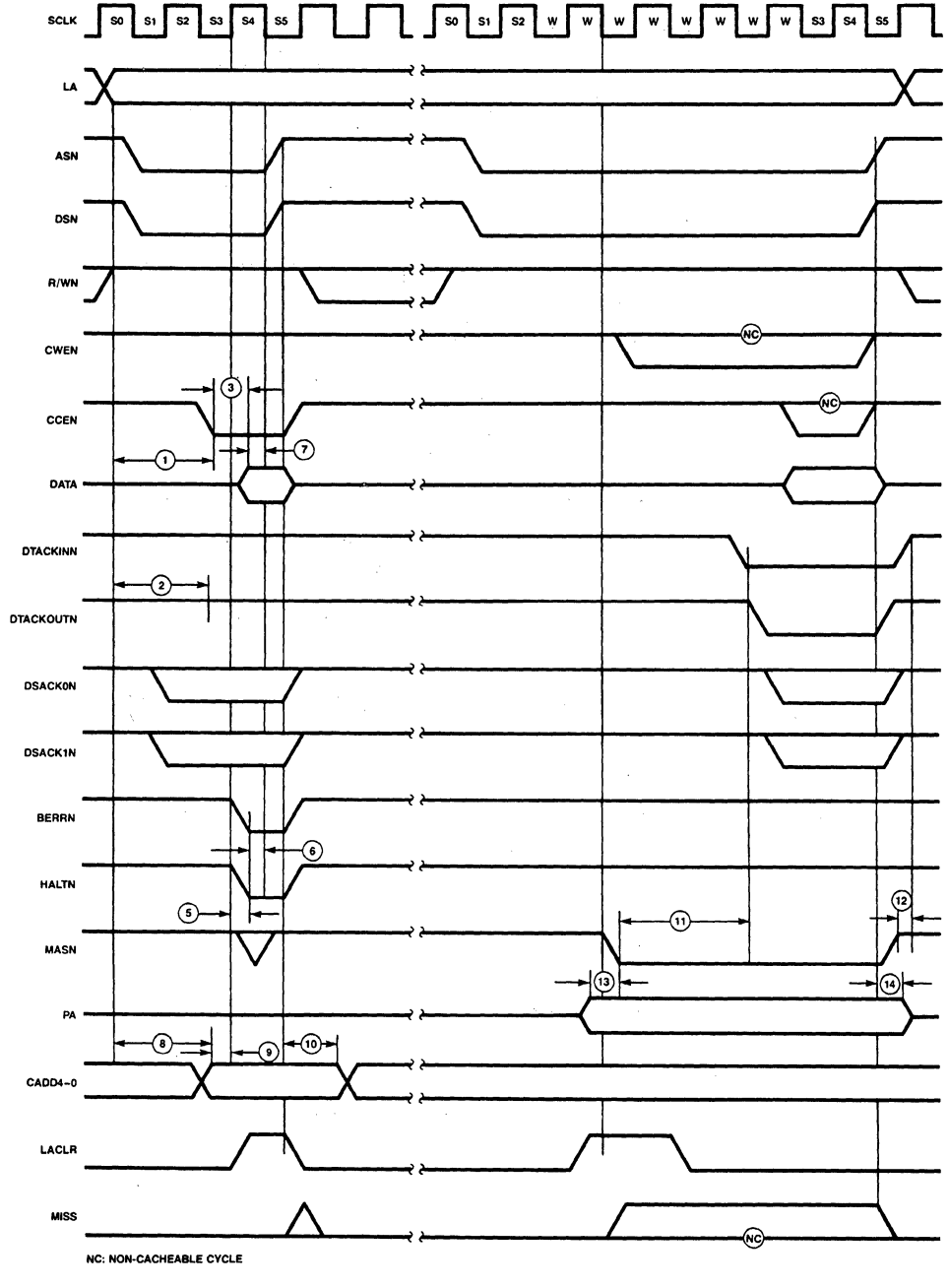


Figure 17. Read MISS Followed By Cache Fill

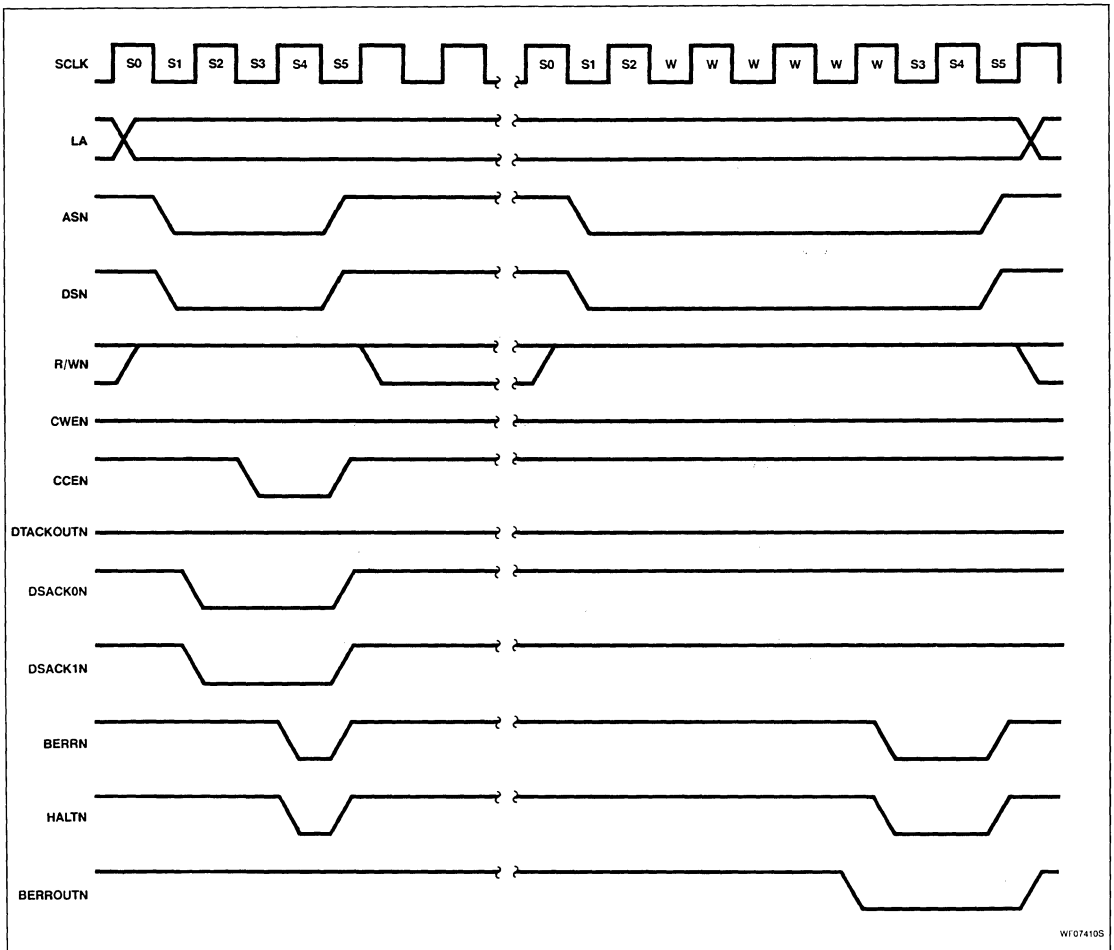


Figure 18. Memory Read Access Violation/TLB MISS

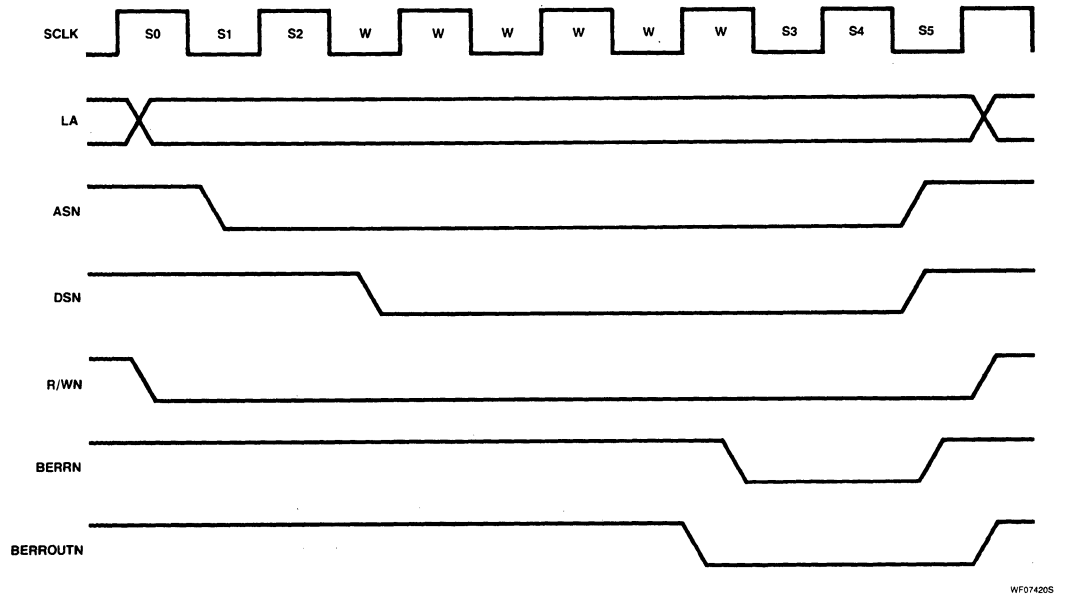
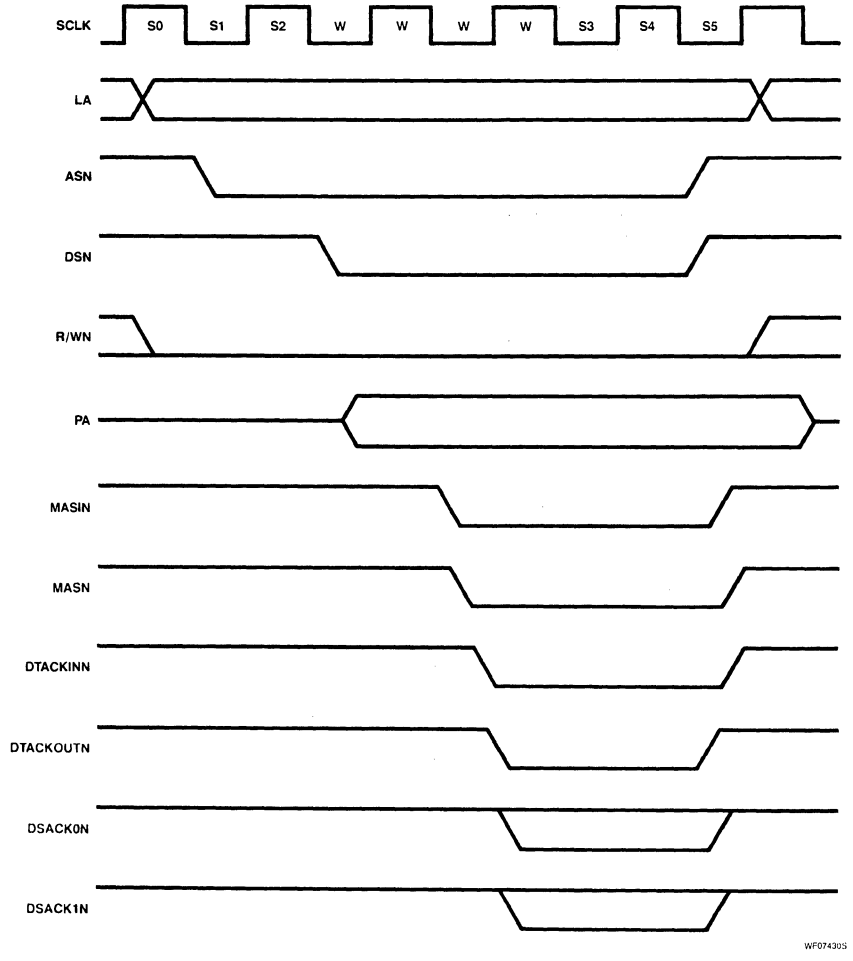


Figure 19. Memory Write TLB MISS/Access Violation/Dirty Not Set



NOTE:
Signal MASIN is the memory address strobe signal generated by BMAC.

Figure 20. Memory Write BMAC Not Enabled

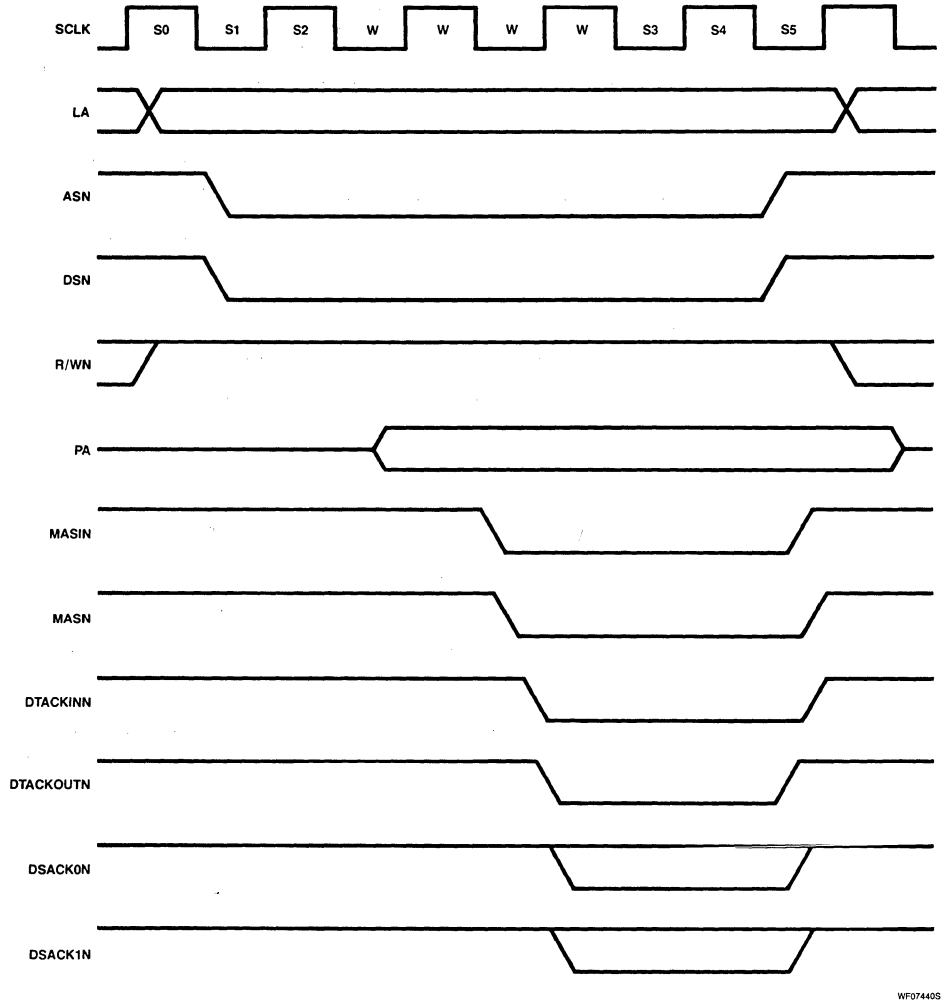


Figure 21. Memory Read BMAC Not Enabled/CPU-Cycle

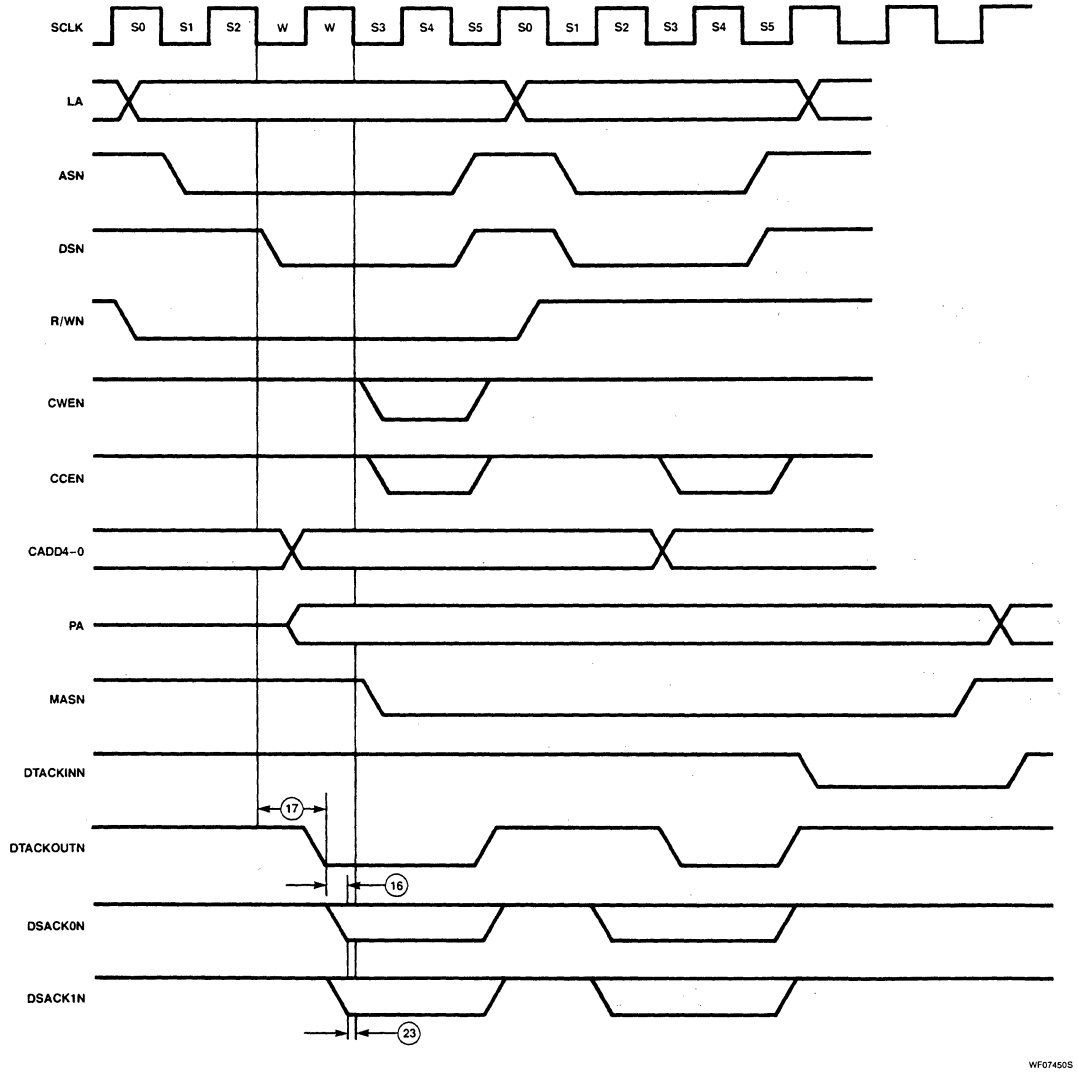


Figure 22. Memory Write Cycle

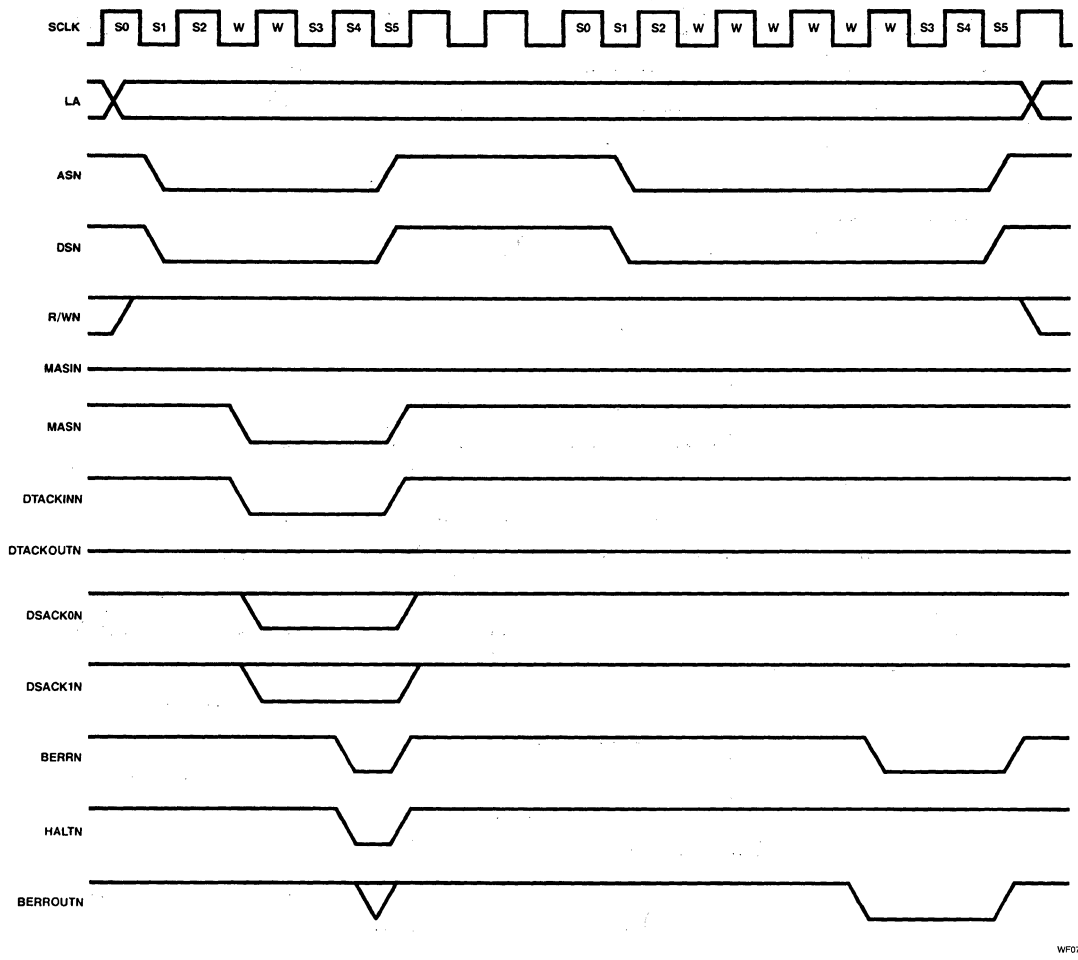


Figure 23. Memory Read Access Violation/TLB MISS Cache Disabled

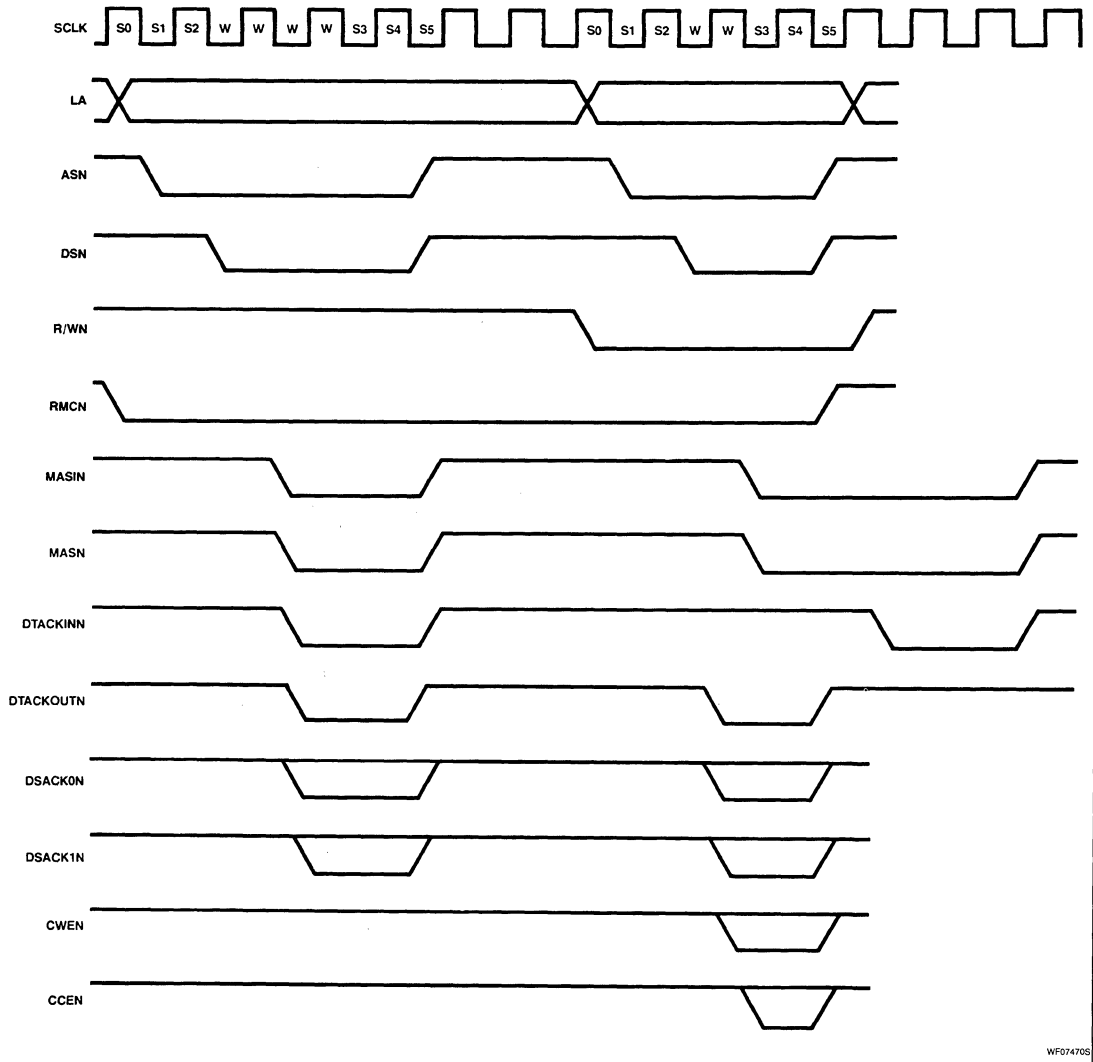


Figure 24. Read Modify Write (Read Must Be Non-Cacheable)

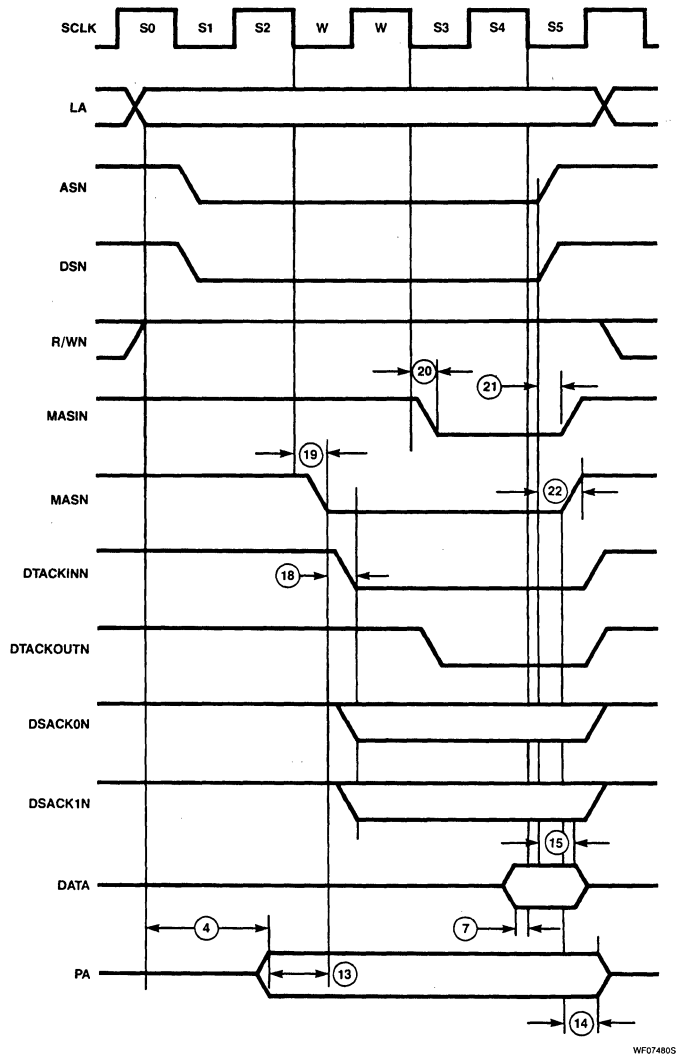


Figure 25. Memory Read Cache Disabled

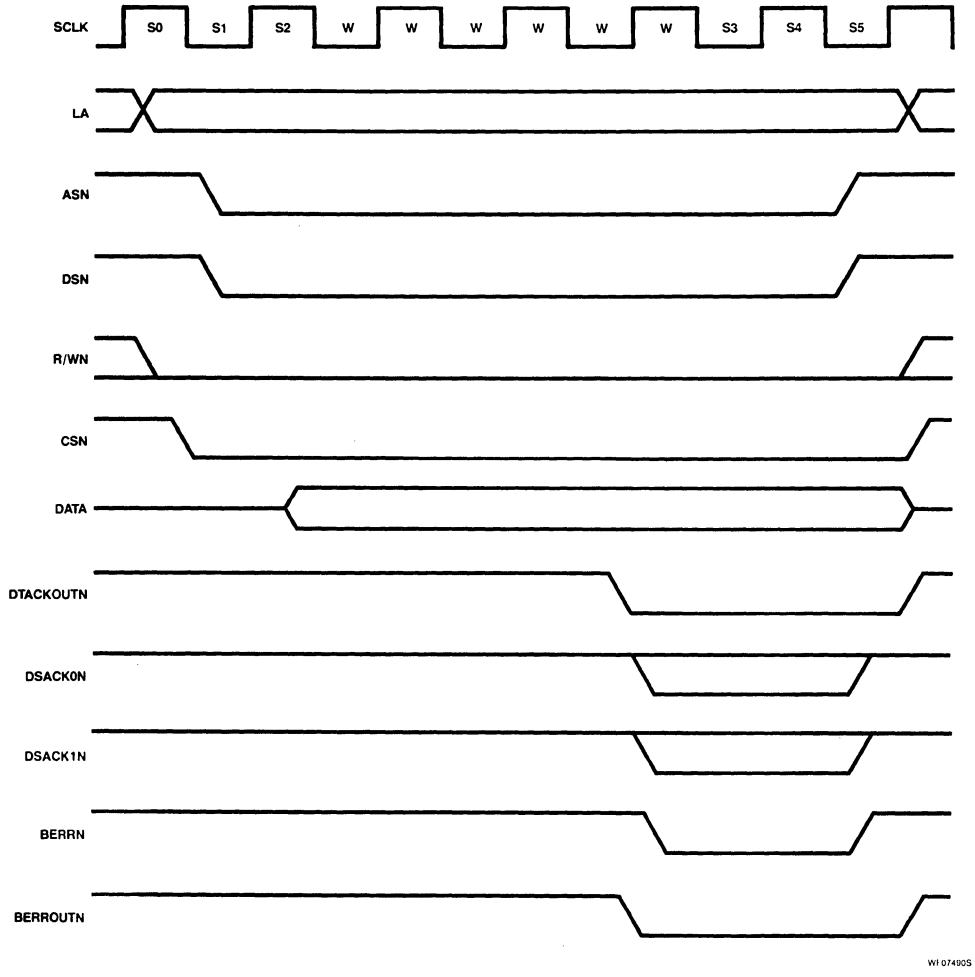


Figure 26. Register Write

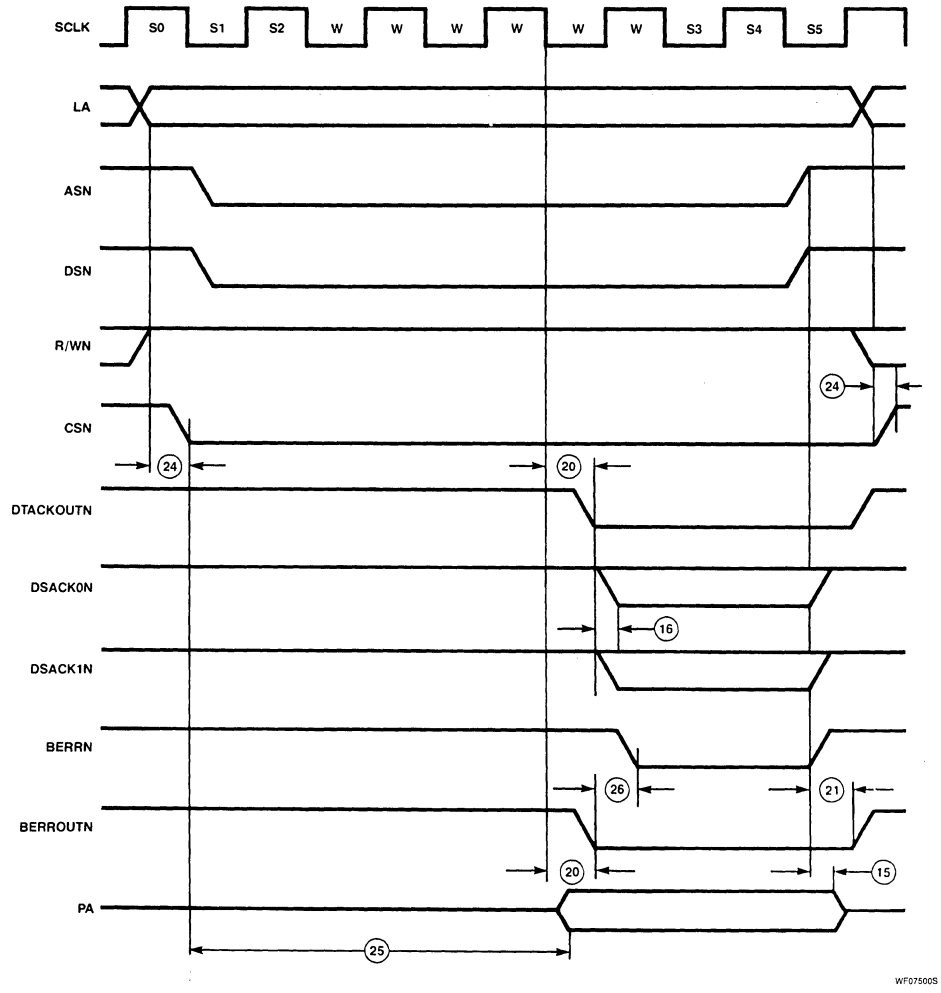


Figure 27. Register Read

Signetics

Microprocessor Products

Section 3
Microcontroller Products

INDEX

January 1986

SCN80 Series
Single Chip 8-Bit Microcontroller

Product Specification

Microprocessor Products

DESCRIPTION

The Signetics SCN80 Series microcontrollers are self-contained, 8-bit processors which contain the system timing, control logic, RAM data memory, ROM program memory (8048/49/50 only), and I/O lines necessary to implement dedicated control functions. All SCN80 Series devices are pin and program compatible, differing only in the size of the on-board program ROM and data RAM, as follows:

TYPE	RAM SIZE	ROM SIZE
SCN8048	64 x 8	1K x 8
SCN8049	128 x 8	2K x 8
SCN8050	256 x 8	4K x 8
SCN8035	64 x 8	—
SCN8039	128 x 8	—
SCN8040	256 x 8	—

Program memory can be expanded externally up to a maximum total of 4K bytes without paging. Data memory can also be expanded externally. I/O capabilities can be expanded using standard devices or the 8243 I/O expander.

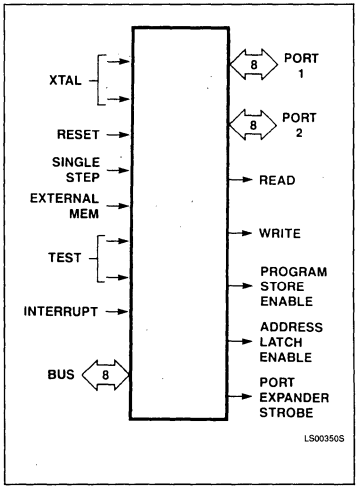
The SCN80 Series processors are designed to be efficient control processors as well as arithmetic processors. They provide an instruction set which allows the user to directly set and reset individual lines within its I/O ports as well as test individual bits within the accumulator. A large variety of branch and table look-up instructions make these processors very efficient in implementing standard logic functions. Also, special attention has been given to code efficiency. Over 70% of the instructions are a single byte long and all others are only 2 bytes long.

An on-chip 8-bit counter is provided which can count, under program control, either internal clock pulses (with a divide by 32 prescaler) or external events. The counter can be programmed to cause an interrupt on terminal count.

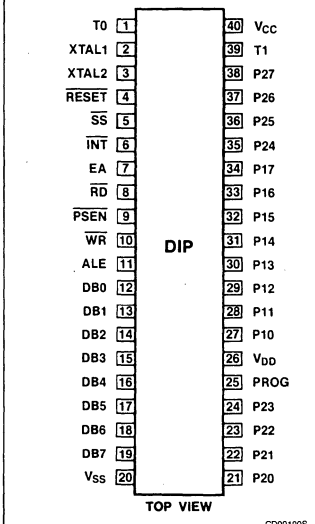
FEATURES

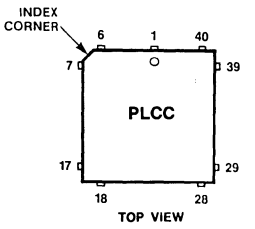
- **8-bit CPU, ROM, RAM, I/O in a 40-pin package**
- **24 quasi bidirectional I/O lines**
- **Two test inputs**
- **Internal counter/timer**
- **Single-level vectored interrupts: external, counter/timer**
- **Over 90 instructions, 70% single byte**
- **1.36μs or 2.5μs instruction cycle, all instructions one or two cycles**
- **Expandable memory and I/O**
- **Low voltage standby**
- **TTL compatible inputs and outputs**
- **Single +5V power supply**

LOGIC SYMBOL



PIN CONFIGURATION

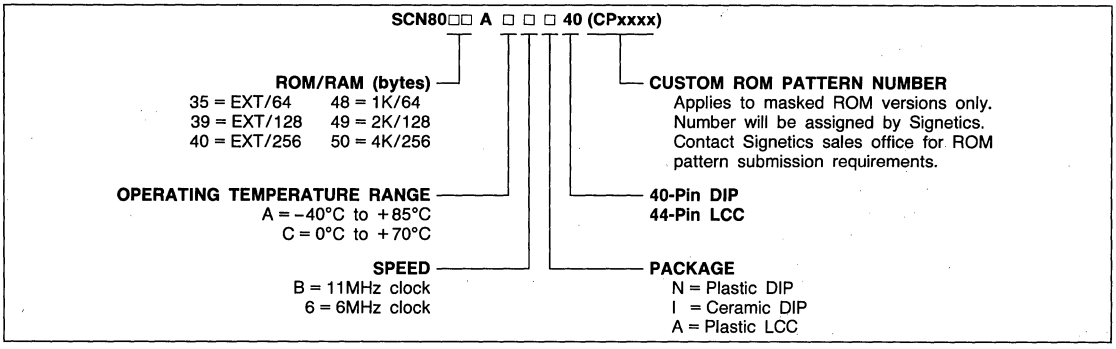




Pin	Function	Pin	Function
1	NC	23	NC
2	T0	24	P20
3	XTAL1	25	P21
4	XTAL2	26	P22
5	$\overline{RESET}$	27	P23
6	$\overline{SS}$	28	PROG
7	$\overline{INT}$	29	V_{DD}
8	EA	30	P10
9	$\overline{RD}$	31	P11
10	$\overline{PSEN}$	32	P12
11	$\overline{WR}$	33	P13
12	NC	34	NC
13	ALE	35	P14
14	DB0	36	P15
15	DB1	37	P16
16	DB2	38	P17
17	DB3	39	P24
18	DB4	40	P25
19	DB5	41	P26
20	DB6	42	P27
21	DB7	43	T1
22	V_{SS}	44	V_{CC}

Single Chip 8-Bit Microcontroller — SCN80 Series

ORDERING CODE



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	Circuit ground potential.
V_{DD}	26	29	I	Low power standby.
V_{CC}	40	44	I	**Main Power Supply:** +5V during operation.
PROG	25	28	O	Output strobe for 8243 I/O expander.
P10 – P17	27 – 34	30 – 33, 35 – 38	I/O	**Port 1:** 8-bit quasi-bidirectional port.
P20 – P27	21 – 24, 35 – 38	24 – 27, 39 – 42	I/O	**Port 2:** 8-bit quasi-bidirectional port. P20-23 contain the four high-order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 8243.
DB0 – DB7	12 – 19	14 – 21	I/O	**Data Bus:** True bidirectional port which can be written or read synchronously using the $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched. Contains the eight low-order program counter bits during an external program memory fetch and receives the addressed instruction under the control of $\overline{PSEN}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{RD}$ and $\overline{WR}$.
T0	1	2	I	Input pin testable using the conditional transfer instructions JT0 and JNT0. T0 and be designated as a clock output using the ENT0 CLK instruction.
T1	39	43	I	Input pin testable using the JT1 and JNT1 instructions. Can be designated the timer/counter input using the STRT CNT instruction.
XTAL1	2	3	I	**Crystal 1:** One side of the crystal input for internal oscillator. Also input for external source (non-TTL V_{IH}).
XTAL2	3	4	I	**Crystal 2:** Other side of crystal input.
$\overline{INT}$	6	7	I	**Interrupt:** Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset. Also testable with conditional jump instruction. Interrupt must remain low for at least three machine cycles for proper operation.
$\overline{RESET}$	4	5	I	**Reset:** Used to initialize the microcomputer. Active low. Internal pullup ~75KΩ. During program verification the address is latched by a "0" to "1" transition on $\overline{RESET}$ and the data at the addressed location is output on BUS.
$\overline{RD}$	8	9	O	**Read:** Output strobe activated during a bus read. Can be used to enable data onto the bus from an external device. Used as a read strobe to external data memory.
$\overline{WR}$	10	11	O	**Write:** Output strobe during a bus write. Used as write strobe to external data memory.
ALE	11	13	O	**Address Latch Enable:** Occurs once during each cycle and is useful as a clock output. The negative edge of ALE strobes address into external data and program memory.
$\overline{PSEN}$	9	10	O	**Program Store Enable:** Output occurs only during a fetch to external program memory.
$\overline{SS}$	5	6	I	**Single Step:** Can be used in conjunction with ALE to "single step" the processor through each instruction.
EA	7	8	I	**External Access:** Forces all program memory fetches to reference external memory. Useful for emulation and debug, and essential for testing and program verification.

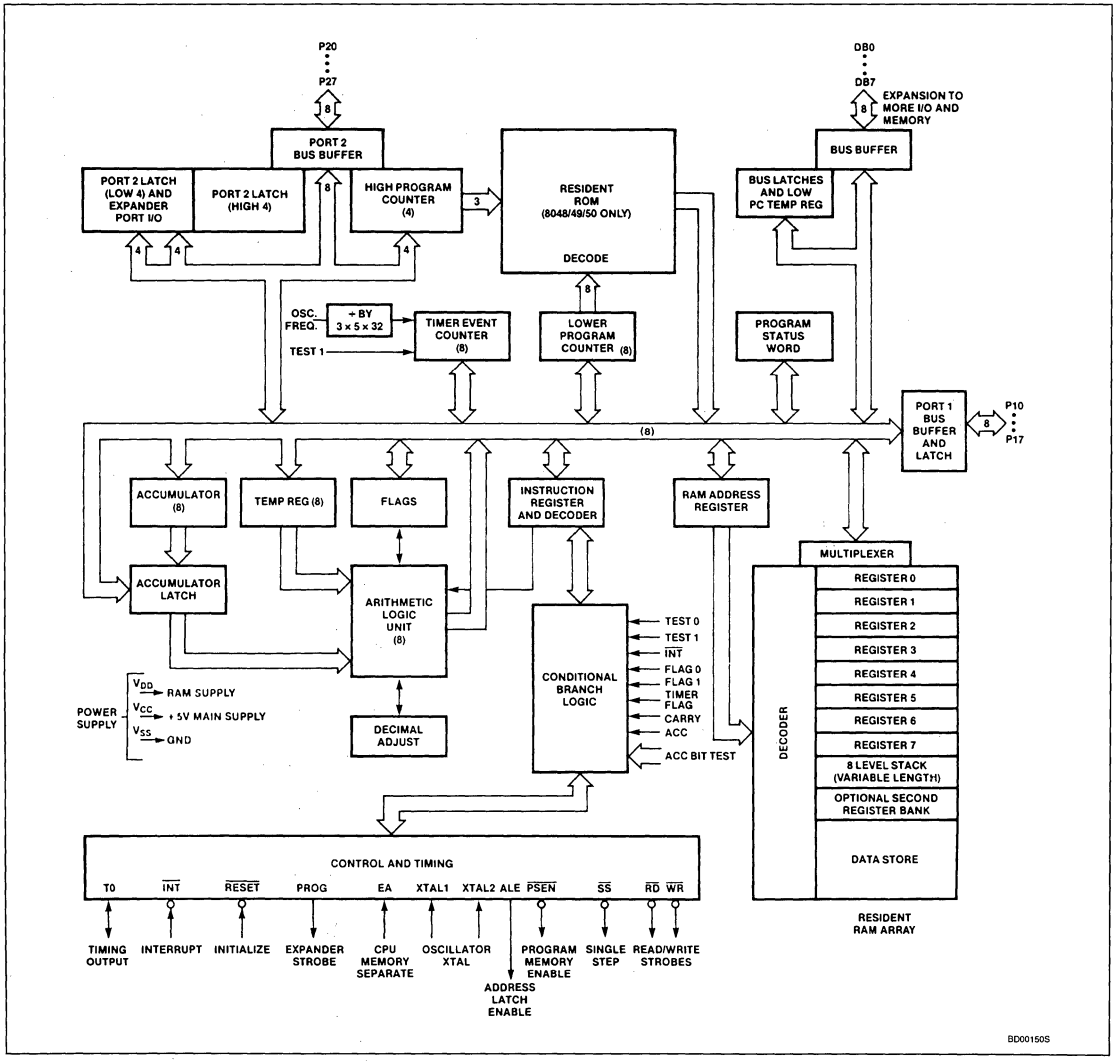
NOTE:

1. Each pin on these ports can be assigned, under program control, to be an input or an output. A pin is designated as an input by writing a logic "1" to the pin. $\overline{RESET}$ sets all pins to the input mode. Each pin has an internal pullup of approximately 50kΩ.

FUNCTIONAL DESCRIPTION

The following is a general functional description of the SCN80 Series microcomputers. Refer to the block diagram.

BLOCK DIAGRAM



PROGRAM MEMORY

Resident program memory consists of up to 4K bytes of ROM. The program memory is divided into pages of 256 bytes each. As shown in the memory map, figure 1, program memory is also divided into two 2048-byte banks, MB0 and MB1. 4096 bytes can be addressed directly. If more memory is required, an I/O port can be used to address locations over 4095.

There are three locations in program memory of special importance. These locations contain the first instruction to be executed upon the occurrence of one of three events.

LOCATION	EVENT
0	Activation then deactivation of the $\overline{\text{RESET}}$ line.
3	Activation of the $\overline{\text{INT}}$ line when the external interrupt is enabled.
7	An overflow of the timer/counter if the T/C interrupt is enabled.

DATA MEMORY

Resident data memory, as shown in figure 2, consists of up to 256 bytes of RAM. All locations are indirectly addressable by either of two RAM pointer registers at locations 0 and 1. The first eight locations of RAM (0–7) are designated as working registers and are directly addressable by several instructions.

By selecting register bank 1, RAM locations 24–31 become the working registers, replacing those in register bank 0 (0–7).

RAM locations 8–23 are designated as the stack. Two locations (bytes) are used per CALL, allowing nesting of up to eight subroutines.

If additional RAM is required, up to 256 bytes may be added and addressed directly using the MOVX instructions. If more RAM is required an I/O port can be used to select one (256-byte) bank of external memory at a time.

PROGRAM COUNTER AND STACK

The Program Counter (PC) is a 12-bit counter/register that points to the location from which the next instruction is to be fetched. The 8048 and 8049 will automatically address external memory when the boundary of their internal memory is exceeded. All processors access external memory if EA is high.

An interrupt or CALL to a subroutine causes the contents of the program counter to be stored in one of the 8 register pairs of the program counter stack. The pair to be used is determined by a 3-bit stack pointer which is part of the Program Status Word (PSW). Data RAM locations 8 through 23 are available as stack registers and are used to store the program counter and 4 bits of PSW. The stack pointer, when initialized to 000, points to RAM locations 8 and 9. The first subroutine jump or interrupt results in the program counter contents being transferred to locations 8 and 9 of the RAM array. The stack pointer is then incremented by one to point to locations 10 and 11 in anticipation of another CALL. Nesting of subroutines within subroutines can continue up to eight times without overflowing the stack. If overflow does occur the deepest address stored (location 8 and 9) will be overwritten and lost since the stack pointer overflows from 111 to 000. It also underflows from 000 to 111.

The end of a subroutine, which is signalled by a return instruction (RET or RETR), causes the stack pointer to be decremented and the contents of the resulting register pair to be transferred to the program counter.

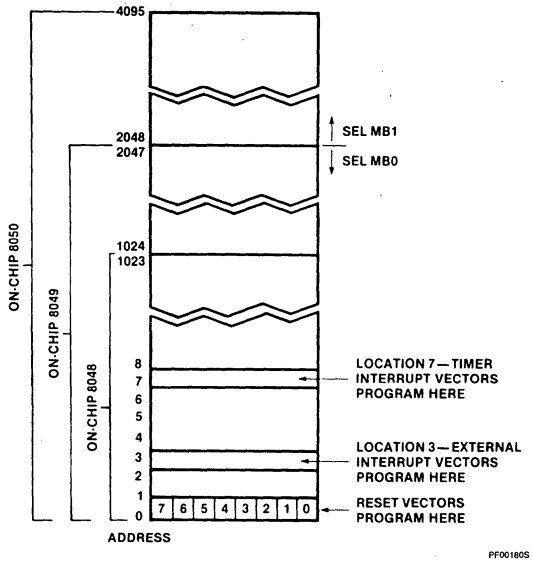


Figure 1. Program Memory Map

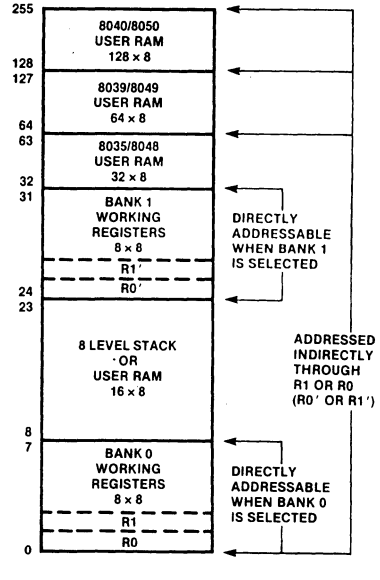


Figure 2. Data Memory Map

OSCILLATOR AND CLOCK

The processor contains its own internal oscillator and clock driver. A crystal, inductor, or external pulse generator may be used to determine the oscillator frequency (see figure 3). The output of the oscillator is divided by three and can be output on the T0 pin by executing the ENT0 CLK instruction. This CLK signal is divided by five to define a machine (instruction) cycle. It is available on pin 11 as ALE.

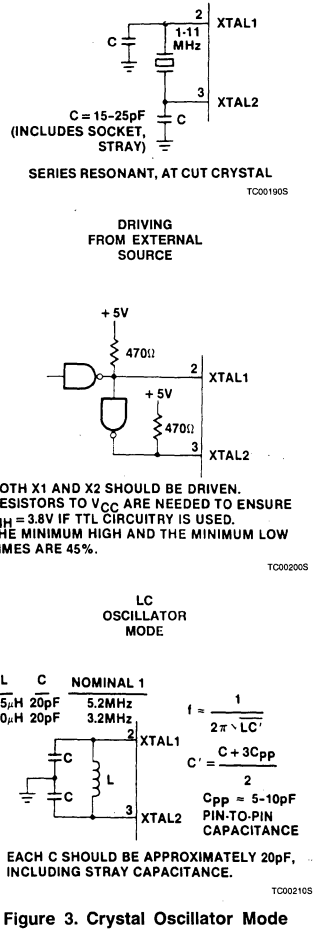


Figure 3. Crystal Oscillator Mode

TIMER/EVENT COUNTER

An internal counter is available which can count either external events or machine cycles (÷ 32). The machine cycles are divided by 32 before they are input to the 8-bit counter. External events are input directly to the counter. The maximum frequency that can be counted is one third of the frequency of the cycle counter. The minimum positive duty cycle that can be detected is 0.2 t_{CY}. The counter is under program control and can be made to generate an interrupt to the processor when it overflows.

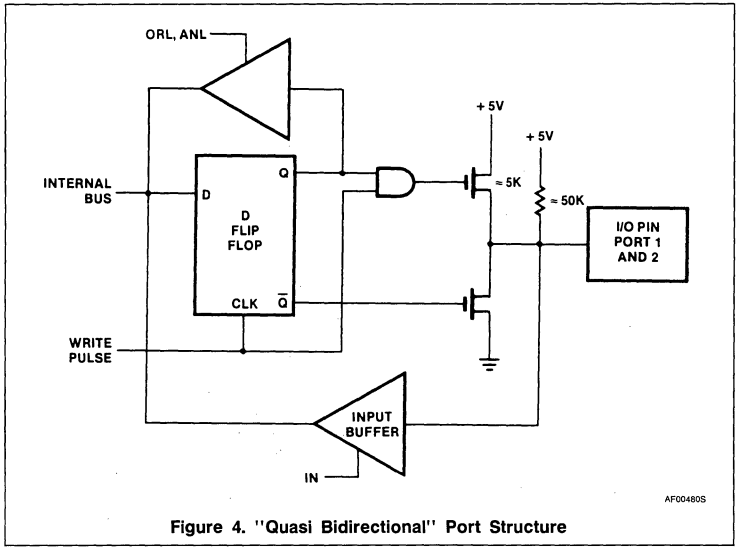


Figure 4. "Quasi Bidirectional" Port Structure

INTERRUPT

An interrupt may be generated by either an external input ($\overline{INT}$, pin 6) or the overflow of the internal counter, when enabled. In either case, the processor completes execution of the present instruction and then does a CALL to the interrupt service routine. After service, a RETR instruction restores the machine to the state it was prior to the interrupt. The external interrupt has priority over the internal interrupt.

INPUT/OUTPUT

The processor has 27 lines which can be used for input or output functions. These lines are grouped as 3 ports of 8 lines each which serve as either inputs, outputs or bidirectional ports and 3 "test" inputs which can alter program sequences when tested by conditional jump instructions.

Ports 1 and 2

Ports 1 and 2 are each 8 bits wide and have identical characteristics. Data written to these ports is statically latched and remains unchanged until rewritten. As input ports these lines are non-latching, i.e., inputs must be present until read by an input instruction. Inputs are fully TTL compatible and outputs will drive one standard TTL load.

The lines of ports 1 and 2 are called quasi-bidirectional because of a special output circuit structure which allows each line to serve as an input, an output, or both even though outputs are statically latched. Figure 4 shows the circuit configuration. Each line is continuously pulled up to +5V through a resistive device of relatively high impedance (~50K). This pullup is sufficient to provide the source current for a TTL high level yet can be pulled low by a standard TTL gate thus allowing the same pin to be used for both input and output. To provide fast switching times in a "0" to "1" transition a relatively low impedance device (~50KΩ) is switched in momentarily (~500ns) whenever a "1" is written to the line. When a "0" is written to the line, a low impedance (~3000Ω) device overcomes the light pullup and provides TTL current sinking capability.

Since the pulldown transistor is a low impedance device a "1" must first be written to any line which is to be used as an input. Reset initializes all lines to the high impedance "1" state. This structure allows input and output on the same pin and also allows a mix of input lines and output lines on the same port. The quasi-bidirectional port in combination with the ANL and ORL logical instructions provide an efficient means for handling single line inputs and outputs within an 8-bit processor.

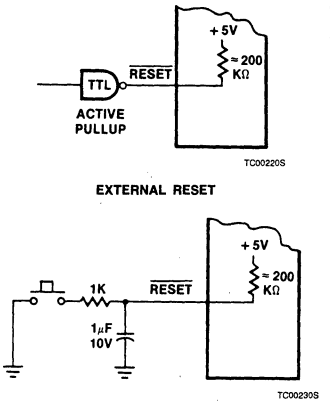


Figure 5. Reset Circuits

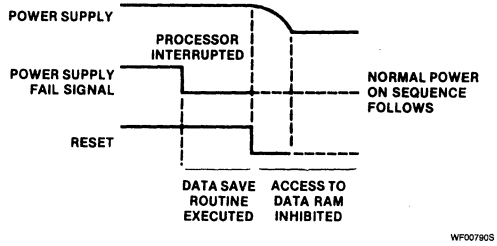


Figure 6. Power Down Sequence

BUS

BUS is also an 8-bit port which is a true bidirectional port with associated input and output strobes. If the bidirectional feature is not needed, BUS can serve as either a statically latched output port or non-latching input port. Input and output lines on this port cannot be mixed.

As a static port, data is written and latched using the OUTL instruction and input using the INS instruction. The INS and OUTL instructions generate pulses on the corresponding $\overline{RD}$ and $\overline{WR}$ output strobe lines; however, in the static port mode they are generally not used. As a bidirectional port, the MOVX instructions are used to read and write to the port. A write to the port generates a pulse on the $\overline{WR}$ output line and output data is valid at the trailing edge of $\overline{WR}$. A read of the port generates a pulse on the $\overline{RD}$ output line and input data must be valid at the trailing edge of $\overline{RD}$. When not being written or read, the BUS lines are in a high impedance state.

Test and INT inputs

Three pins serve as inputs and are testable with the conditional jump instruction. These are T0, T1, and $\overline{INT}$. These pins allow inputs to cause program branches without the necessity to load an input port into the accumulator. The T0, T1, and $\overline{INT}$ pins have other possible functions as well.

RESET INPUT

The reset input provides a means for initialization for the processor. This Schmitt-trigger input has an internal pullup resistor which in combination with an external 1μF capacitor provides an internal reset pulse of sufficient length to guarantee all circuitry is reset. If the reset pulse is generated externally, the reset pin must be held at ground (0.5V) for at least 10 milliseconds after the power supply is within tolerance. Only five machine cycles (12.5μs @ 6MHz) are required if power is already on and the oscillator has stabilized. Typical circuitry is shown in figure 5.

SINGLE STEP

By proper control of the $\overline{SS}$ line, the mircrocomputer can be made to execute one instruction and then pause or wait until the single step switch is activated again.

POWER DOWN MODE

The SCN80 Series devices permit power to be removed from all but the data RAM array for low power standby operation. In the power down mode the contents of data RAM can be maintained while drawing typically 5% of normal operating power.

V_{CC} serves as the 5V supply pin for the bulk of the circuitry while the V_{DD} pin supplies only the RAM array. In normal operation both pins are at + 5V. In standby, V_{CC} is at ground and only V_{DD} is maintained at its specified voltage. Applying $\overline{RESET}$ to the processor through the $\overline{RESET}$ pin inhibits any access to the RAM by the processor and guarantees that RAM cannot be inadvertently altered as power is removed from V_{CC}.

A typical power down sequence occurs as shown in figure 6.

INSTRUCTION SET

The SCN80 Series instruction set consists of over 90 one and two byte instructions (see table 1). Program code efficiency is high because: (1) working registers and program variables are stored in RAM, which require only one byte to address and (2) program memory is divided into pages of 256 bytes each, which means that branch destination addresses require one byte.

The instruction set efficiently manipulates and tests bits in addition to performing logical and arithmetic operations upon and the testing of bytes. A set of move instructions operates indirectly upon either RAM or ROM, which permits efficient access of pointers and data tables. The indirect jump instruction performs a multi (up to 256) way branch upon the content of the accumulator to addresses stored in a lookup table. The "decrement register and jump if not zero" instruction saves a byte every time it is used versus using separate increment and test instructions.

The on-chip counter enables either external events or time to be counted off-line from the main program. The processor can either test the counter (under program control) or cause its overflow to generate an interrupt. These features are highly desirable for real time applications. See table 2 for instruction timing.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
Input voltages with respect to V_{SS}[3]	−0.5 to +7	V
Power dissipation	1.5	W

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to 70°C, $V_{CC} = V_{DD} = 5V \pm 10\%$, $V_{SS} = 0V$[4,5,6]

PARAMETER		TEST CONDITIONS[7]	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage All except XTAL1, XTAL2		−0.5		0.8	V
V_{IL1}	XTAL1, XTAL2		−0.5		0.6	V
V_{IH}	Input high voltage All except $\overline{RESET}$, XTAL1, XTAL2		2.0		V_{CC}	V
V_{IH1}	$\overline{RESET}$, XTAL1, XTAL2		3.8		V_{CC}	V
V_{OL}	Output low voltage	$I_{OL} = 2.0mA$			0.4	V
V_{OH}	Output high voltage All except BUS	$I_{OH} = -125\mu A$	2.4			V
	BUS	$I_{OH} = -400\mu A$	2.4			V
I_{IL}	Input leakage current Port 1, Port 2, EA, $\overline{SS}$	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$			−500	μA
	T1, $\overline{INT}$	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$			±10	μA
I_{OL}	Output leakage current BUS, T0 (high inpedance state)	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$			±10	μA
I_{DD}	Standby supply current 8035/8048	$\overline{RESET} \leq V_{IL}$ All inputs = 0V			2.5	mA
	8039/8049	$V_{CC} = 0V$			4.5	mA
	8040/8050				8.5	mA
$I_{DD} + I_{CC}$	Total supply current 8035/8048	$\overline{RESET} \leq V_{IL}$		45	80	mA
	8039/8049			50	95	mA
	8040/8050			60	110	mA
V_{DD}	Standby power supply		2.2			V
$T_A = -40$ to 85°C, Automotive temperature range[8]						
V_{IH}	Input high voltage All except XTAL1 and XTAL2		2.2			V
I_{IL}	Input current Port 1, Port 2, EA, $\overline{SS}$				−750	μA
I_{DD}	Standby supply current 8035/8048				3.75	mA
	8039/8049				6.75	mA
	8040/8050				12.75	mA
$I_{CC} + I_{DD}$	Total supply current 8035/8048				90	mA
	8039/8049				105	mA
	8040/8050				120	mA

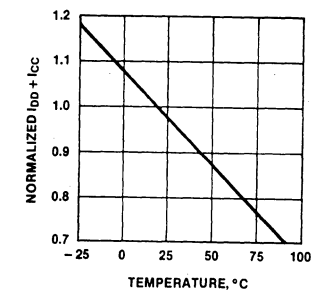
3

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = V_{DD} = 5V \pm 10\%$, $V_{SS} = 0V$[4,5,6]

PARAMETER		TEST CONDITIONS[7]	11 MHz VERSIONS		6 MHz VERSIONS		UNIT
			Min	Max	Min	Max	
(Refer to figures 7, 8 and 9)							
t_{LL}	ALE pulse width		150		400		ns
t_{AL}	Address set-up to ALE		70		150		ns
t_{LA}	Address hold from ALE		50		80		ns
t_{CC}	Control pulse width ($\overline{PSEN}$, $\overline{RD}$, $\overline{WR}$)		300		700		ns
t_{DW}	Data set-up before $\overline{WR}$		250		500		ns
t_{WD}	Data hold after $\overline{WR}$		40		120		ns
t_{CY}	Cycle time		1.36	3.75	2.5	15.0	μs
t_{DR}	Data hold		0	100	0	200	ns
t_{RD}	$\overline{PSEN}$, $\overline{RD}$ to data in			200		500	ns
t_{AW}	Address set-up to $\overline{WR}$		200		230		ns
t_{AD}	Address set-up to data in			400		950	ns
t_{AFC}	Address float to $\overline{RD}$, $\overline{PSEN}$		−10		0		ns
t_{CA}	Control pulse to ALE		10		10		ns
(Refer to figure 10)							
t_{CP}	Port control set-up before falling edge of PROG		100		110		ns
t_{PC}	Port control hold after falling edge of PROG		60		130		ns
t_{PR}	PROG to time P2 input must be valid			650		810	ns
t_{DP}	Output data set-up time		200		250		ns
t_{PD}	Output data hold time		20		65		ns
t_{PF}	Input data hold time		0	150	0	150	ns
t_{PP}	PROG pulse width		700		1200		ns
t_{PL}	Port 2 I/O data set-up		250		350		ns
t_{LP}	Port 2 I/O data hold		20		150		ns

NOTES:

1. Stresses above those listed under absolute maximum ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on + 150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maximum.
4. Paramotors are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground (V_{SS}). For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical ssupply voltages and typical processing parameters.
7. Control outputs: $C_L = 80pF$
   Bus outputs: $C_L = 150pF$
   $t_{CY} = 1.36\mu s$ for 11 MHz versions
   $t_{CY} = 2.5\mu s$ for 6 MHz versions
8. Where no specification is shown, the commercial temperature range specification applies.



NORMALIZED TOTAL SUPPLY CURRENT vs. TEMPERATURE (TYPICAL)

TIMING DIAGRAMS

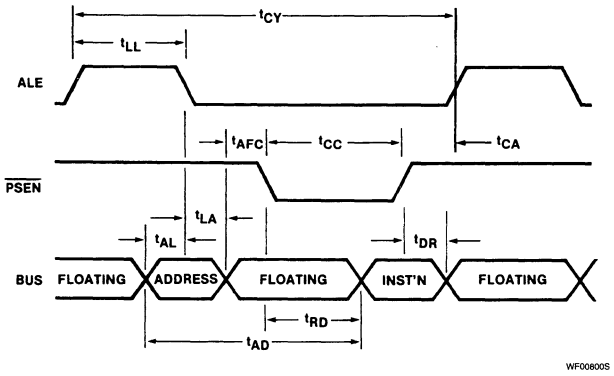


Figure 7. Instruction Fetch from External Program Memory

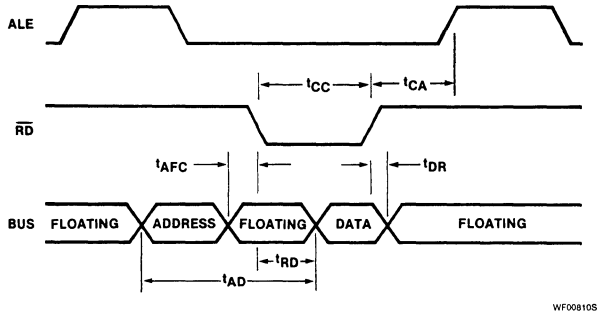


Figure 8. Read from External Data Memory

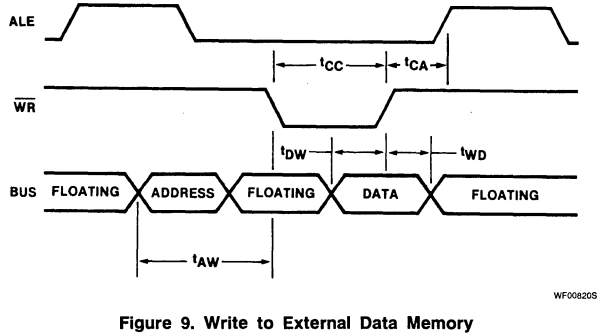


Figure 9. Write to External Data Memory

TIMING DIAGRAMS (Continued)

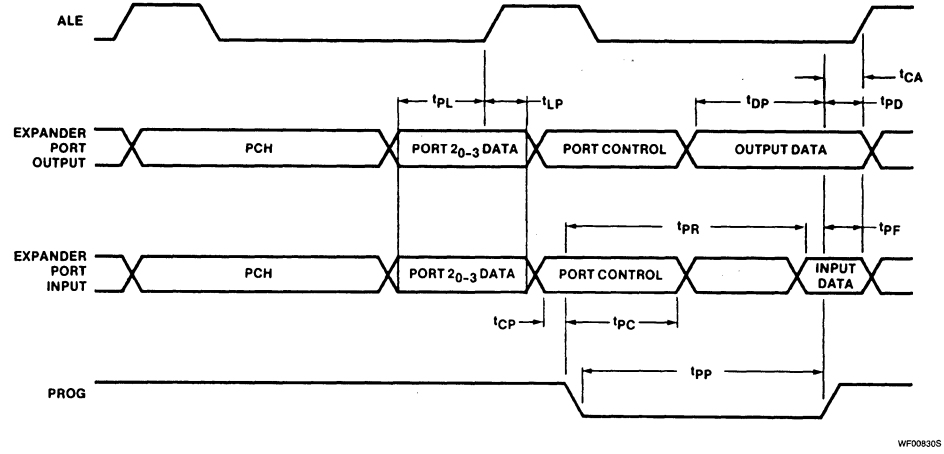


Figure 10. Port 2 Timing

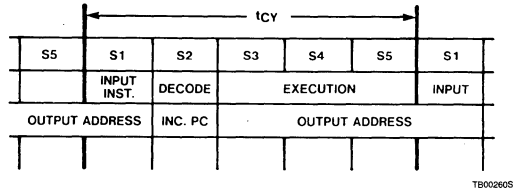


Figure 11. Instruction Cycle

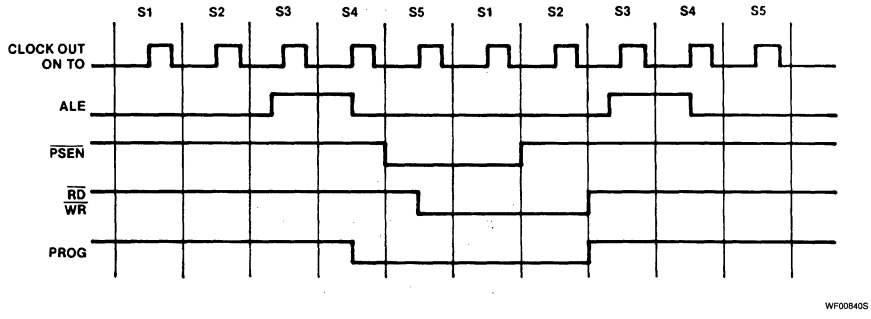


Figure 12. Instruction Cycle Timing

Table 1. INSTRUCTION SET

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	CYCLES	BYTES	FLAGS C	AC	F0	F1	F2
Accumulator										
ADD A, # data	(A) ← (A) + data	Add immediate the specified data to the accumulator.	0 0 0 0 0 1 1 / d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2	•	•			
ADD A, Rr	(A) ← (A) + (Rr) for r = 0 – 7	Add contents of designated register to the accumulator.	0 1 1 0 1 r r r	1	1	•	•			
ADD A, @ Rr	(A) ← (A) + ((Rr)) for r = 0 – 1	Add indirect the contents the data memory location to the accumulator.	0 1 1 0 0 0 0 r	1	1	•	•			
ADDC A, # data	(A) ← (A) + (C) + data	Add immediate with carry the specified data to the accumulator.	0 0 0 1 0 0 1 1 / d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2	•	•			
ADDC A, Rr	(A) ← (A) + (C) + (Rr) for r = 0 – 7	Add with carry the contents of the designated register to the accumulator.	0 1 1 1 1 r r r	1	1	•	•			
ADDC A, @ Rr	(A) ← (A) + (C) + ((Rr)) for r = 0 – 1	Add indirect with carry the contents of data memory location to the accumulator.	0 1 1 1 0 0 0 r	1	1	•	•			
ANL A, # data	(A) ← (A) AND data	Logical AND specified immediate data with accumulator.	0 1 0 1 0 0 1 1 / d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ANL A, Rr	(A) ← (A) AND (Rr) for r = 0 – 7	Logical AND contents of designated register with accumulator.	0 1 0 1 1 r r r	1	1					
ANL A, @ Rr	(A) ← (A) AND ((Rr)) for r = 0 – 1	Logical AND indirect the contents of data memory with accumulator.	0 1 0 1 0 0 0 r	1	1					
CPL A	(A) ← NOT (A)	Complement the contents of the accumulator.	0 0 1 1 0 1 1 1	1	1					
CLR A	(A) ← 0	Clear the contents of the accumulator.	0 0 1 0 0 1 1 1	1	1					
DA A		Decimal adjust the contents of the accumulator.	0 1 0 1 0 1 1 1	1	1	•	•			
DEC A	(A) ← (A) – 1	Decrement the accumulator's contents by 1.	0 0 0 0 0 1 1 1	1	1					
INC A	(A) ← (A) + 1	Increment the accumulator's contents by 1.	0 0 0 1 0 1 1 1	1	1					
ORL A, # data	(A) ← (A) OR data	Logical OR specified immediate data with accumulator.	0 1 0 0 0 0 1 1 / d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ORL A, Rr	(A) ← (A) OR (Rr) for r = 0 – 7	Logical OR contents of designated register with accumulator.	0 1 0 0 1 r r r	1	1					
ORL A, @ Rr	(A) ← (A) OR ((Rr)) for r = 0 – 1	Logical OR indirect the contents of data memory location with accumulator.	0 1 0 0 0 0 0 r	1	1					
RL A	(An + 1) ← (An) (A0) ← (A7) for N = 0 ← 6	Rotate accumulator left by 1-bit without carry.	1 1 1 0 0 1 1 1	1	1					
RLC A	(An + 1) ← (An); n = 0 – 6 (A0) ← (C) (C) ← (A7)	Rotate accumulator left by 1-bit through carry.	1 1 1 1 0 1 1 1	1	1	•				
RR A	(An) ← (An + 1); n = 0 – 6 (A7) ← (A0)	Rotate accumulator right by 1-bit without carry.	0 1 1 1 0 1 1 1	1	1					
RRC A	(An) ← (An + 1); n = 0 – 6 (A7) ← (C) (C) ← (A0)	Rotate accumulator right by 1-bit through carry.	0 1 1 0 0 1 1 1	1	1	•				
SWAP A	(A4–7) ← (A0 – 3)	Swap the 2 4-bit nibbles in the accumulator.	0 1 0 0 0 1 1 1	1	1					
XRL A, # data	(A) ← (A) XOR data	Logical XOR specified immediate data with accumulator.	1 1 0 1 0 0 1 1 / d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
XRL A, Rr	(A) ← (A) XOR (Rr) for r = 0 – 7	Logical XOR contents of designated register with accumulator.	1 1 0 1 1 r r r	1	1					
XRL A, @ Rr	(A) ← (A) XOR ((Rr)) for r = 0 – 1	Logical XOR indirect the contents of data memory location with accumulator.	1 1 0 1 0 0 0 r	1	1					

Table 1. INSTRUCTION SET (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	CYCLES	BYTES	FLAGS C	AC	F0	F1	F2
Branch										
DJNZ Rr, addr	(Rr) ← (Rr) – 1; r = 0 – 7 if (Rr) ≠ 0: (PC 0 – 7) ← addr	Decrement the specified register and test contents.	1 1 1 0 1 r r r a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JBb addr	(PC 0 – 7) ← addr if Bb = 1 (PC) ← (PC) + 2 if Bb = 0	Jump to specified address if accumulator bit is set.	b_2 b_1 b_0 1 0 0 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JC addr	(PC 0 – 7) ← addr if C = 1 (PC) ← (PC) + 2 if C = 0	Jump to specified address if carry flag is set.	1 1 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JF0 addr	(PC 0 – 7) ← addr if F0 = 1 (PC) ← (PC) + 2 if F0 = 0	Jump to specified address if flag F0 is set.	1 0 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JF1 addr	(PC 0 – 7) ← addr if F1 = 1 (PC) ← (PC) + 2 if F1 = 0	Jump to specified address if flag F1 is set.	0 1 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JMP addr	(PC 8 – 10) ← addr 8 – 10 (PC 0 – 7) ← addr 0 – 7 (PC 11) ← (DBF)	Direct jump to specified address within the 2K address block.	a_{10} a_9 a_8 0 0 1 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JMPP @ A	(PC 0 – 7) ← ((A))	Jump indirect to specified address within address page.	1 0 1 1 0 0 1 1	2	1					
JNC addr	(PC 0 – 7) ← addr if C = 0 (PC) ← (PC) + 2 if C = 1	Jump to specified address if carry flag is low.	1 1 1 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNI	(PC 0 – 7) ← addr if $\overline{INT}$ = 0 (PC) ← (PC) + 2 if $\overline{INT}$ = 1	Jump to specified address if $\overline{INT}$ input is low.	1 0 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNT0 addr	(PC 0 – 7) ← addr if T0 = 0 (PC) ← (PC) + 2 if T0 = 1	Jump to specified address if test 0 is low.	0 0 1 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNT1 addr	(PC 0 – 7) ← addr if T1 = 0 (PC) ← (PC) + 2 if T1 = 1	Jump to specified address if test 1 is low.	0 1 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNZ addr	(PC 0 – 7) ← addr if A = 0 (PC) ← (PC) + 2 if A = 0	Jump to specified address if accumulator is non-zero.	1 0 0 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JTF addr	(PC 0 – 7) ← addr if TF = 1 (PC) ← (PC) + 2 if TF = 0	Jump to specified address if timer flag is set to 1.	0 0 0 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JT0 addr	(PC 0 – 7) ← addr if T0 = 1 (PC) ← (PC) + 2 if T0 = 0	Jump to specified address if test 0 is a 1.	0 0 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JT1 addr	(PC 0 – 7) ← addr if T1 = 1 (PC) ← (PC) + 2 if T1 = 0	Jump to specified address if test 1 is a 1.	0 1 0 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JZ addr	(PC 0 – 7) ← addr if A = 0 (PC) ← (PC) + 2 if A ≠ 0	Jump to specified address if accumulator is 0.	1 1 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
Control										
EN I		Enable the external ($\overline{INT}$) interrupt.	0 0 0 0 0 1 0 1	1	1					
DIS I		Disable the external ($\overline{INT}$) interrupt.	0 0 0 1 0 1 0 1	1	1					
SEL RB0	(BS) ← 0	Select bank 0 (locations 0 – 7) of data memory.	1 1 0 0 0 1 0 1	1	1					•

Table 1. INSTRUCTION SET (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	CYCLES	BYTES	FLAGS C	AC	F0	F1	F2
Control (Cont.)										
SEL RB1	(BS) ← 1	Select bank 1 (locations 24 – 31) of data memory.	1 1 0 1 0 1 0 1	1	1					•
SEL MB0	(DBF) ← 0	Select program memory bank 0, addresses 0 – 2047.	1 1 1 0 0 1 0 1	1	1					
SEL MB1	(DBF) ← 1	Select program memory bank 1, addresses 2048 – 4095	1 1 1 1 0 1 0 1	1	1					
ENT0 CLK		Enable clock output on T0 pin.	0 1 1 1 0 1 0 1	1	1					
Data moves										
MOV A, # data	(A) ← data	Move immediate the specified data into the accumulator.	0 0 1 0 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
MOV A, Rr	(A) ← (Rr); r = 0 – 7	Move the contents of the designated register into the accumulator.	1 1 1 1 1 r r r	1	1					
MOV A, @ Rr	(A) ← ((Rr)); r = 0 – 1	Move indirect the contents of data memory location into the accumulator.	1 1 1 1 0 0 0 r	1	1					
MOV A, PSW	(A) ← (PSW)	Move contents of the program status word into the accumulator.	1 1 0 0 0 1 1 1	1	1					
MOV Rr, # data	(Rr) ← data; r = 0 – 7	Move immediate the specified data into the designated register.	1 0 1 1 1 r r r d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
MOV Rr, A	(Rr) ← (A); r = 0 – 7	Move accumulator contents into the designated register.	1 0 1 0 1 r r r	1	1					
MOV @ Rr, A	((Rr)) ← (A); r = 0 – 1	Move indirect accumulator contents into data memory location.	1 0 1 0 0 0 0 r	1	1					
MOV @ Rr, # data	((Rr)) ← data; r = 0 – 1	Move indirect the specified data into data memory.	1 0 1 1 0 0 0 r d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
MOV PSW, A	(PSW) ← (A)	Move contents of accumulator into the program status word.	1 1 0 1 0 1 1 1	1	1	•	•	•		•
MOVP A, @ A	(A) ← ((A))	Move data in the current page into the accumulator.	1 0 1 0 0 0 1 1	2	1					
MOVP3 A, @ A	(A) ← ((A)) in page 3	Move data in page 3 into the accumulator.	1 1 1 0 0 0 1 1	2	1					
MOVX A, @ Rr	(A) ← ((Rr)); r = 0 – 1	Move indirect the contents of external memory location into the accumulator.	1 0 0 0 0 0 0 r	2	1					
MOVX @ Rr, A	((Rr)) ← (A); r = 0 – 1	Move indirect the contents of the accumulator into external memory.	1 0 0 1 0 0 0 r	2	1					
XCH A, Rr	(A) ↔ (Rr); r = 0 – 7	Exchange the accumulator and designated register's contents.	0 0 1 0 1 r r r	1	1					
XCH A, @ Rr	(A) ↔ ((Rr)); r = 0 – 1	Exchange indirect contents of accumulator and location in data memory.	0 0 1 0 0 0 0 r	1	1					
XCHD A, @ Rr	(A 0 – 3) ↔ (Rr)(0 – 3) r = 0 – 1	Exchange indirect 4-bit contents of accumulator and data memory.	0 0 1 1 0 0 0 r	1	1					
Flags										
CPL C	(C) ← NOT (C)	Complement content of carry bit.	1 0 1 0 0 1 1 1	1	1	•				
CPL F0	(F0) ← NOT (F0)	Complement content of flag F0.	1 0 0 1 0 1 0 1	1	1			•		
CPL F1	(F1) ← NOT (F1)	Complement content of flag F1.	1 0 1 1 0 1 0 1	1	1				•	
CLR C	(C) ← 0	Clear content of carry bit to 0.	1 0 0 1 0 1 1 1	1	1	•				
CLR F0	(F0) ← 0	Clear content of flag 0 to 0.	1 0 0 0 0 1 0 1	1	1			•		
CLR F1	(F1) ← 0	Clear content of flag 1 to 0.	1 0 1 0 0 1 0 1	1	1					•
Input/output										
ANL BUS, # data	(BUS) ← (BUS) AND data	Logical AND immediate specified data with BUS.	1 0 0 1 1 0 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ANL Pp, # data	(Pp) ← (Pp) AND data p = 1 – 2	Logical AND immediate specified data with designated port (1 or 2).	1 0 0 1 1 0 p p d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ANLD Pp, A	(Pp) ← (Pp) AND (A 0 – 3) p = 4 – 7	Logical AND contents of accumulator with designated port (4 – 7).	1 0 0 1 1 1 p p	2	1					

Table 1. INSTRUCTION SET (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	CYCLES	BYTES	FLAGS C	AC	F0	F1	F2
Input/output (Cont.)										
IN A, Pp	(A) ← (Pp); p = 1 – 2	Input data from designated port (1 – 2) into accumulator.	0 0 0 0 1 0 p p	2	1					
INS A, BUS	(A) ← (BUS)	Input strobed BUS data into accumulator.	0 0 0 0 1 0 0 0	1	2					
MOVD A, Pp	(A 0 – 3) ← (Pp); p = 4 – 7 (A 4 – 7) ← 0	Move contents of designated port (4 – 7) into accumulator.	0 0 0 0 1 1 p p	2	1					
MOVD Pp, A	(Pp) ← A 0 – 3; p = 4 – 7	Move contents of accumulator to designated port (4 – 7).	0 0 1 1 1 1 p p	1	1					
ORLD Pp, A	(Pp) ← (Pp) OR (A 0 – 3) p = 4 – 7	Logical OR contents of accumulator with designated port (4 – 7).	1 0 0 0 1 1 p p	1	1					
ORL BUS, # data	(BUS) ← (BUS) OR data	Logical OR immediate specified data with BUS.	1 0 0 0 1 0 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ORL Pp, # data	(Pp) ← (Pp) OR data p = 1 – 2	Logical OR immediate specified data with designated port (1 – 2).	1 0 0 0 1 0 p p d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
OUTL BUS, A	(BUS) ← (A)	Output contents of accumulator onto BUS.	0 0 0 0 0 0 1 0	1	2					
OUTL Pp, A	(Pp) ← (A); p = 1 – 2	Output contents of accumulator to designated port (1 – 2).	0 0 1 1 1 0 p p	1	1					
Registers										
DEC Rr	(Rr) ← (Rr) – 1; r = 0 – 7	Decrement contents of designated register by 1.	1 1 0 0 1 r r r	1	1					
INC Rr	(Rr) ← (Rr) + 1; r = 0 – 7	Increment contents of designated register by 1.	0 0 0 1 1 r r r	1	1					
INC @ Rr	((Rr)) ← ((Rr)) + 1; r = 0 – 1	Increment indirect the contents of data memory location by 1.	0 0 0 1 0 0 0 r	1	1					
Subroutine										
CALL addr	((SP)) ← (PC), (PSW 4 – 7) (SP) ← (SP) + 1 (PC 8 – 10) ← addr 8 – 10 (PC 0 – 7) ← addr 0 – 7 (PC 11) ← DBF	Call designated subroutine.	a_{10} a_9 a_8 1 0 1 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
RET	(SP) ← (SP) – 1 (PC) ← ((SP))	Return from subroutine without restoring program status word.	1 0 0 0 0 0 1 1	2	1					
RETR	(SP) ← (SP) – 1 (PC) ← ((SP)) (PSW 4 – 7) ← ((SP))	Return from subroutine restoring program status word.	1 0 0 1 0 0 1 1	2	1					
Timer/counter										
EN TCNTI		Enable timer/counter interrupt.	0 0 1 0 0 1 0 1	1	1					
DIS TCNTI		Disable timer/counter interrupt.	0 0 1 1 0 1 0 1	1	1					
MOV A, T	(A) ← (T)	Move contents of timer/counter into accumulator.	0 1 0 0 0 0 1 0	1	1					
MOV T, A	(T) ← (A)	Move contents of accumulator into timer/counter.	0 1 1 0 0 0 1 0	1	1					
STOP TCNT		Stop count for event counter or timer.	0 1 1 0 0 1 0 1	1	1					
STRT CNT		Start count for event counter.	0 1 0 0 0 1 0 1	1	1					
STRT T		Start count for timer.	0 1 0 1 0 1 0 1	1	1					
Miscellaneous										
NOP		No operation performed	0 0 0 0 0 0 0 0	1	1					

NOTES:
1. Instruction code designations r and p form the binary representation of the registers and ports involved.
2. The dot under the appropriate flag bit indicates that its content is subject to change by the instruction in which it appears.
3. Numerical subscripts appearing in the FUNCTION column reference the specific bits affected.

SYMBOL DEFINITIONS

SYMBOL	DESCRIPTION
A	The accumulator
AC	The auxiliary carry flag
addr	Program memory address (11 bits)
Bb	Bit designator (b = 0 – 7)
BS	The bank switch
C	Carry flag
CLK	Clock signal
CNT	Event counter
D	Nibble designator (4 bits)
DBF	Program memory bank flip-flop
data	Number or expression (8 bits)
F_0, F_1	Flags 0, 1
I	Interrupt
$\overline{INT}$	External interrupt
P	"In-Page" operation designator
P_p	Port designator (p = 1, 2 or 4 – 7)
PSW	Program status word
Rr	Register designator (r = 0, 1 or 0 – 7)
SP	Stack pointer
T	Timer
TF	Timer flag
T0, T1	Testable inputs 0, 1
#	Prefix for immediate data
@	Prefix for indirect address
$	Program counter's current value
←	Replaced by
↔	Exchanged with

3

Table 2. INSTRUCTION TIMING**

	CYCLE 1					CYCLE 2				
INSTRUCTION	**S1**	**S2**	**S3**	**S4**	**S5**	**S1**	**S2**	**S3**	**S4**	**S5**
IN A,P	Fetch Instruction	Increment Program Counter	—	Increment Timer	—	—	Read Port	* —	—	—
OUTL P,A	Fetch Instruction	Increment Program Counter	—	Increment Timer	Output To Port	—	—	* —	—	—
ANL P, # data	Fetch Instruction	* Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	* Increment Program Counter	Output To Port	—
ORL P, # data	Fetch Instruction	* Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	* Increment Program Counter	Output To Port	—
INS A, BUS	Fetch Instruction	Increment Program Counter	—	Increment Timer	—	—	Read Port	* —	—	—
OUTL BUS, A	Fetch Instruction	Increment Program Counter	—	Increment Timer	Output To Port	—	—	* —	—	—
ANL BUS, # data	Fetch Instruction	* Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	* Increment Program Counter	Output To Port	—
ORL BUS, # data	Fetch Instruction	* Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	* Increment Program Counter	Output To Port	—
MOVX @R,A	Fetch Instruction	Increment Program Counter	Output RAM Address	Increment Timer	Output Data to RAM	—	—	* —	—	—
MOVX A,@R	Fetch Instruction	Increment Program Counter	Output RAM Address	Increment Timer	—	—	Read Data	* —	—	—
MOVD A, P_i	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	—	—	Read P2 Lower	* —	—	—
MOVD P_i, A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data TO P2 Lower	—	—	* —	—	—
ANLD P, A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data	—	—	* —	—	—
ORLD P, A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data	—	—	* —	—	—
J (CONDITIONAL)	Fetch Instruction	* Increment Program Counter	Sample Condition	Increment Timer	—	Fetch Immediate Data	—	* Update Program Counter	—	—
START CNT/STRT T	Fetch Instruction	* Increment Program Counter	—	—	Start Counter					
STOP TCNT	Fetch Instruction	* Increment Program Counter	—	—	Stop Counter					
EN I	Fetch Instruction	* Increment Program Counter	—	Enable Interrupt	—					
DIS I	Fetch Instruction	* Increment Program Counter	—	Disable Interrupt	—					
ENT0 CLK	Fetch Instruction	* Increment Program Counter	—	Enable Clock	—					

NOTES:
*Valid instruction address are output at this time if external program memory is being accessed.
**See figures 11 and 12 for instruction cycle and cycle timing.

Signetics

SCC80 Series
CMOS Single-Chip 8-Bit Microcontroller

Preliminary Specification

Microprocessor Products

DESCRIPTION

The Signetics SCC80C Series is a family of CMOS, high-performance, single-chip microcontrollers which are functionally-compatible, low-power CMOS versions of the popular NMOS SCN80 Series. Signetics CMOS technology combines the high-speed and density characteristics of HMOS with the low-power attributes of CMOS.

All SCC80C Series devices are pin and program compatible, differing only in the size of the on-board program ROM and data RAM, as follows:

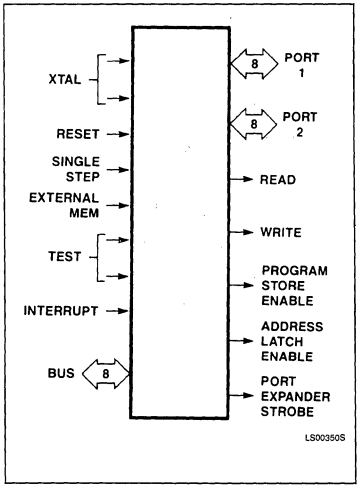
TYPE	RAM SIZE	ROM SIZE
SCC80C48	64 x 8	1K x 8
SCC80C49	128 x 8	2K x 8
SCC80C50	256 x 8	4K x 8
SCC80C35	64 x 8	–
SCC80C39	128 x 8	–
SCC80C40	256 x 8	–

The SCC80C Series devices also provide three 8-bit I/O ports, two test inputs and an interrupt input, for a total of 27 I/O lines. An on-chip, 8-bit timer/counter is provided which can count, under program control, either internal clock pulses (with a divide-by-32 prescaler) or external events. The counter can be programmed to cause an interrupt on terminal count. Systems that require extra capacity can be expanded using CMOS external memories and peripherals.

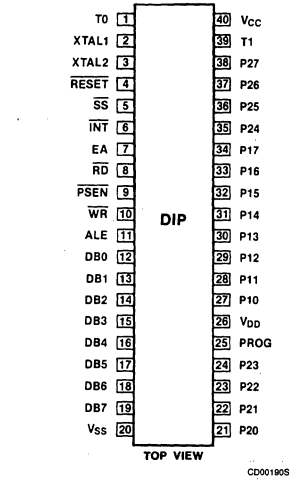
FEATURES

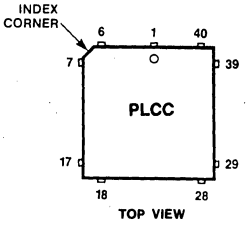
- **CMOS microcontroller pin compatible with Signetics SCN80 Series**
  - **80C48/80C49/80C50 with ROM**
  - **80C35/80C39/80C40 without ROM**
- **SCC80C48 Series**
  - **1 to 4K ROM**
  - **64 to 256 bytes RAM**
  - **3 x 8-bit I/O ports**
  - **Test and interrupt inputs**
  - **Timer/counter**
- **ROM and RAM expandable with external components**
- **1.36μs instruction cycle @ 11MHz**
  - **All instructions 1 or 2 cycles**
- **Three low-power consumption selections**
  - **Normal operation: 12mA @ 5V, 11MHz**
  - **Idle mode: 5mA @ 5V, 11MHz**
  - **Power down: 2μA @ 2.0V**
- **Low-power applications permit:**
  - **Battery operation**
  - **Reduced cooling requirements**
  - **Maintain operation through power line interruptions**
- **TTL compatible V_{CC} = 5V ±10%**
  CMOS compatible V_{CC} = 5V ±20%

LOGIC SYMBOL



PIN CONFIGURATION

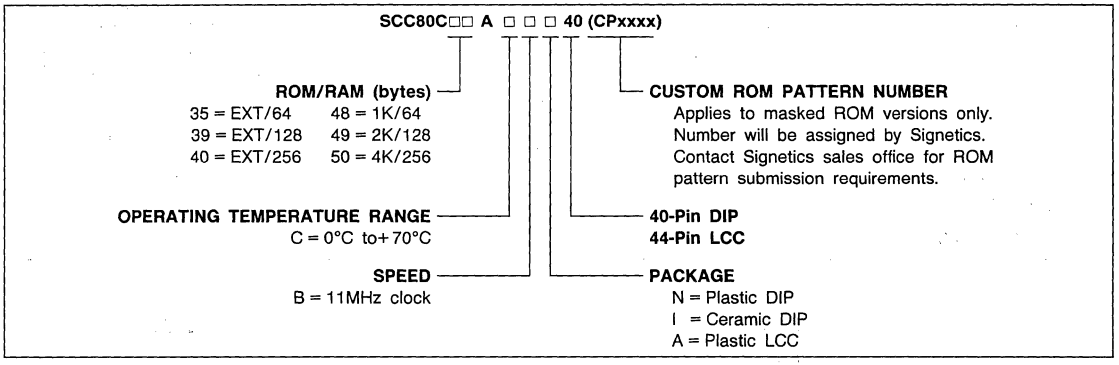




Pin	Function	Pin	Function
1	NC	23	NC
2	T0	24	P20
3	XTAL1	25	P21
4	XTAL2	26	P22
5	$\overline{RESET}$	27	P23
6	$\overline{SS}$	28	PROG
7	$\overline{INT}$	29	V_{DD}
8	EA	30	P10
9	$\overline{RD}$	31	P11
10	$\overline{PSEN}$	32	P12
11	$\overline{WR}$	33	P13
12	NC	34	NC
13	ALE	35	P14
14	DB0	36	P15
15	DB1	37	P16
16	DB2	38	P17
17	DB3	39	P24
18	DB4	40	P25
19	DB5	41	P26
20	DB6	42	P27
21	DB7	43	T1
22	V_{SS}	44	V_{CC}

CMOS Single-Chip 8-Bit Microcontroller — SCC80 Series

ORDERING CODE



3

PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	Circuit ground potential.
V_{DD}	26	29	I	Low power standby.
V_{CC}	40	44	I	**Main Power Supply:** +5V during operation.
PROG	25	28	O	Output strobe for 82C43 I/O expander.
P10 – P17	27 – 34	30 – 33, 35 – 38	I/O	**Port 1:** 8-bit quasi-bidirectional port.
P20 – P27	21 – 24, 35 – 38	24 – 27, 39 – 42	I/O	**Port 2:** 8-bit quasi-bidirectional port. P20 – 23 contain the four high-order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 82C43.
DB0 – DB7	12 – 19	14 – 21	I/O	**Data Bus:** True bidirectional port which can be written or read synchronously using the $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched. Contains the eight low-order program counter bits during an external program memory fetch and receives the addressed instruction under the control of $\overline{PSEN}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{RD}$ and $\overline{WR}$.
T0	1	2	I	Input pin testable using the conditional transfer instructions JT0 and JNT0. T0 can be designated as a clock output using the ENT0 CLK instruction.
T1	39	43	I	Input pin testable using the JT1 and JNT1 instructions. Can be designated the timer/counter input using the STRT CNT instruction.
XTAL1	2	3	I	**Crystal 1:** One side of the crystal input for internal oscillator. Also input for external source (non-TTL V_{IH}).
XTAL2	3	4	I	**Crystal 2:** Other side of crystal input.
$\overline{INT}$	6	7	I	**Interrupt:** Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset. Also testable with conditional jump instruction. Interrupt must remain low for at least three machine cycles for proper operation.
$\overline{RD}$	8	9	O	**Read:** Output strobe activated during a bus read. Can be used to enable data onto the bus from an external device. Used as a read strobe to external data memory.
$\overline{RESET}$	4	5	I	**Reset:** Input which is used to initialize the processor (non-TTL V_{IH}).
$\overline{WR}$	10	11	O	**Write:** Output strobe during a bus write. Used as write strobe to external data memory.
ALE	11	13	O	**Address Latch Enable:** Occurs once during each cycle and is useful as a clock output. The negative edge of ALE strobes address into external data and program memory.
$\overline{PSEN}$	9	10	O	**Program Store Enable:** Output occurs only during a fetch to external program memory.
$\overline{SS}$	5	6	I	**Single Step:** Can be used in conjunction with ALE to "single step" the processor through each instruction.
EA	7	8	I	**External Access:** Forces all program memory fetches to reference external memory. Useful for emulation and debug, and essential for testing and program verification.

CMOS Single-Chip 8-Bit Microcontroller — SCC80 Series

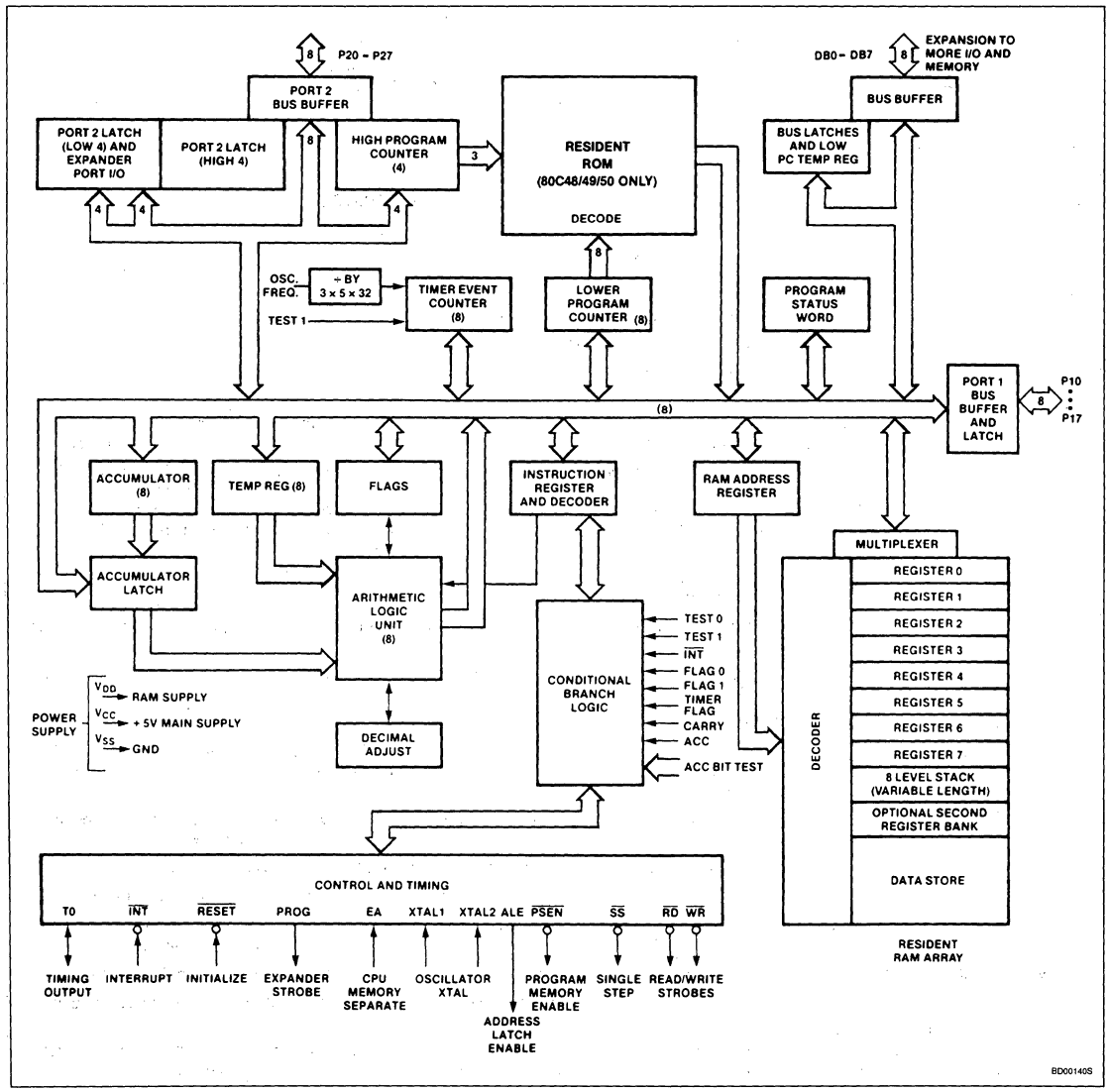
The CMOS design of the SCC80C Series opens new application areas that require battery operation, low power standby, wider supply voltage ranges, and the ability to maintain operation during AC power line interruptions. The use of cooling fans can be reduced or eliminated, making possible the design of less noisy or sealed equipment. These applications include mobile, portable and hand-held equipment for the automotive, telecommunications, and consumer markets.

IDLE MODE

The 80C48/80C49/80C50, when placed in the idle mode, keeps the oscillator, the internal timer and the external interrupt and counter pins functioning, and maintains the internal register and RAM status.

To place the 80C48/80C49/80C50 in idle mode, a command instruction (op code 01H) is executed. To terminate idle mode, a reset must be performed or interrupts must be enabled and an interrupt signal generated. There are two interrupt sources that can restore normal operation. One is an external signal applied to the interrupt pin. The other is from the overflow of the timer/counter. When either interrupt is invoked, the CPU is taken out of idle mode and vectors to the interrupt's service routine address. Along with the idle mode, the standard 8048 Series power-down mode is still maintained.

BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
Voltage on any pin with respect to ground[3]	−0.5 to V_{CC} +1	V
Maximum voltage on any pin with respect to ground[3]	7	V
Power dissipation	1.5	W

DC ELECTRICAL CHARACTERISTICS T_A = 0°C to +70°C, $V_{CC} = V_{DD} = 5V \pm 20\%$: $|V_{CC}-V_{DD}| \leq 1.5V$; $V_{SS} = 0V$[6]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IL}	Input low voltage (except X1,$\overline{RESET}$)		−0.5	0.8	V
V_{IL1}	Input low voltage (X1, $\overline{RESET}$)		−0.5	0.6	V
V_{IH}	Input high voltage (except XTAL1, $\overline{RESET}$)		$0.4V_{CC}$	V_{CC}	V
V_{IH1}	Input high voltage (X1, $\overline{RESET}$)		$0.7V_{CC}$	V_{CC}	V
V_{OL}	Output low voltage (bus)	I_{OL} = 2.0mA		0.45	V
V_{OL1}	Output low voltage ($\overline{RD}$,$\overline{WR}$,$\overline{PSEN}$,ALE)	I_{OL} = 1.8mA		0.45	V
V_{OL2}	Output low voltage (PROG)	I_{OL} = 1.0mA		0.45	V
V_{OL3}	Output low voltage (all other outputs)	I_{OL} = 1.6 mA		0.45	V
V_{OH}	Output high voltage (bus)	I_{OH} = −400μA	$0.75V_{CC}$		V
V_{OH1}	Output high voltage ($\overline{RD}$,$\overline{WR}$,$\overline{PSEN}$, ALE)	I_{OH} = −100μA	$0.75V_{CC}$		V
V_{OH2}	Output high voltage (all other outputs)	I_{OH} = −40μA	$0.75V_{CC}$		V
I_{LI}	Input leakage current (T1,$\overline{INT}$,EA)	$V_{SS} \leq V_{IN} \leq V_{CC}$		±5	μA
I_{LI1}	Input leakage current (P10-P17,P20-P27,$\overline{SS}$)	$V_{SS} \leq V_{IN} \leq V_{CC}$		−500	μA
I_{LO}	Output leakage current (bus,T0,high-impedance state)	$V_{SS} \leq V_{IN} \leq V_{CC}$		±5	μA
I_{LR}	Input leakage current ($\overline{RESET}$)	$V_{SS} \leq V_{IN} \leq V_{CC}$	−20	−300	μA
I_{PD}	Power-down standby current	V_{DD} = 2V, $\overline{RESET} \leq V_{IL}$		2	μA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other conditions above those in the AC and DC Electrical Characteristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages greater than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground. For testing, all input signals swing between 0.45V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V, and at output voltages of 0.8V and 2.0V, as appropriate.
6. Typical values are at +25°C, typical supply voltages and typical processing parameters.
7. Control outputs C_L = 80pF, bus outputs C_L = 150pF.
8. Bus high-impedance load = 20pF.
9. f(t) assumes 50% duty cycle on X1,X2. Maximum clock period is for a 1MHz crystal input.

ABSOLUTE MAXIMUM UNLOADED CURRENT

I_{CC} ACTIVE CURRENT (mA)			
V_{CC}	4V	5V	6V
1MHz	1.5	2.3	3
6MHz	5	6.8	8.5
11MHz	9	12	15

I_{CC} IDLE CURRENT (mA)			
V_{CC}	4V	5V	6V
1MHz	0.7	1	1.2
6MHz	2	3	4
11MHz	3.5	4.8	6

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = V_{DD} = 5V \pm 20\%$: $|V_{CC}-V_{DD}| \leq 1.5V$; $V_{SS} = 0V$[6,7]

PARAMETER		f(t)	11MHz		UNIT
			Min	Max	
t[9]	Clock period	1/xtal freq	90.9	1000	ns
t_{LL}	ALE pulse width	3.5t – 170	150		ns
t_{AL}[8]	Addr set-up to ALE	2t – 110	70		ns
t_{LA}	Addr hold from ALE	t – 40	50		ns
t_{CC1}	Control pulse width ($\overline{RD}$,$\overline{WR}$)	7.5t – 200	480		ns
t_{CC2}	Control pulse width ($\overline{PSEN}$)	6t – 200	350		ns
t_{DW}	Data set-up before $\overline{WR}$	6.5t – 200	390		ns
t_{WD}	Data hold after $\overline{WR}$	t – 50	40		ns
t_{DR}	Data hold ($\overline{RD}$,$\overline{PSEN}$)	1.5t – 30	0	110	ns
t_{RD1}	$\overline{RD}$ to data in	6t – 170		350	ns
t_{RD2}	$\overline{PSEN}$ to data in	4.5t – 170		190	ns
t_{AW}	Addr set-up to $\overline{WR}$	5t – 150	300		ns
t_{AD1}	Addr set-up to data ($\overline{RD}$)	10.5t – 220		730	ns
t_{AD2}	Addr set-up to data ($\overline{PSEN}$)	7.5t – 220		460	ns
t_{AFC1}[8]	Addr float to $\overline{RD}$,$\overline{WR}$	2t – 40	140		ns
t_{AFC2}[8]	Addr float to $\overline{PSEN}$	0.5t – 40	10		ns
t_{LAFC1}	ALE to control ($\overline{RD}$,$\overline{WR}$)	3t – 75	200		ns
t_{LAFC2}	ALE to control ($\overline{PSEN}$)	1.5t – 75	60		ns
t_{CA1}	Control to ALE ($\overline{RD}$,$\overline{WR}$,PROG)	t – 40	50		ns
t_{CA2}	Control to ALE ($\overline{PSEN}$)	4t – 40	320		ns
t_{CP}	Port control set-up to PROG	1.5t – 80	50		ns
t_{PC}	Port control hold to PROG	4t – 260	100		ns
t_{PR}	PROG to P2 input valid	8.5t – 120		650	ns
t_{PF}	Input data hold from PROG	1.5t	0	140	ns
t_{DP}	Output data set-up	6t – 290	250		ns
t_{PD}	Output data hold	1.5t – 90	40		ns
t_{PP}	PROG pulse width	10.5t – 250	700		ns
t_{PL}	Port 2 I/O set-up to ALE	4t – 200	160		ns
t_{LP}	Port 2 I/O hold to ALE	1.5t – 120	15		ns
t_{PV}	Port output from ALE	4.5t + 100		510	ns
t_{0PRR}	T0 rep rate	3t	270		ns
t_{CY}	Cycle time	15t	1.36	15.0	μs

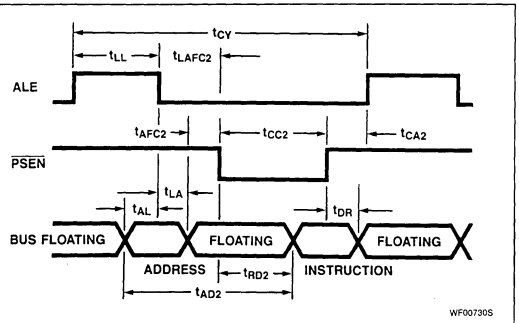


Figure 1. Instruction Fetch from Program Memory

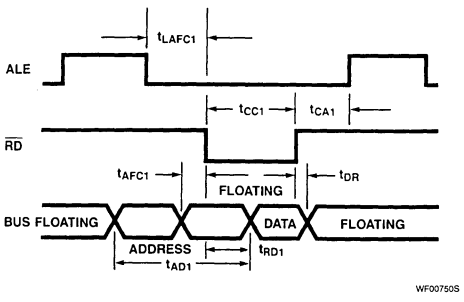


Figure 2. Read from External Data Memory

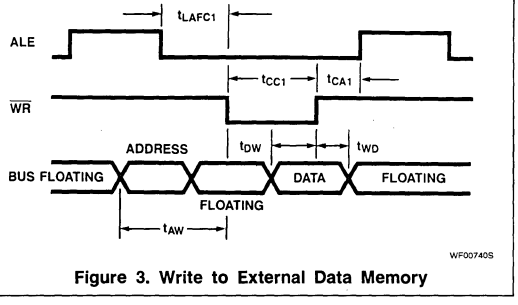


Figure 3. Write to External Data Memory

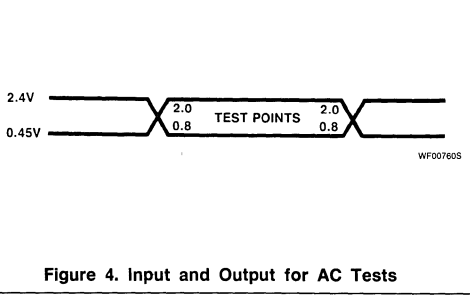


Figure 4. Input and Output for AC Tests

3

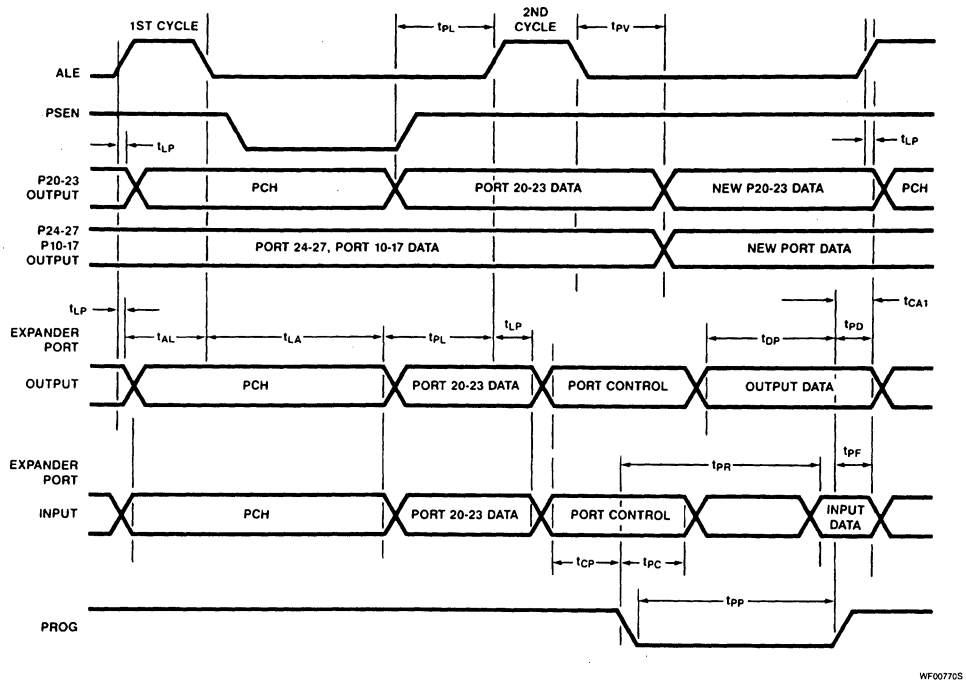
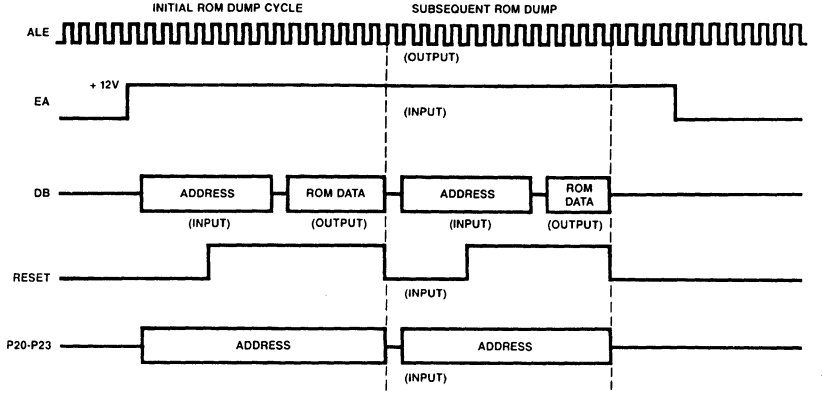


Figure 5. Port 1/Port 2 Timing



	C48	C49	C50
A10	0	ADDR	ADDR
A11	0	0	ADDR

$V_{CC} = V_{DD} = +5V$
$V_{SS} = 0V$

Figure 6. Suggested ROM Verification Algorithm for CHMOS Devices Only

Table 1. INSTRUCTION SET

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	CYCLES	BYTES	FLAGS C	AC	F0	F1	BS
Accumulator										
ADD A, # data	(A) ← (A) + data	Add immediate the specified data to the accumulator.	0 0 0 0 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2	•	•			
Add A, Rr	(A) ← (A) + (Rr) for r = 0 – 7	Add contents of designated register to the accumulator.	0 1 1 0 1 r r r	1	1	•	•			
Add A, @ Rr	(A) ← (A) + ((Rr)) for r = 0 – 1	Add indirect the contents the data memory location to the accumulator.	0 1 1 0 0 0 0 r	1	1	•	•			
ADDC A, # data	(A) ← (A) + (C) + data	Add immediate with carry the specified data to the accumulator.	0 0 0 1 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2	•	•			
ADDC A, Rr	(A) ← (A) + (C) + (Rr) for r = 0 – 7	Add with carry the contents of the designated register to the accumulator.	0 1 1 1 1 r r r	1	1	•	•			
ADDC A, @ Rr	(A) ← (A) + (C) + ((Rr)) for r = 0 – 1	Add indirect with carry the contents of data memory location to the accumulator.	0 1 1 1 0 0 0 r	1	1	•	•			
ANL A, # data	(A) ← (A) AND data	Logical AND specified immediate data with accumulator.	0 1 0 1 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ANL A, Rr	(A) ← (A) AND (Rr) for r = 0 – 7	Logical AND contents of designated register with accumulator.	0 1 0 1 1 r r r	1	1					
ANL A, @ Rr	(A) ← (A) AND((Rr)) for r = 0 – 1	Logical AND indirect the contents of data memory with accumulator.	0 1 0 1 0 0 0 r	1	1					
CPL A	(A) ← NOT (A)	Complement the contents of the accumulator.	0 0 1 1 0 1 1 1	1	1					
CLR A	(A) ← 0	Clear the contents of the accumulator.	0 0 1 0 0 1 1 1	1	1					
DA A		Decimal adjust the contents of the accumulator.	0 1 0 1 0 1 1 1	1	1	•	•			
DEC A	(A) ← (A) – 1	Decrement the accumulator's contents by 1.	0 0 0 0 0 1 1 1	1	1					
INC A	(A) ← (A) + 1	Increment the accumulator's contents by 1.	0 0 0 1 0 1 1 1	1	1					
ORL A, # data	(A) ← (A) OR data	Logical OR specified immediate data with accumulator.	0 1 0 0 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ORL A, Rr	(A) ← (A) OR (Rr) for r = 0 – 7	Logical OR contents of designated register with accumulator.	0 1 0 0 1 r r r	1	1					
ORL A, @ Rr	(A) ← (A) OR ((Rr)) for r = 0 – 1	Logical OR indirect the contents of data memory location with accumulator.	0 1 0 0 0 0 0 r	1	1					
RL A	(An + 1) ← (An) (A_0) ← (A_7) for N = 0 ← 6	Rotate accumulator left by 1-bit without carry.	1 1 1 0 0 1 1 1	1	1					
RLC A	(An + 1) ← (An); n = 0 – 6 (A_0) ← (C) (C) ← (A_7)	Rotate accumulator left by 1-bit through carry.	1 1 1 1 0 1 1 1	1	1	•				
RR A	(An) ← (An + 1); n = 0 – 6 (A_7) ← (A_0)	Rotate accumulator right by 1-bit without carry.	0 1 1 1 0 1 1 1	1	1					
RRC A	(An) ← (An + 1); n = 0 – 6 (A_7) ← (C) (C) ← (A_0)	Rotate accumulator right by 1-bit through carry.	0 1 1 0 0 1 1 1	1	1	•				
SWAP A	(A_{4-7}) ← (A_0 – 3)	Swap the 2 4-bit nibbles in the accumulator.	0 1 0 0 0 1 1 1	1	1					
XRL A, # data	(A) ← (A) XOR data	Logical XOR specified immediate data with accumulator.	1 1 0 1 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
XRL A, Rr	(A) ← (A) XOR (Rr) for r = 0 – 7	Logical XOR contents of designated register with accumulator.	1 1 0 1 1 r r r	1	1					
XRL A, @ Rr	(A) ← (A) XOR ((Rr)) for r = 0 – 1	Logical XOR indirect the contents of data memory location with accumulator.	1 1 0 1 0 0 0 r	1	1					

Table 1. INSTRUCTION SET (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	CYCLES	BYTES	FLAGS C	AC	F0	F1	BS
Branch										
DJNZ Rr, addr	(Rr) ← (Rr) – 1; r = 0 – 7 if (Rr) = 0: (PC 0 – 7) ← addr	Decrement the specified register and test contents	1 1 1 0 1 r r r a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JBd addr	(PC 0 – 7) ← addr if Bb = 1 (PC) ← (PC) + 2 if Bb = 0	Jump to specified address if accumulator bit is set.	b_2 b_1 b_0 1 0 0 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JC addr	(PC 0 – 7) ← addr if C = 1 (PC) ← (PC) + 2 if C = 0	Jump to specified address if carry flag is set.	1 1 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JF0 addr	(PC 0 – 7) ← addr if F0 = 1 (PC) ← (PC) + 2 if F0 = 0	Jump to specified address if flag F0 is set.	1 0 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JF1 addr	(PC 0 – 7) ← addr if F1 = 1 (PC) ← (PC) + 2 if F1 = 0	Jump to specified address if flag F1 is set.	0 1 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JMP addr	(PC8 – 10) ← addr 8 – 10 (PC0 – 7) ← addr 0 – 7 (PC11) ← (DBF)	Direct jump to specified address within the 2K address block.	a_{10} a_9 a_8 0 0 1 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JMPP @ A	(PC0 – 7) ← ((A))	Jump indirect to specified address within address page.	1 0 1 1 0 0 1 1	2	1					
JNC addr	(PC0 – 7) ← addr if C = 0 (PC) ← (PC) + 2 if C = 1	Jump to specified address if carry flag is low.	1 1 1 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNI	(PC0 – 7) ← addr if $\overline{INT}$ = 0 (PC) ← (PC) + 2 if $\overline{INT}$ = 1	Jump to specified address if $\overline{INT}$ input is low.	1 0 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNT0 addr	(PC0 – 7) ← addr if T0 = 0 (PC) ← (PC) + 2 if T0 = 1	Jump to specified address if test 0 is low.	0 0 1 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNT1 addr	(PC0 – 7) ← addr if T1 = 0 (PC) ← (PC) + 2 if T1 = 1	Jump to specified address if test 1 is low.	0 1 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JNZ addr	(PC0 – 7) ← addr if A ≠ 0 (PC) ← (PC) + 2 if A = 0	Jump to specified address if accumulator is non-zero.	1 0 0 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JTF addr	(PC0 – 7) ← addr if TF = 1 (PC) ← (PC) + 2 if TF = 0	Jump to specified address if timer flag is set to 1.	0 0 0 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JT0 addr	(PC0 – 7) ← addr if T0 = 1 (PC) ← (PC) + 2 if T0 = 0	Jump to specified address if test 0 is a 1.	0 0 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JT1 addr	(PC0 – 7) ← addr if T1 = 1 (PC) ← (PC) + 2 if T1 = 0	Jump to specified address if test 1 is a 1.	0 1 0 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
JZ addr	(PC0 – 7) ← addr if A = 0 (PC) ← (PC) + 2 if A ≠ 0	Jump to specified address if accumulator is 0.	1 1 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
Control										
EN I		Enable the external ($\overline{INT}$) interrupt.	0 0 0 0 0 1 0 1	1	1					
DIS I		Disable the external ($\overline{INT}$) interrupt.	0 0 0 1 0 1 0 1	1	1					
SEL RB0	(BS) ← 0	Select bank 0 (locations 0 – 7) of data memory.	1 1 0 0 0 1 0 1	1	1					•
SEL RB1	(BS) ← 1	Select bank 1(locations 24 – 31) of data memory.	1 1 0 1 0 1 0 1	1	1					•
SEL MB0	(DBF) ← 0	Select program memory bank 0, addresses 0 – 2047.	1 1 1 0 0 1 0 1	1	1					
SEL MB1	(DBF) ← 1	Select program memory bank 1, addresses 2048 – 4095	1 1 1 1 0 1 0 1	1	1					
ENT0 CLK		Enable clock output on T0 pin.	0 1 1 1 0 1 0 1	1	1					

Table 1. INSTRUCTION SET (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE	CYCLES	BYTES	FLAGS				
			D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0			C	AC	F0	F1	BS
Data moves										
MOV A, # data	(A) ← data	Move immediate the specified data into the accumulator.	0 0 1 0 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
MOV A, Rr	(A) ← (Rr); r = 0 – 7	Move the contents of the designated register into the accumulator.	1 1 1 1 1 r r r	1	1					
MOV A, @ Rr	(A) ← ((Rr)); r = 0 – 1	Move indirect the contents of data memory location into the accumulator.	1 1 1 1 0 0 0 r	1	1					
MOV A, PSW	(A) ← (PSW)	Move contents of the program status word into the accumulator.	1 1 0 0 0 1 1 1	1	1					
MOV Rr, # data	(Rr) ← data; r = 0 – 7	Move immediate the specified data into the designated register.	1 0 1 1 1 r r r d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
MOV Rr, A	(Rr) ← (A); r = 0 – 7	Move accumulator contents into the designated register.	1 0 1 0 1 r r r	1	1					
MOV @ Rr, A	((Rr)) ← (A); r = 0 – 1	Move indirect accumulator contents into data memory location.	1 0 1 0 0 0 0 r	1	1					
MOV @ Rr, # data	((Rr)) ← data; r = 0 – 1	Move indirect the specified data into data memory.	1 0 1 1 0 0 0 r d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
MOV PSW, A	(PSW) ← (A)	Move contents of accumulator into the program status word.	1 1 0 1 0 1 1 1	1	1	•	•	•		•
MOVP A, @ A	(A) ← ((A))	Move data in the current page into the accumulator.	1 0 1 0 0 0 1 1	2	1					
MOVP3 A, @ A	(A) ← ((A)) in page 3	Move data in page 3 into the accumulator.	1 1 1 0 0 0 1 1	2	1					
MOVX A, @ Rr	(A) ← ((Rr)); r = 0 – 1	Move indirect the contents of external memory location into the accumulator.	1 0 0 0 0 0 0 r	2	1					
MOVX @ Rr, A	((Rr)) ← (A); r = 0 – 1	Move indirect the contents of the accumulator into external memory.	1 0 0 1 0 0 0 r	2	1					
XCH A, Rr	(A) ↔ (Rr); r = 0 – 7	Exchange the accumulator and designated register's contents.	0 0 1 0 1 r r r	1	1					
XCH A, @ Rr	(A) ↔ ((Rr)); r = 0 – 1	Exchange indirect contents of accumulator and location in data memory.	0 0 1 0 0 0 0 r	1	1					
XCHD A, @ Rr	(A0 – 3) ↔ (Rr)(0 – 3); r = 0 – 1	Exchange indirect 4-bit contents of accumulator and data memory.	0 0 1 1 0 0 0 r	1	1					
Flags										
CPL C	(C) ← NOT (C)	Complement content of carry bit.	1 0 1 0 0 1 1 1	1	1	•				
CPL F0	(F0) ← NOT (F0)	Complement content of flag F0.	1 0 0 1 0 1 0 1	1	1			•		
CPL F1	(F1) ← NOT(F1)	Complement content of flag F1.	1 0 1 1 0 1 0 1	1	1				•	
CLR C	(C) ← 0	Clear content of carry bit to 0.	1 0 0 1 0 1 1 1	1	1	•				
CLR F0	(F0) ← 0	Clear content of flag 0 to 0.	1 0 0 0 0 1 0 1	1	1			•		
CLR F1	(F1) ← 0	Clear content of flag 1 to 0.	1 0 1 0 0 1 0 1	1	1					•

Table 1. INSTRUCTION SET (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	INSTRUCTION CODE D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	CYCLES	BYTES	FLAGS C	AC	F0	F1	BS
Input/output										
ANL BUS, # data	(BUS) ← (BUS) AND data	Logical AND immediate specified data with BUS.	1 0 0 1 1 0 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ANL Pp, # data	(Pp) ← (Pp) AND data p = 1 – 2	Logical AND immediate specified data with designated port (1 or 2).	1 0 0 1 1 0 P P d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ANLD Pp, A	(Pp) ← (Pp) AND (A 0 – 3) p = 4 – 7	Logical AND contents of accumulator with designated port (4 – 7).	1 0 0 1 1 1 p p	2	1					
IN A, Pp	(A) ← (Pp); p = 1 – 2	Input data from designated port (1 – 2) into accumulator.	0 0 0 0 1 0 p p	2	1					
INS A, BUS	(A) ← (BUS)	Input strobed BUS data into accumulator.	0 0 0 0 1 0 0 0	1	2					
MOVD A, Pp	(A 0 – 3) ← (Pp); p = 4 – 7 (A 4 – 7) ← 0	Move contents of designated port (4 – 7) into accumulator.	0 0 0 0 1 1 p p	2	1					
MOVD Pp, A	(Pp) ← A 0 – 3; p = 4 – 7	Move contents of accumulator designated port (4 – 7).	0 0 1 1 1 1 p p	1	1					
ORLD Pp, A	(Pp) ← (Pp) OR (A 0 – 3) p = 4 – 7	Logical OR contents of accumulator with designated port (4 – 7).	1 0 0 0 1 1 p p	1	1					
ORL BUS, # data	(BUS) ← (BUS) OR data.	Logical OR immediate specified data with BUS.	1 0 0 0 1 0 0 0 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
ORL Pp, # data	(Pp) ← (Pp) OR data p = 1 – 2	Logical OR immediate specified data with designated port (1 – 2).	1 0 0 0 1 0 p p d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
OUTL BUS, A	(BUS) ← (A)	Output contents of accumulator onto BUS.	0 0 0 0 0 0 1 0	1	2					
OUTL Pp, A	(Pp) ← c(A); p = 1 – 2	Output contents of accumulator to designated port (1 – 2).	0 0 1 1 1 0 p p	1	1					
Registers										
DEC Rr	(Rr) ← (Rr) – 1; r = 0 – 7	Decrement contents of designated register by 1.	1 1 0 0 1 r r r	1	1					
INC Rr	(Rr) ← (Rr) + 1; r = 0 – 7	Increment contents of designated register by 1.	0 0 0 1 1 r r r	1	1					
INC @ Rr	((Rr)) ← ((Rr)) + 1; r = 0 – 1	Increment indirect the contents of data memory location by 1.	0 0 0 1 0 0 0 r	1	1					
Subroutine										
CALL addr	((SP)) ← (PC), (PSW 4 – 7) (SP) ← (SP) + 1 (PC 8 – 10) ← addr 8 – 10 (PC0 – 7) ← addr 0 – 7 (PC 11) ← DBF	Call designated subroutine.	a_{10} a_9 a_8 1 0 1 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
RET	(SP) ← (SP) – 1 (PC) ← ((SP))	Return from subroutine without restoring program status word.	1 0 0 0 0 0 1 1	2	1					
RETR	(SP) ← (SP) – 1 (PC) ← ((SP)) (PSW 4 – 7) ← ((SP))	Return from subroutine restoring program status word.	1 0 0 1 0 0 1 1	2	1					
Timer/counter										
EN TCNTI		Enable timer/counter interrupt.	0 0 1 0 0 1 0 1	1	1					
DIS TCNTI		Disable timer/counter interrupt.	0 0 1 1 0 1 0 1	1	1					
MOV A, T	(A) ← (T)	Move contents of timer/counter into accumulator.	0 1 0 0 0 0 1 0	1	1					
MOV T, A	(T) ← (A)	Move contents of accumulator into timer/counter.	0 1 1 0 0 0 1 0	1	1					
STOP TCNT		Stop count for event counter or timer.	0 1 1 0 0 1 0 1	1	1					
STRT CNT		Start count for event counter.	0 1 0 0 0 1 0 1	1	1					
STRT T		Start count for timer.	0 1 0 1 0 1 0 1	1	1					
Miscellaneous										
NOP		No operation performed	0 0 0 0 0 0 0 0	1	1					
IDL		Enter idle mode	0 0 0 0 0 0 0 1	1	1					

NOTES:
1. Instruction code designations r and p form the binary representation of the registers and ports involved.
2. The dot under the appropriate flag bit indicates that its content is subject to change by the instruction it appears in.
3. Numerical subscripts appearing in the FUNCTION column reference the specific bits affected.

SYMBOL DEFINITIONS

SYMBOL	DESCRIPTION
A	The accumulator
AC	The auxiliary carry flag
addr	Program memory address (11 bits)
Bb	Bit designator (b = 0 – 7)
BS	The bank switch
C	Carry flag
CLK	Clock signal
CNT	Event counter
D	Nibble designator (4 bits)
DBF	Program memory bank flip-flop
data	Number or expression (8 bits)
F_0, F_1	Flags 0, 1
I	Interrupt
$\overline{INT}$	External interrupt

SYMBOL	DESCRIPTION
P	"In-Page" operation designator
Pp	Port designator (p = 1, 2 or 4 – 7)
PSW	Program status word
Rr	Register designator (r = 0, 1 or 0 – 7)
SP	Stack pointer
T	Timer
TF	Timer flag
T0, T1	Testable inputs 0, 1
#	Prefix for immediate data
@	Prefix for indirect address
$	Program counter's current value
←	Replaced by
↔	Exchanged with

3

Signetics

SCC80C31, SCC80C51
CMOS Single-Chip 8-Bit Microcontroller

Objective Specification

Microprocessor Products

DESCRIPTION

The Signetics SCC80C31/SCC80C51 is a high-performance microcontroller fabricated with Signetics' high-density CMOS technology. The CMOS 80C31/80C51 is functionally compatible with the NMOS 8031/8051 microcontroller. The Signetics CMOS technology combines the high speed and density characteristics of HMOS with the low power attributes of CMOS.

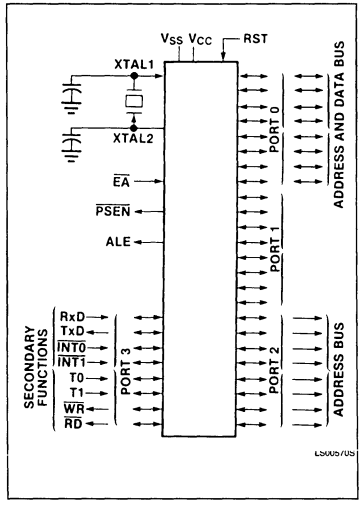
The SCC80C31/SCC80C51 contains a 4K x 8 ROM; a 128 x 8 RAM; 32 I/O lines; two 16-bit timer/counters; a five-source, two-priority level, nested interrupt structure; a serial I/O port for either multiprocessor communications, I/O expansion or full duplex UART; and on-chip oscillator and clock circuits. In addition, the 80C31/80C51 has two software selectable modes of reduced activity for further power reduction – idle and power down.

Idle mode freezes the CPU while allowing the RAM, timers, serial port, and interrupt system to continue functioning. Power-down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

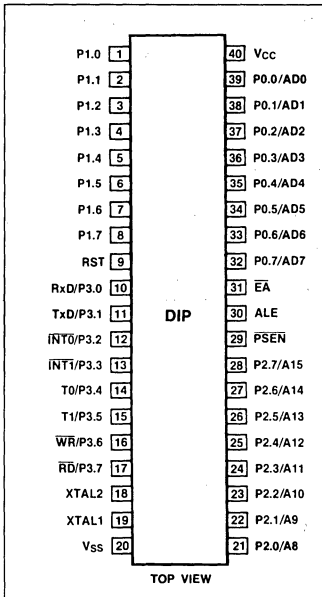
FEATURES

- **80C31 microcontroller without ROM**
- **80C51 control-oriented microcontroller**
- **8031/8051 compatible**
  - **4K x 8 ROM**
  - **128 x 8 RAM**
  - **Two 16-bit timer/counters**
  - **Full duplex serial channel**
  - **Boolean processor**
- **Memory addressing capability**
  - **64K ROM and 64K RAM**
- **Lower power consumption:**
  - **Normal operation: 24mA @ 5V, 12MHz**
  - **Idle mode: 3mA @ 5V, 12MHz**
  - **Power-down mode: 50μA @ 2V**
- **CMOS and TTL compatible**

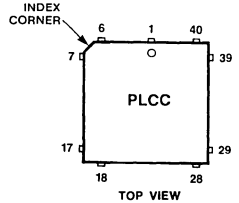
LOGIC SYMBOL



PIN CONFIGURATION



CD00362S



CD0044PS

Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0	24	A8/P2.0
3	P1.1	25	A9/P2.1
4	P1.2	26	A10/P2.2
5	P1.3	27	A11/P2.3
6	P1.4	28	A12/P2.4
7	P1.5	29	A13/P2.5
8	P1.6	30	A14/P2.6
9	P1.7	31	A15/P2.7
10	RST	32	PSEN
11	RxD/P3.0	33	ALE
12	NC	34	NC
13	TxD/P3.1	35	EA
14	$\overline{INT0}$/P3.2	36	P0.7/AD7
15	$\overline{INT1}$/P3.3	37	P0.6/AD6
16	T0/P3.4	38	P0.5/AD5
17	T1/P3.5	39	P0.4/AD4
18	$\overline{WR}$/P3.6	40	P0.3/AD3
19	$\overline{RD}$/P3.7	41	P0.2/AD2
20	XTAL2	42	P0.1/AD1
21	XTAL1	43	P0.0/AD0
22	VSS	44	VCC

CMOS Single-Chip 8-Bit Microcontroller — SCC80C31, SCC80C51

ORDERING CODE

SCC80C□□ A □ □ □ 40 (CPxxxx)

ROM/RAM (bytes)
31 = EXT/128
51 = 4K/128

OPERATING TEMPERATURE RANGE
C = 0°C to +70°C

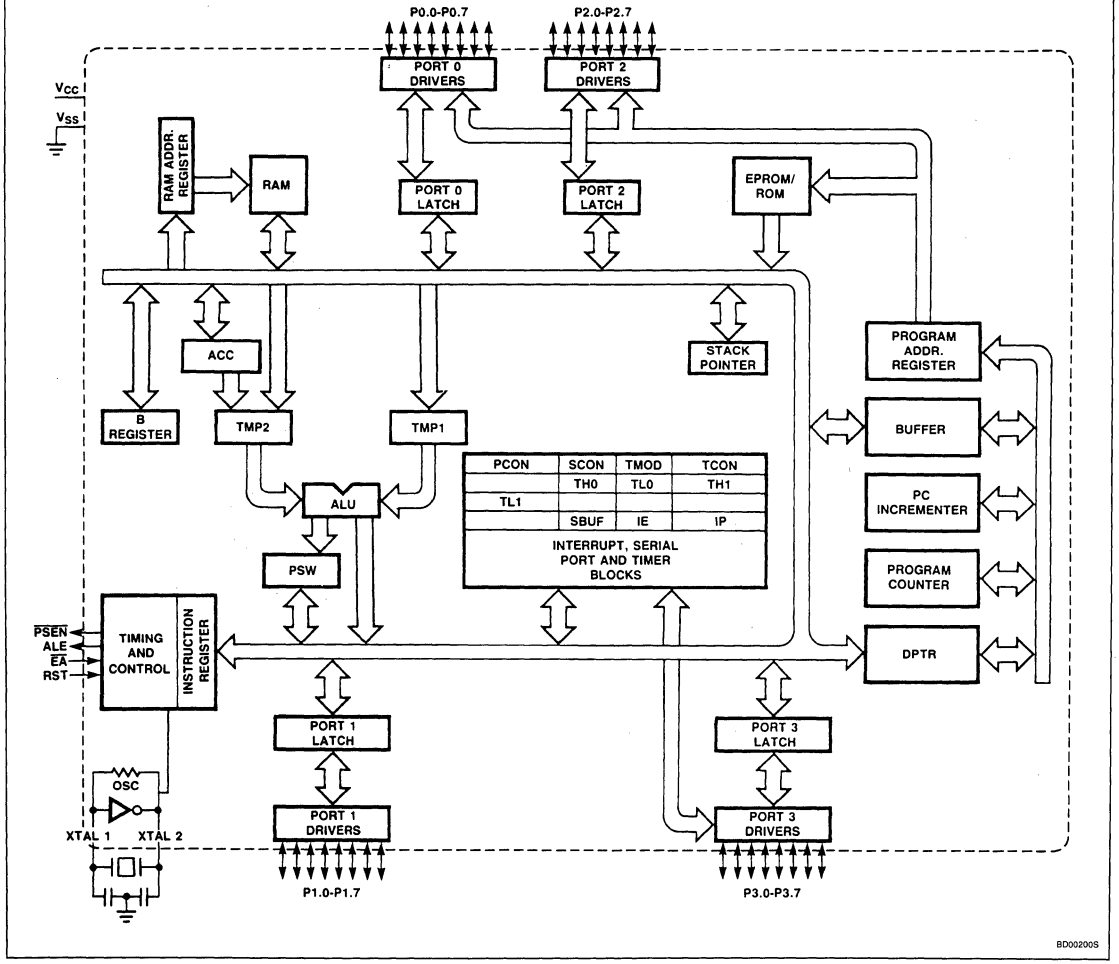
SPEED
C = 12MHz clock

CUSTOM ROM PATTERN NUMBER
Applies to masked ROM versions only. Number will be assigned by Signetics. Contact Signetics sales office for ROM pattern submission requirements.

40-Pin DIP
44-Pin LCC

PACKAGE
N = Plastic DIP
I = Ceramic DIP
A = Plastic LCC

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	Circuit ground potential.
V_{CC}	40	44	I	Supply voltage during normal, idle, and power-down operation.
P0.0–P0.7	39–32	43–36	I/O	**Port 0:** Port 0 is an 8-bit open-drain, bidirectional I/O port. It is also the multiplexed low-order address and data bus during accesses to external memory. It also outputs instruction bytes during program verification. External pullups are required during program verification. Port 0 can sink eight LS TTL inputs.
P1.0–P1.7	1–8	2–9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pullups. It receives the low-order address byte during program verification. In the 80C51, port 1 can sink/source three LS TTL inputs. It can drive CMOS inputs without external pullups.
P2.0–P2.7	21–18	24–31	I/O	**Port 2:** Port 2 Is an 8-bit bidirectional I/O port with internal pullups. It emits the high-order address byte during accesses to external memory. It also receives the high-order address bits and control signals during program verification in the 80C51. Port 2 can sink/source three LS TTL inputs. It can drive CMOS inputs without external pullups.
P3.0–P3.7	10–17	11, 13–19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pullups. Port 3 can sink/source three LS TTL inputs. It can drive CMOS inputs without external pullups. It also serves the functions of various special functions of the 80C51 family, as listed below:
	10	11	I	RXD (P3.0): Serial input port
	11	13	O	TXD (P3.1): Serial output port
	12	14	I	$\overline{INT0}$ (P3.2): External interrupt
	13	15	I	$\overline{INT1}$ (P3.3): External interrupt
	14	16	I	T0 (P3.4): Timer 0 external input
	15	17	I	T1 (P3.5): Timer 1 external input
	16	18	O	$\overline{WR}$ (P3.6): External data memory write strobe
	17	19	O	$\overline{RD}$ (P3.7): External data memory read strobe.
RST	9	10	I	**Reset:** A high level on this pin for two machines cycles while the oscillator is running resets the device. An internal pulldown resistor permits power-on reset using only a capacitor connected to the V_{CC}. This pin does not receive the power-down voltage. This function has been transferred to the V_{CC} pin.
ALE	30	33	O	**Address Latch Enable:** Output for latching the low byte of the address during accesses to external memory. ALE is activated at a constant rate of 1/6 the oscillator frequency except during an external data memory access, at which time one ALE pulse is skipped. ALE can sink/source eight LS TTL inputs. It can drive CMOS inputs without an external pullup.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** Output is the read strobe to external program memory. $\overline{PSEN}$ is activated twice each machine cycle during fetches from external program memory. (However, when executing out of external program memory, two activations of $\overline{PSEN}$ are skipped during each access to external data memory.) $\overline{PSEN}$ is not activated during fetches from internal program memory. $\overline{PSEN}$ can sink/source eight LS TTL inputs. It can drive CMOS inputs without an external pullup.
$\overline{EA}$	31	35	I	**Instruction Execution Control:** When $\overline{EA}$ is held high, the CPU executes out of internal program memory (unless the program counter exceeds 0FFFH). When $\overline{EA}$ is held low, the CPU executes only out of external program memory. $\overline{EA}$ must not be floated.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting amplifier that forms the oscillator. Receives the external oscillator signal when an external oscillator is used.
XTAL2	18	20	O	**Crystal 2:** Output of the inverting amplifier that forms the oscillator, and input to the internal clock generator. This pin should be floated when an external oscillator is used.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
Voltage from V_{CC} to V_{SS}[3]	−0.5 to +7.0	V
Voltage from any pin to V_{SS}[3]	−0.5 to V_{CC}+0.5	V
Power dissipation	2.0	W

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 4.0V$ to $6.0V$, $V_{SS} = 0V$[4,5]

PARAMETER		TEST CONDITIONS	LIMITS Min	LIMITS Max	UNIT
V_{IL}	Input low voltage	$V_{CC} > 4.5V$ $V_{CC} < 4.5V$	−0.5 −0.5	0.8 0.5	V
V_{IH}	Input high voltage (except XTALs and RST)	$V_{CC} < 5.5V$ $V_{CC} > 5.5V$	2.0 2.5	V_{CC} V_{CC}	V
V_{IH1}	Input high voltage to RST for reset	$V_{CC} < 5.5V$ $V_{CC} > 5.5V$	3.0 3.5	V_{CC} V_{CC}	V
V_{IH2}	Input high voltage to XTAL1 and XTAL2		0.8 V_{CC}	V_{CC}	V
V_{PD}	Power down voltage to V_{CC} in PD mode		2.0	6.0	V
V_{OL}[6]	Output low voltage ports 1, 2, 3	$I_{OL} = 1.6mA$		0.45	V
V_{OL1}[6]	Output low voltage port 0, ALE, $\overline{PSEN}$	$I_{OL} = 3.2mA$		0.45	V
V_{OH}	Output high voltage ports 1, 2, 3	$I_{OH} = 10\mu A$ $I_{OH} = -80\mu A, V_{CC} = 5V \pm 10\%$	0.9V_{CC} 2.4		V
V_{OH1}	Output high voltage port 0 (in external bus mode), ALE, $\overline{PSEN}$	$I_{OH} = -40\mu A$ $I_{OH} = 400\mu A, V_{CC} = 5V \pm 10\%$	0.9 V_{CC} 2.4		V
I_{IL}	Logical 0 input current ports 1, 2, 3	$V_{IN} = 0.45V$		−50	μA
I_{LI}	Input leakage current	$V_{IN} = V_{SS}$ or $V_{IN} = V_{CC}$		±10	μA
I_{CC}	Power supply current normal operations[8]	$f_{OSC} = 12MHz$ $f_{OSC} = 1.2MHz$		24 2.4	mA
I_{CC1}	Idle Mode[8]	$f_{OSC} = 12MHz$ $f_{OSC} = 1.2MHz$		3.0 0.3	mA
I_{CC2}	Power supply current (power-down mode)[8]	$V_{CC} = 2V$		50	μA
R_{RST}	RST pulldown resistor		4		kΩ
C_{IO}	Capacitance of I/O buffer	$f_C = 1MHz$		10	pF

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying voltages greater than the rated maxima.
4. Parameters are valid over operating temperature range unle ss otherwise specified.
5. All voltage measurements are referenced to ground. For tes ting, all input signals swing between 0.45V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and at output voltages of 0.8V and 2.0V as appropriate.
6. V_{OL} is degraded when the 80C31/80C51 rapidly discharg es external capacitance. This AC noise is most pronounced during emission of address data. When using external memory, locate the latch or buffer as close to the 80C31/80C51 as possible.

Datum	Emitting Ports	Degraded I/O Lines	V_{OL}(Peak Max)
Address	P2, P0	P1, P3	0.8V
Write data	P0	P1, P3, ALE	0.8V

7. $C_L = 100pF$ for port 0, ALE and $\overline{PSEN}$ outputs; $C_L = 80pF$ for all other ports.
8. All outputs connected: $V_{CC} = 5V$; rise and fall times of clock drive < 10ns.

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 20\%$, $V_{SS} = 0V$[4,5,7]

PARAMETER		VARIABLE CLOCK $1/t_{CLCL}$ = 1.2MHz to 12MHz		UNIT
		Min	Max	
Program memory char (figure 1)				
t_{LHLL}	ALE pulse width	$2t_{CLCL} - 40$		ns
t_{AVLL}	Address valid to ALE	$t_{CLCL} - 40$		ns
t_{LLAX}	Address hold after ALE	$t_{CLCL} - 35$		ns
t_{LLIV}	ALE to valid instr in		$4t_{CLCL} - 100$	ns
t_{LLPL}	ALE to $\overline{PSEN}$	$t_{CLCL} - 25$		ns
t_{PLPH}	$\overline{PSEN}$ pulse width	$3t_{CLCL} - 35$		ns
t_{PLIV}	$\overline{PSEN}$ to valid instr in		$3t_{CLCL} - 125$	ns
t_{PXIX}	Input instr hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	Input instr float after $\overline{PSEN}$		$t_{CLCL} - 20$	ns
t_{PXAV}	$\overline{PSEN}$ to address valid	$t_{CLCL} - 8$		ns
t_{AVIV}	Address to valid instr in		$5t_{CLCL} - 115$	ns
t_{PLAZ}	$\overline{PSEN}$ to address float		0	ns
External data memory char (figures 2 and 3)				
t_{RLRH}	$\overline{RD}$ pulse width	$6t_{CLCL} - 100$		ns
t_{WLWH}	$\overline{WR}$ pulse width	$6t_{CLCL} - 100$		ns
t_{LLAX}	Address hold after ALE	$t_{CLCL} - 35$		ns
t_{RLDV}	$\overline{RD}$ to valid data in		$5t_{CLCL} - 165$	ns
t_{RHDX}	Data hold after $\overline{RD}$	0		ns
t_{RHDZ}	Data float after $\overline{RD}$		$2t_{CLCL} - 70$	ns
t_{LLDV}	ALE to valid data in		$8t_{CLCL} - 150$	ns
t_{AVDV}	Address to valid data in		$9t_{CLCL} - 165$	ns
t_{LLWL}	ALE to $\overline{WR}$ or $\overline{RD}$	$3t_{CLCL} - 50$	$3t_{CLCL} + 50$	ns
t_{AVWL}	Address to $\overline{WR}$ or $\overline{RD}$	$4t_{CLCL} - 130$		ns
t_{DVWX}	Data valid to $\overline{WR}$ transition	$t_{CLCL} - 60$		ns
t_{QVWH}	Data set-up to $\overline{WR}$ high	$7t_{CLCL} - 150$		ns
t_{WHQX}	Data hold after $\overline{WR}$	$t_{CLCL} - 50$		ns
t_{RLAZ}	$\overline{RD}$ low to address float		0	ns
t_{WHLH}	$\overline{WR}$ or $\overline{RD}$ high to ALE high	$t_{CLCL} - 50$	$t_{CLCL} + 50$	ns
External clock (figure 4)				
t_{CLCL}	Oscillator period	83.3	833	ns
t_{CHCX}	High time	20		ns
t_{CLCX}	Low time	20		ns
t_{CLCH}	Rise time		20	ns
t_{CHCL}	Fall time		20	ns

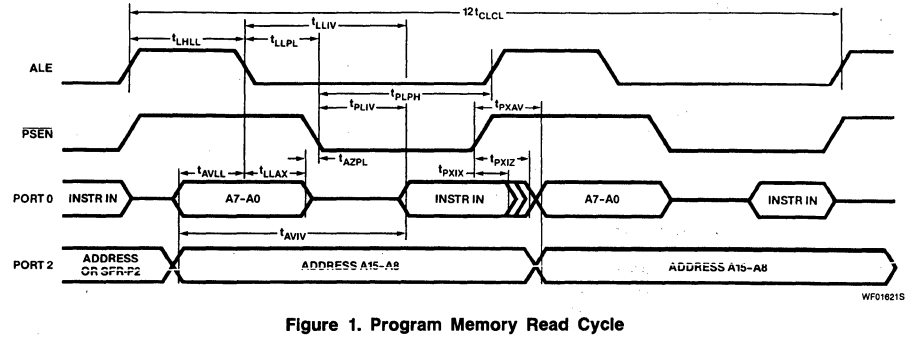


Figure 1. Program Memory Read Cycle

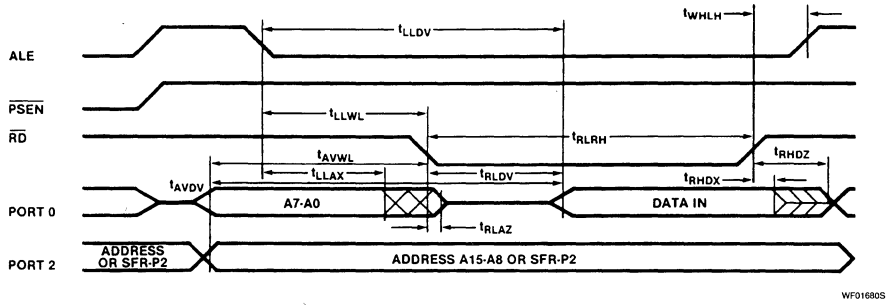


Figure 2. Data Memory Read Cycle

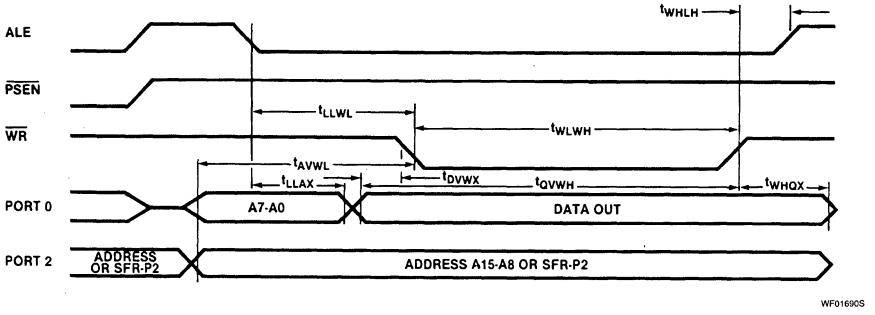


Figure 3. Data Memory Write Cycle

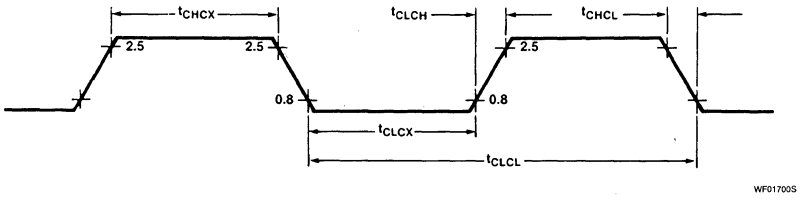


Figure 4. External Clock Drive XTAL1

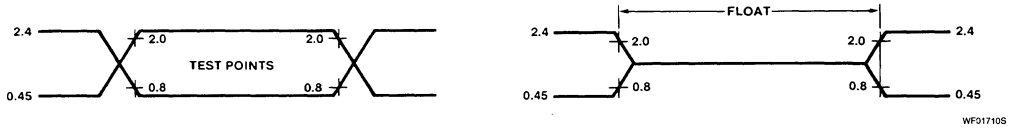


Figure 5. AC Testing Input/Output, Float Waveforms

NOTES:
1. AC testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0".
2. Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".
3. For timing purposes, the float state is defined as the point at which a P0 pin sinks 3.2mA or sources 400μA at the voltage test levels.

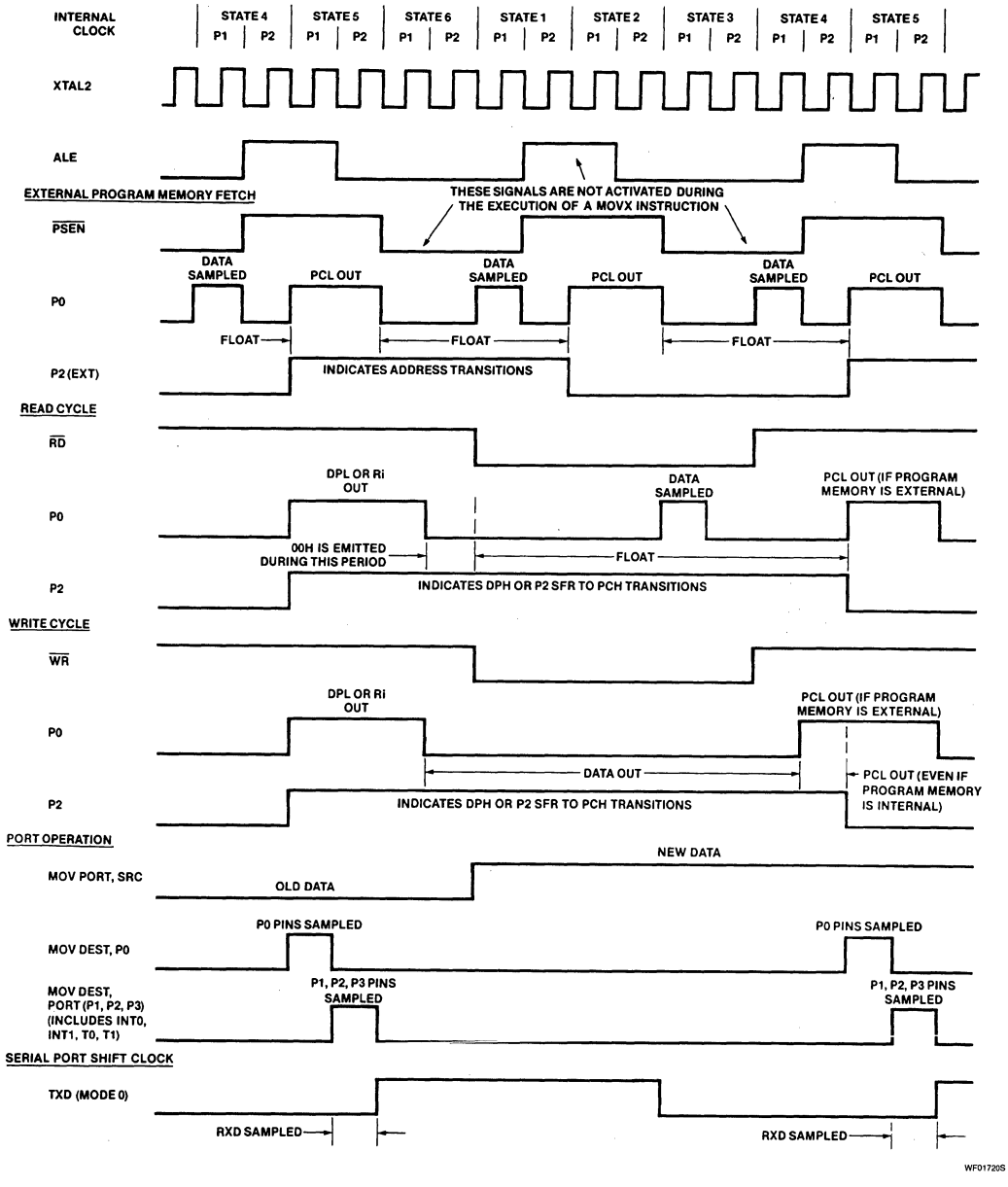


Figure 6. Timing Waveforms

NOTE:
This diagram indicates when signals are clocked internally. The time it takes the signals to propagate to the pins, however, ranges from 25ns to 125ns. This propagation is dependent on variables such as temperature and pin loading. Propagation also varies from output to output and component to component. Typically though (T_A = 25°C, fully loaded), $\overline{RD}$ and $\overline{WR}$ propagation delays are approximately 50ns. The other signals are typically 85ns. Propagation delays are incorporated in the AC specifications.

Table 1. INSTRUCTION SET

MNEMONIC		DESCRIPTION	BYTE	CYCLES
Arithmetic operations				
ADD	A,Rn	Add register to accumulator	1	1
ADD	A,direct	Add direct byte to accumulator	2	1
ADD	A,@Ri	Add indirect RAM to accumulator	1	1
ADD	A,R#data	Add immediate data to accumulator	2	1
ADDC	A,Rn	Add register to accumulator with carry	1	1
ADDC	A,direct	Add direct byte to A with carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with carry flag	1	1
ADDC	A,#data	Add immediate data to A with carry flag	2	1
SUBB	A,Rn	Subtract register from A with borrow	1	1
SUBB	A,direct	Subtract direct byte from A with borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A w/borrow	1	1
SUBB	A,#data	Subtract immed. data from A w/borrow	2	1
INC	A	Increment accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement Indirect RAM	1	1
INC	DPTR	Increment data pointer	1	2
MUL	AB	Mulitiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal adjust accumulator	1	1
Logical operations				
ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive-OR register to accumulator	1	1
XRL	A,direct	Exclusive-OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive-OR indirect RAM to A	1	1
XRL	A,#data	Exclusive-OR immediate data to A	2	1
XRL	direct,A	Exclusive-OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear accumulator	1	1
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate A left through the carry flag	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate A right through carry flag	1	1
SWAP	A	Swap nibbles within the accumulator	1	1
Data transfer				
MOV	A,Rn	Move register to accumulator	1	1
MOV	A,direct	Move direct byte to accumulator	2	1
MOV	A,@Ri	Move indirect RAM to accumulator	1	1
MOV	A,#data	Move immediate data to accumulator	2	1
MOV	Rn,A	Move accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1

Table 1. INSTRUCTION SET (Continued)

MNEMONIC		DESCRIPTION	BYTE	CYCLES
Data transfer (continued)				
MOV	DPTR,#data16	Load data pointer with a 16-bit constant	3	2
MOVC	A,@A + DPTR	Move code byte relative to DPTR to A	1	2
MOVC	A,@A + PC	Move code byte relative to PC to A	1	2
MOVX	A,@Ri	Move external RAM (8-bit addr) to A	1	2
MOVX	A,@DPTR	Move external RAM (16-bit addr) to A	1	2
MOVX	@Ri,A	Move A to external RAM (8-bit addr)	1	2
MOVX	@DPTR,A	Move A to external RAM (16-bit addr)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with accumulator	1	1
XCH	A,direct	Exchange direct byte with accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with A	1	1
XCHD	A,@Ri	Exchange low-order digit ind. RAM w/A	1	1
Boolean variable manipulation				
CLR	C	Clear carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set carry flag	1	1
SETB	bit	Set direct bit	2	1
CPL	C	Complement carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to carry flag	2	2
ANL	C,/bit	AND complement of direct bit to carry	2	2
ORL	C,bit	OR direct bit to carry flag	2	2
ORL	C,/bit	OR complement of direct bit to carry	2	2
MOV	C,bit	Move direct bit to carry flag	2	1
MOV	bit,C	Move carry flag to direct bit	2	2
Program and machine control				
ACALL	addr11	Absolute subroutine call	2	2
LCALL	addr16	Long subroutine call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute jump	2	2
LJMP	addr16	Long jump	3	2
SJMP	rel	Short jump (relative addr)	2	2
JMP	@A + DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if accumulator is zero	2	2
JNZ	rel	Jump if accumulator is not zero	2	2
JC	rel	Jump if carry flag is set	2	2
JNC	rel	Jump if no carry flag	2	2
JB	bit,rel	Jump if direct bit set	3	2
JNB	bit,rel	Jump if direct bit not set	3	2
JBC	bit,rel	Jump if direct bit is set & clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & jump if not equal	3	2
CJNE	A,#data,rel	Comp. immed. to A & jump if not equal	3	2
CJNE	Rn,#data,rel	Comp. immed. to reg. & jump if not equal	3	2
CJNE	@Ri,#data,rel	Comp. immed. to ind. & jump if not equal	3	2
DJNZ	Rn,rel	Decrement register & jump if not zero	2	2
DJNZ	direct,rel	Decrement direct & jump if not zero	3	2
NOP		No operation	1	1

Notes on data addressing modes:

Rn	– Working register R0-R7
direct	– 128 internal RAM locations, any I/O port, control or status register
@Ri	– Indirect Internal RAM location addressed by register R0 or R1
#data	– 8-bit constant included in instruction
#data16	– 16-bit constant included as bytes 2 & 3 of instruction
bit	– 128 software flags, any I/O pin, control or status bit

Notes on program addressing modes:

addr16	– Destination address for LCALL & LJMP may be anywhere within the 64-kilobyte program memory address space.
addr11	– Destination address for ACALL & AJMP will be within the same 2-kilobyte page of program memory as the first byte of the following instruction.
rel	– SJMP and all conditional jumps include an 8-bit offset byte. Range is +127 – 128 bytes relative to first byte of the following instruction.

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A,#data
25	2	ADD	A,data addr
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr,code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A,#data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Continued)

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr,#data
44	2	ORL	A,#data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr,#data
54	2	ANL	A,#data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr,A
63	3	XRL	data addr,#data
64	2	XRL	A,#data
65	2	XRL	A,data addr
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data
79	2	MOV	R1,#data
7A	2	MOV	R2,#data
7B	2	MOV	R3,#data
7C	2	MOV	R4,#data
7D	2	MOV	R5,#data
7E	2	MOV	R6,#data

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Continued)

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
7F	2	MOV	R7,#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A + PC
84	1	DIV	AB
85	3	MOV	data addr,data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR,#data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A + DPTR
94	2	SUBB	A,#data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0
97	1	SUBB	A,@R1
98	1	SUBB	A,R0
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,/bit addr
A1	2	AJMP	code addr
A2	2	MOV	C,bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,data addr
A7	2	MOV	@R1,data addr
A8	2	MOV	R0,data addr
A9	2	MOV	R1,data addr
AA	2	MOV	R2,data addr
AB	2	MOV	R3,data addr
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,/bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A,#data,code addr
B5	3	CJNE	A,data addr,code addr
B6	3	CJNE	@R0,#data,code addr
B7	3	CJNE	@R1,#data,code addr
B8	3	CJNE	R0,#data,code addr
B9	3	CJNE	R1,#data,code addr
BA	3	CJNE	R2,#data,code addr
BB	3	CJNE	R3,#data,code addr
BC	3	CJNE	R4,#data,code addr
BD	3	CJNE	R5,#data,code addr
BE	3	CJNE	R6,#data,code addr
BF	3	CJNE	R7,#data,code addr

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Continued)

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr,code addr
D6	1	XCHD	A,@R0
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,code addr
D9	2	DJNZ	R1,code addr
DA	2	DJNZ	R2,code addr
DB	2	DJNZ	R3,code addr
DC	2	DJNZ	R4,code addr
DD	2	DJNZ	R5,code addr
DE	2	DJNZ	R6,code addr
DF	2	DJNZ	R7,code addr
E0	1	MOVX	A,@DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A,@R0
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,data addr
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	data addr,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A

SCN8031AH, SCN8051AH
Single-Chip 8-Bit Microcontroller

Product Specification

Microprocessor Products

DESCRIPTION

The Signetics SCN8031AH/8051AH is a high-performance microcontroller fabricated using Signetics' highly reliable +5V, depletion-load, N-channel, silicon-gate, N500 MOS process technology. It provides the hardware features, architectural enhancements and new instructions that are necessary to make it a powerful and cost-effective controller for applications requiring up to 64K bytes of program memory and/or up to 64K bytes of data storage.

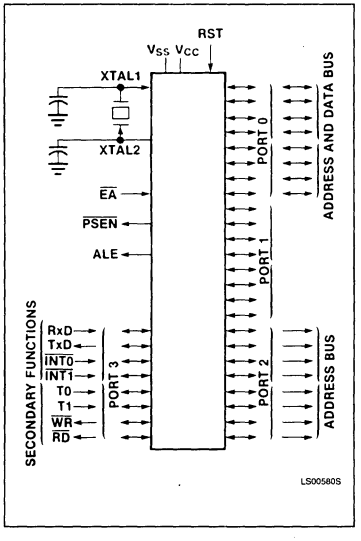
The SCN8051AH contains a 4K x 8 read-only program memory; a 128 x 8 read/write data memory; 32 I/O lines; two 16-bit timer/counters; a five-source, two-priority-level, nested interrupt structure; a serial I/O port for either multiprocessor communications, I/O expansion, or full duplex UART; and on-chip oscillator and clock circuits. The SCN8031AH is identical, except that it lacks the program memory. For systems that require extra capability, the SCN8051AH can be expanded using standard TTL compatible memories and byte oriented peripheral controllers.

The SCN8051AH microcontroller, like its SCN8048 predecessor, is efficient both as a controller and as an arithmetic processor. It has extensive facilities for binary and BCD arithmetic and excels in bit-handling capabilities. Efficient use of program memory results from an instruction set consisting of 44% one-byte, 41% two-byte, and 15% three-byte instructions. With a 12MHz crystal, 58% of the instructions execute in 1μs, 40% in 2μs and multiply and divide require only 4μs. Among the many instructions added to the standard SCN8048 instruction set are multiply, divide, subtract, and compare.

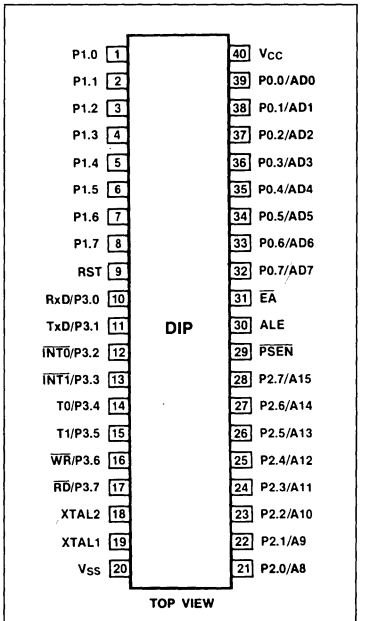
FEATURES

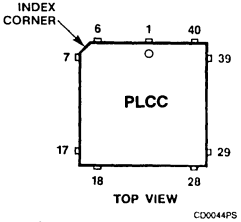
- **Reduced supply current**
- **4K x 8 ROM (SCN8051AH)**
- **128 x 8 RAM**
- **Four 8-bit ports, 32 I/O lines**
- **Two 16-bit timer/event counters**
- **High-performance full-duplex serial channel**
- **External memory expandable to 128K**
- **Boolean processor**
- **SCN80 series architecture enhanced with:**
  - **Non-paged jumps**
  - **Direct addressing**
  - **Four 8-register banks**
  - **Stack depth up to 128-bytes**
  - **Multiply, divide, subtract, compare**
- **Most instructions execute in 1μs**
- **4μs multiply and divide**

LOGIC SYMBOL



PIN CONFIGURATION





Pin	Function	Pin	Function
1	NC	23	NC
2	P1.0	24	A8/P2.0
3	P1.1	25	A9/P2.1
4	P1.2	26	A10/P2.2
5	P1.3	27	A11/P2.3
6	P1.4	28	A12/P2.4
7	P1.5	29	A13/P2.5
8	P1.6	30	A14/P2.6
9	P1.7	31	A15/P2.7
10	RST	32	$\overline{PSEN}$
11	RxD/P3.0	33	ALE
12	NC	34	NC
13	TxD/P3.1	35	$\overline{EA}$
14	$\overline{INT0}$/P3.2	36	AD7/P0.7
15	$\overline{INT1}$/P3.3	37	AD6/P0.6
16	T0/P3.4	38	AD5/P0.5
17	T1/P3.5	39	AD4/P0.4
18	$\overline{WR}$/P3.6	40	AD3/P0.3
19	$\overline{RD}$/P3.7	41	AD2/P0.2
20	XTAL2	42	AD1/P0.1
21	XTAL1	43	AD0/P0.0
22	VSS	44	VCC

ORDERING CODE

SCN80□□ H □ □ □ 40 (CPxxxx)

ROM/RAM (bytes)
31 = EXT/128
51 = 4K/128

POWER CONSUMPTION
H = Reduced Power AH

OPERATING TEMPERATURE RANGE
A = −40°C to +85°C
C = 0°C to +70°C

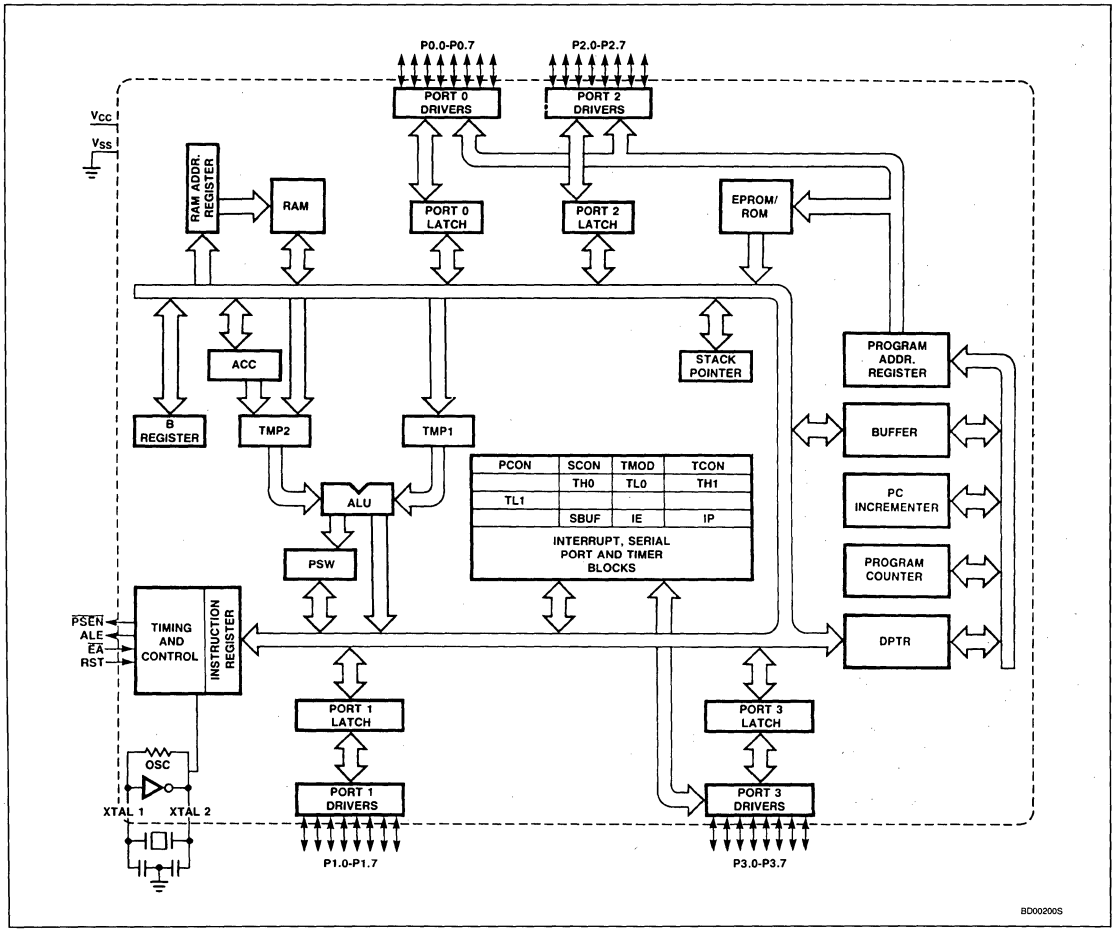
SPEED
C = 12MHz clock

CUSTOM ROM PATTERN NUMBER
Applies to masked ROM versions only. Number will be assigned by Signetics. Contact Signetics sales office for ROM pattern submission requirements.

40-PIN DIP
44-PIN PLCC

PACKAGE
N = Plastic DIP
I = Ceramic DIP
A = Plastic LCC

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	Circuit ground potential.
V_{CC}	40	44	I	Supply voltage during normal operation and program verification.
P0.0 – P0.7	39 – 32	43 – 36	I/O	**Port 0:** Port 0 is an 8-bit open-drain, bidirectional I/O port. It is also the multiplexed low-order address and data bus during accesses to external memory. It also outputs instruction bytes during program verification. External pullups are required during program verification. Port 0 can sink (and in bus operation source) eight LS TTL inputs.
P1.0 – P1.7	1 – 8	2 – 9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pullups. It receives the low-order address byte during program verification. Port 1 can sink/source three LS TTL inputs.
P2.0 – P2.7	21 – 28	24 – 31	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pullups. It emits the high-order address byte during accesses to external memory. It also receives the high-order address bits and control signals during program verification. Port 2 can sink/source three LS TTL inputs.
P3.0 – P3.7	10 – 17	11, 13 – 19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pullups. Port 3 can sink/source three LS TTL inputs. It also serves the functions of various special features as listed below. The output latch corresponding to a secondary function must be programmed to a logic 1 for that function to operate.
	10	11	I	RXD (P3.0): Serial input port
	11	13	O	TXD (P3.1): Serial output port
	12	14	I	$\overline{INT0}$ (P3.2): External interrupt 0
	13	15	I	$\overline{INT1}$ (P3.3): External interrupt 1
	14	16	I	T0 (P3.4): Timer 0 external input
	15	17	I	T1 (3.5): Timer 1 external input
	16	18	O	$\overline{WR}$ (P3.6): External data memory write strobe
	17	19	O	$\overline{RD}$ (P3.7): External data memory read strobe
RST	9	10	I	**Reset:** A high level on this pin for two machine cycles while the oscillator is running resets the device. An external pulldown resistor (≈8.2K) from reset to V_{SS} permits power-on reset using only a capacitor (≈10μF) connected from this pin to V_{CC}.
ALE	30	33	O	**Address Latch Enable:** Output for latching the low byte of the address during accesses to external memory. ALE is activated at a constant rate of 1/6 the oscillator frequency except during an external data memory access, at which time one ALE pulse is skipped. ALE can sink/source eight LS TTL inputs.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** Output is the read strobe to external program memory. $\overline{PSEN}$ is activated twice each machine cycle during fetches from external program memory. (However, when executing out of external program memory, two activations of $\overline{PSEN}$ are skipped during each access to external data memory.) $\overline{PSEN}$ is not activated during fetches from internal program memory. $\overline{PSEN}$ can sink/source eight LS TTL inputs.
$\overline{EA}$	31	35	I	**Instruction Execution Control:** When $\overline{EA}$ is held high, the CPU executes out of internal program memory (unless the program counter exceeds OFFFH). When $\overline{EA}$ is held low, the CPU executes only out of external program memory. $\overline{EA}$ must not be floated.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting amplifier that forms the oscillator. This pin should be connected to V_{SS} when an external oscillator is used.
XTAL2	18	20	O	**Crystal 2:** Output of the inverting amplifier that forms the oscillator, and input to the internal clock generator. XTAL2 receives the oscillator signal when an external oscillator is used.

ABSOLUTE MAXIMUM RATINGS

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +7.0	V
Power dissipation	1.0	W

DC ELECTRICAL CHARACTERISTICS T_A=0°C to +70°C, V_{CC} = 4.5 to 5.5V, V_{SS} = 0V[4,5]

PARAMETER		TEST CONDITIONS	LIMITS Min	LIMITS Max	UNIT
V_{IL}	Input low voltage		−0.5	0.8	V
V_{IH}	Input high voltage (except RST and XTAL2)		2.0	V_{CC} + 0.5	V
V_{IH1}	Input high voltage to RST for reset, XTAL2	XTAL1 to V_{SS}	2.5	V_{CC} + 0.5	V
V_{OL}	Output low voltage ports 1, 2, 3[6]	I_{OL} = 1.6mA		0.45	V
V_{OL1}	Output low voltage port 0, ALE, $\overline{PSEN}$[6]	I_{OL} = 3.2mA		0.45	V
V_{OH}	Output high voltage ports 1, 2, 3	I_{OH} = −80μA	2.4		V
V_{OH1}	Output high voltage port 0, ALE $\overline{PSEN}$	I_{OH} = −400μA	2.4		V
I_{IL}	Logical 0 input current ports 1, 2, 3	V_{IN} = 0.45V		−800	μA
I_{IH1}	Input high current to RST for reset	$V_{IN} < V_{CC} - 1.5V$		500	μA
I_{LI}	Input leakage current to port 0, $\overline{EA}$	$0.45 < V_{IN} < V_{CC}$		±10	μA
I_{IL2}	Logical 0 input current for XTAL2	XTAL1 = V_{SS}, V_{IN} = 0.45V		−2.5	mA
I_{CC}	Power supply current	All outputs disconnected		125	mA
C_{IO}	Capacitance of I/O buffer	f_C = 1MHz, T_A = 25°C		10	pF

T_A = −40°C to +85°C – Extended temperature range – SCN8051HAC only

V_{IH}	Input high voltage (except RST and XTAL2)		2.2	V_{CC}+0.5	V
V_{IH1}	Input high voltage to RST for reset, XTAL2	XTAL1 to V_{SS}	2.7		V
I_{CC}	Power supply current	All outputs disconnected		175	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any conditions other than those described in the AC and DC Electrical Charactoristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging offects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying voltages greater than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground. For testing, all input signals swing between 0.45V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and at output voltages of 0.8V and 2.0V as appropriate.
6. V_{OL} is degraded when the SCN8031AH/SCN8051AH rapidly disch arges external capacitance. This AC noise is most pronounced during omission of address data. When using external memory, locate the latch or buffer as close to the SCN8031AH/SCN8051AH as possible.

Datum	Emitting Ports	Degraded I/O Lines	V_{OL}(Peak Max)
Address	P2, P0	P1, P3	0.8V
Write data	P0	P1, P3, ALE	0.8V

7. C_L = 100pF for port 0, ALE and PSEN outputs; C_L = 80pF for all other ports.

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$[4,5,7]

PARAMETER		12MHz CLOCK		VARIABLE CLOCK $1/t_{CLCL}$ = 3.5MHz to 12MHz		UNIT
		Min	Max	Min	Max	
Program memory char (fig. 1)						
t_{LHLL}	ALE pulse width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	Address set-up to ALE	43		$t_{CLCL}-40$		ns
t_{LLAX}	Address hold after ALE	48		$t_{CLCL}-35$		ns
t_{LLIV}	ALE to valid instr in		223		$4t_{CLCL}-100$	ns
t_{LLPL}	ALE to $\overline{PSEN}$	58		$t_{CLCL}-25$		ns
t_{PLPH}	$\overline{PSEN}$ pulse width	215		$3t_{CLCL}-35$		ns
t_{PLIV}	$\overline{PSEN}$ to valid instr in		125		$3t_{CLCL}-125$	ns
t_{PXIX}	Input instr hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	Input instr float after $\overline{PSEN}$		63		$t_{CLCL}-20$	ns
t_{PXAV}	Address valid after $\overline{PSEN}$	75		$t_{CLCL}-8$		ns
t_{AVIV}	Address to valid instr in		302		$5t_{CLCL}-115$	ns
t_{PLAZ}	$\overline{PSEN}$ to address float		25		25	ns
External data memory char (fig. 2 and 3)						
t_{RLRH}	$\overline{RD}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	$\overline{WR}$ pulse width	400		$6t_{CLCL}-100$		ns
t_{LLAX}	Address hold after ALE	48		$t_{CLCL}-35$		ns
t_{RLDV}	$\overline{RD}$ to valid data in		250		$5t_{CLCL}-165$	ns
t_{RHDX}	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	Data float after $\overline{RD}$		97		$2t_{CLCL}-70$	ns
t_{LLDV}	ALE to valid data in		517		$8t_{CLCL}-150$	ns
t_{AVDV}	Address to valid data in		585		$9t_{CLCL}-165$	ns
t_{LLWL}	ALE to $\overline{WR}$ or $\overline{RD}$	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	Address to $\overline{WR}$ or $\overline{RD}$	203		$4t_{CLCL}-130$		ns
t_{WHLH}	$\overline{WR}$ or $\overline{RD}$ high to ALE high	43	123	$t_{CLCL}-40$	$t_{CLCL}+40$	ns
t_{DVWX}	Data valid to $\overline{WR}$ transition	23		$t_{CLCL}-60$		ns
t_{QVWH}	Data set-up before $\overline{WR}$	433		$7t_{CLCL}-150$		ns
t_{WHQX}	Data hold after $\overline{WR}$	33		$t_{CLCL}-50$		ns
t_{RLAZ}	Address float after $\overline{RD}$		25		25	ns
External clock (fig. 4)						
t_{CLCL}	Oscillator period			83.3	286	ns
t_{CHCX}	High time			20		ns
t_{CLCX}	Low time			20		ns
t_{CLCH}	Rise time				20	ns
t_{CHCL}	Fall time				20	ns

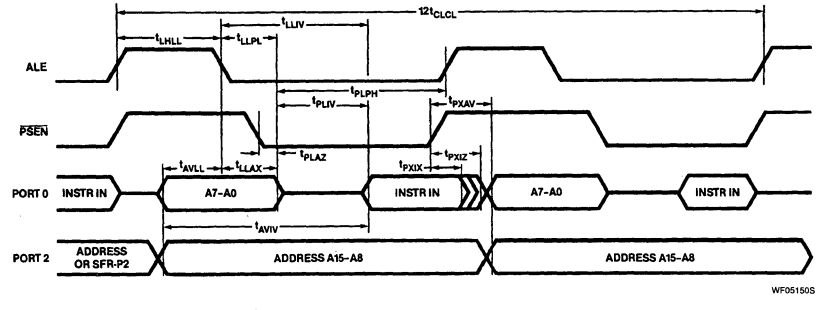


Figure 1. Program Memory Read Cycle

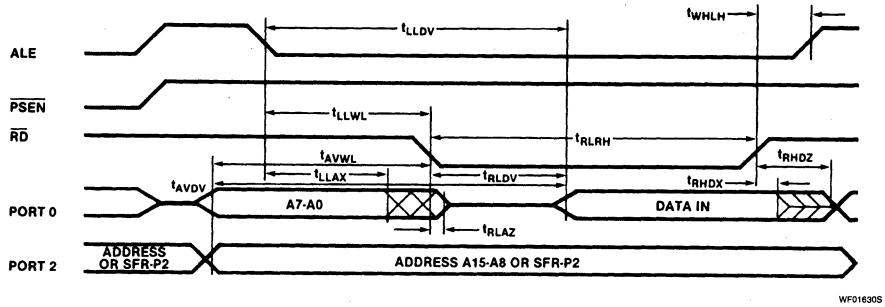


Figure 2. Data Memory Read Cycle

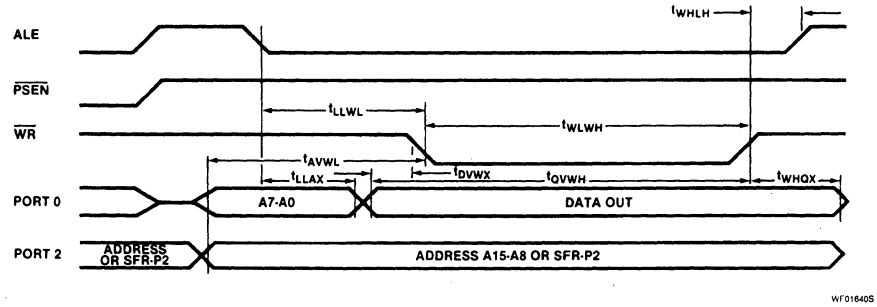


Figure 3. Data Memory Write Cycle

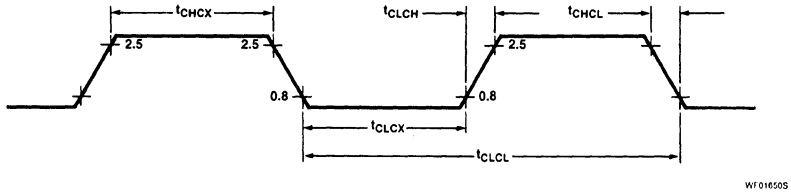


Figure 4. External Clock Drive XTAL2

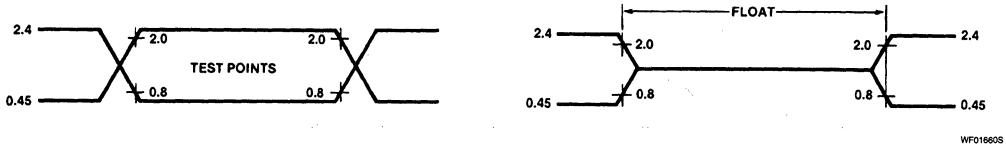


Figure 5. AC Testing Input/Output, Float Waveforms

NOTES:
1. AC testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0".
2. Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".
3. For timing purposes, the float state is defined as the point at which a P0 pin sinks 3.2mA or sources 400μA at the voltage test levels.

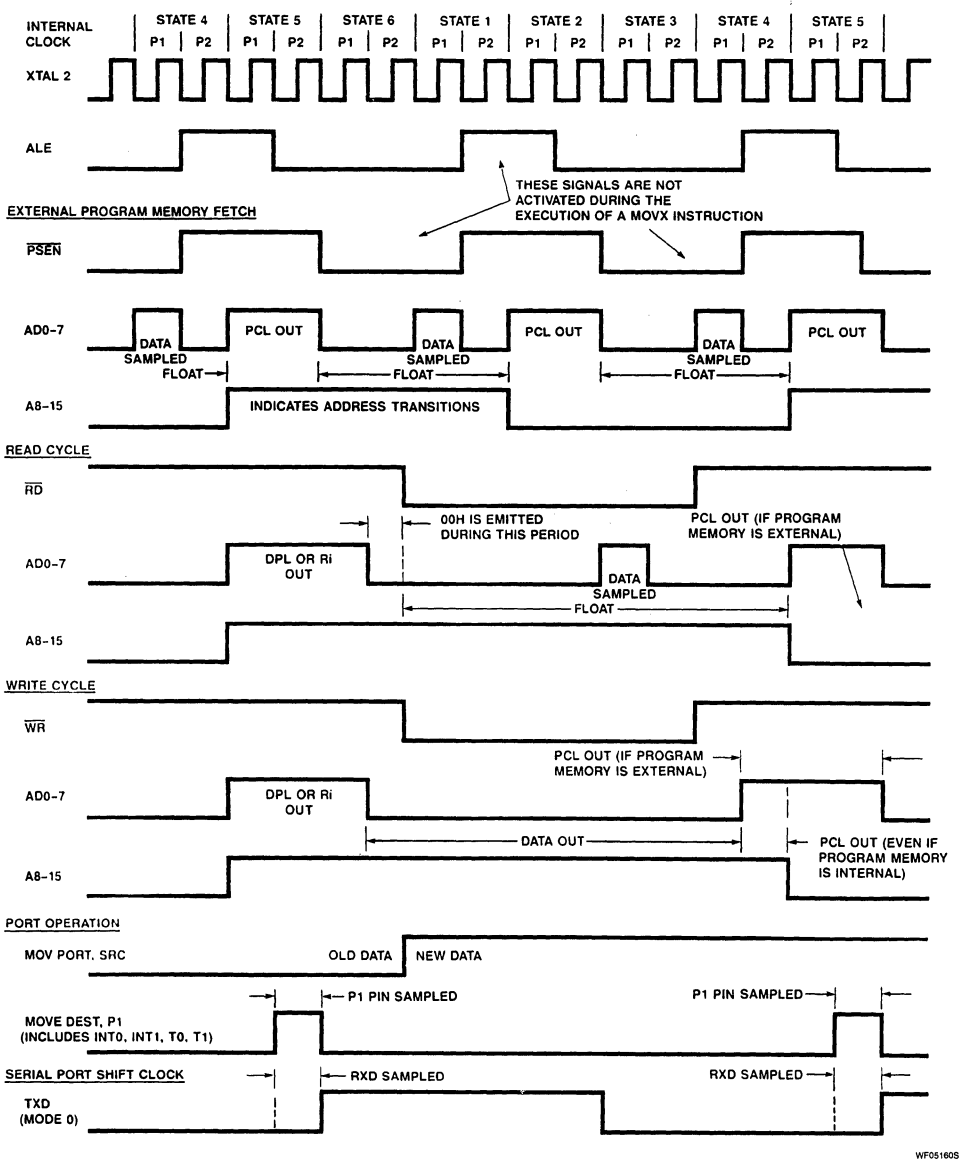


Figure 6. Timing Waveforms

NOTE:
All internal timing is referenced to the internal time states shown at the top of the page. This waveform represents the signal on the X2 input of the oscillator. This diagram represents when these signals are actually clocked within the chip. However, the time it takes a signal to propagate to the pins is in the range of 50-150ns. Propagation delays are dependent on many variables, such as temperature and pin loading. Even the different signals vary. Typically though, $\overline{RD}$ and $\overline{WR}$ have propagation delays of approximately 50ns and the other timing signals approximately 85ns. At room temperature, fully loaded, these differences in propagation delays between signals have been integrated into the timing specs.

Table 1. INSTRUCTION SET

MNEMONIC		DESCRIPTION	BYTE	CYCLES
Arithmetic Operations				
ADD	A,Rn	Add register to accumulator	1	1
ADD	A,direct	Add direct byte to accumulator	2	1
ADD	A,@Ri	Add indirect RAM to accumulator	1	1
ADD	A,R#data	Add immediate data to accumulator	2	1
ADDC	A,Rn	Add register to accumulator with carry	1	1
ADDC	A,direct	Add direct byte to A with carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with carry flag	1	1
ADDC	A,#data	Add immediate data to A with carry flag	2	1
SUBB	A,Rn	Subtract register from A with borrow	1	1
SUBB	A,direct	Subtract direct byte from A with borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A w/borrow	1	1
SUBB	A,#data	Subtract immed. data from A w/borrow	2	1
INC	A	Increment accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement Indirect RAM	1	1
INC	DPTR	Increment data pointer	1	2
MUL	AB	Mulitiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal adjust accumulator	1	1
Logical Operations				
ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive-OR register to accumulator	1	1
XRL	A,direct	Exclusive-OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive-OR indirect RAM to A	1	1
XRL	A,#data	Exclusive-OR immediate data to A	2	1
XRL	direct,A	Exclusive-OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear accumulator	1	1
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate A left through the carry flag	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate A right through carry flag	1	1
SWAP	A	Swap nibbles within the accumulator	1	1
Data Transfer				
MOV	A,Rn	Move register to accumulator	1	1
MOV	A,direct	Move direct byte to accumulator	2	1
MOV	A,@Ri	Move indirect RAM to accumulator	1	1
MOV	A,#data	Move immediate data to accumulator	2	1
MOV	Rn,A	Move accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct, #data	Move immediate data to direct byte	3	2

Table 1. INSTRUCTION SET (Continued)

MNEMONIC		DESCRIPTION	BYTE	CYCLES
Data Transfer (Continued)				
MOV	@Ri,A	Move accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data16	Load data pointer with a 16-bit constant	3	2
MOVC	A,@A + DPTR	Move code byte relative to DPTR to A	1	2
MOVC	A,@A + PC	Move code byte relative to PC to A	1	2
MOVX	A,@Ri	Move external RAM (8-bit addr) to A	1	2
MOVX	A,@DPTR	Move external RAM (16-bit addr) to A	1	2
MOVX	@Ri,A	Move A to external RAM (8-bit addr)	1	2
MOVX	@DPTR,A	Move A to external RAM (16-bit addr)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with accumulator	1	1
XCH	A,direct	Exchange direct byte with accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with A	1	1
XCHD	A,@Ri	Exchange low-order digit ind. RAM w/A	1	1
Boolean Variable Manipulation				
CLR	C	Clear carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set carry flag	1	1
SETB	bit	Set direct bit	2	1
CPL	C	Complement carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to carry flag	2	2
ANL	C,bit	AND complement of direct bit to carry	2	2
ORL	C,bit	OR direct bit to carry flag	2	2
ORL	C,bit	OR complement of direct bit to carry	2	2
MOV	C,bit	Move direct bit to carry flag	2	1
MOV	bit,C	Move carry flag to direct bit	2	2
Program and Machine Control				
ACALL	addr11	Absolute subroutine call	2	2
LCALL	addr16	Long subroutine call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute jump	2	2
LJMP	addr16	Long jump	3	2
SJMP	rel	Short jump (relative addr)	2	2
JMP	@A + DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if accumulator is zero	2	2
JNZ	rel	Jump if accumulator is not zero	2	2
JC	rel	Jump if carry flag is set	2	2
JNC	rel	Jump if no carry flag	2	2
JB	bit,rel	Jump if direct bit set	3	2
JNB	bit,rel	Jump if direct bit not set	3	2
JBC	bit,rel	Jump if direct bit is set & clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & jump if not equal	3	2
CJNE	A,#data, rel	Comp. immed. to A & jump if not equal	3	2
CJNE	Rn,#data, rel	Comp. immed. to reg. & jump if not equal	3	2
CJNE	@Ri,#data, rel	Comp. immed. to ind. & jump if not equal	3	2
DJNZ	Rn,rel	Decrement register & jump if not zero	2	2
DJNZ	direct,rel	Decrement direct & jump if not zero	3	2
NOP		No operation	1	1

Notes on data addressing modes:

Rn	–Working register R0-R7
direct	–128 internal RAM locations, any I/O port, control or status register
@Ri	–Indirect Internal RAM location addressed by register R0 or R1
#data	–8-bit constant included in instruction
#data16	–16-bit constant included as bytes 2 & 3 of instruction
bit	–128 software flags, any I/O pin, control or status bit

Notes on program addressing modes:

addr16	–Destination address for LCALL & LJMP may be anywhere within the 64-kilobyte program memory address space.
addr11	–Destination address for ACALL & AJMP will be within the same 2-kilobyte page of program memory as the first byte of the following instruction.
rel	–SJMP and all conditional jumps include an 8-bit offset byte. Range is +127 – 128 bytes relative to first byte of the following instruction.

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A,#data
25	2	ADD	A,data addr
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr,code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A,#data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Continued)

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr,#data
44	2	ORL	A,#data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr,#data
54	2	ANL	A,#data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr,A
63	3	XRL	data addr,#data
64	2	XRL	A,#data
65	2	XRL	A,data addr
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data
79	2	MOV	R1,#data
7A	2	MOV	R2,#data
7B	2	MOV	R3,#data
7C	2	MOV	R4,#data
7D	2	MOV	R5,#data
7E	2	MOV	R6,#data

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Continued)

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
7F	2	MOV	R7,#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A + PC
84	1	DIV	AB
85	3	MOV	data addr,data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR,#data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A + DPTR
94	2	SUBB	A,#data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0
97	1	SUBB	A,@R1
98	1	SUBB	A,R0
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,/bit addr
A1	2	AJMP	code addr
A2	2	MOV	C,bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,data addr
A7	2	MOV	@R1,data addr
A8	2	MOV	R0,data addr
A9	2	MOV	R1,data addr
AA	2	MOV	R2,data addr
AB	2	MOV	R3,data addr
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,/bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A,#data,code addr
B5	3	CJNE	A,data addr,code addr
B6	3	CJNE	@R0,#data,code addr
B7	3	CJNE	@R1,#data,code addr
B8	3	CJNE	R0,#data,code addr
B9	3	CJNE	R1,#data,code addr
BA	3	CJNE	R2,#data,code addr
BB	3	CJNE	R3,#data,code addr
BC	3	CJNE	R4,#data,code addr
BD	3	CJNE	R5,#data,code addr
BE	3	CJNE	R6,#data,code addr
BF	3	CJNE	R7,#data,code addr

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Continued)

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr,code addr
D6	1	XCHD	A,@R0
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,code addr
D9	2	DJNZ	R1,code addr
DA	2	DJNZ	R2,code addr
DB	2	DJNZ	R3,code addr
DC	2	DJNZ	R4,code addr
DD	2	DJNZ	R5,code addr
DE	2	DJNZ	R6,code addr
DF	2	DJNZ	R7,code addr
E0	1	MOVX	A,@DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A,@R0
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,data addr
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	data addr,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A

Signetics

SCN8032AH, SCN8052AH
Single-chip 8-Bit Microcontroller

Preliminary Specification

Microprocessor Products

DESCRIPTION

The Signetics SCN8032/SCN8052 8-bit Microcontroller is a high-performance single-chip computer fabricated with Signetics' highly-reliable +5 volt, depletion-load, N-Channel, silicon-gate MOS technology. It provides the hardware features, architectural enhancements and new instructions that are necessary to make it a powerful and cost effective controller for applications.

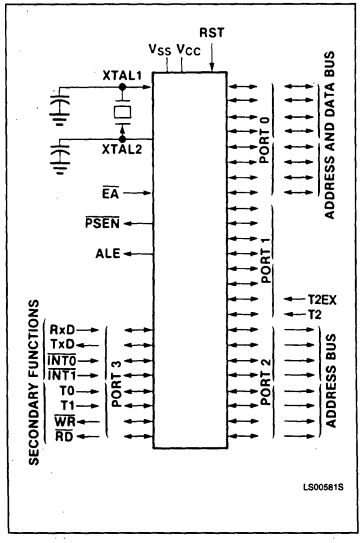
The SCN8032AH contains 256 bytes of read/write data memory; 32 I/O lines configured as four 8-bit ports: three 16-bit timer/counters; a six-source, two-priority level, nested interrupt structure; a programmable serial I/O port; and an on-chip oscillator with clock circuitry. The SCN8052AH has all of these features plus 8K bytes of nonvolatile read-only program memory. Both microcontrollers have memory expansion capabilities of up to 64K bytes of data storage and 64K bytes of program memory that may be realized with standard TTL compatible memories.

The SCN8032AH/SCN8052AH microcontroller is efficient at both computational and control-oriented tasks. This results from its extensive BCD/binary arithmetic and bit-handling facilities. Efficient use of program memory is also achieved by using the familiar compact instruction set of the 8031/8051. Forty-four percent of the instructions are one-byte, 41% two-byte and 15% three-byte. The majority of the instructions execute in just 1.0μs at 12MHz operation. The longest instructions, multiply and divide, require only 4μs at 12MHz.

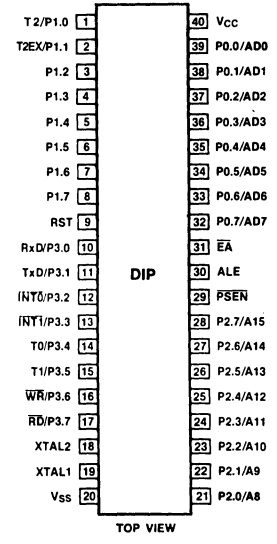
FEATURES

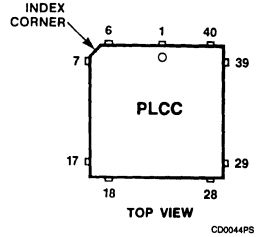
- **SCN8032AH – control-oriented CPU with RAM and I/O**
- **SCN8052AH – an SCN8032AH with factory mask-programmable ROM**
- **8K x 8 ROM (SCN8052AH only)**
- **256 x 8 RAM**
- **32 I/O lines (four 8-bit ports)**
- **Three 16-bit timer/counters**
- **Programmable full-duplex serial channel**
- **Variable transmit/receive baud rate capability**
- **Timer 2 capture capability**
- **128K accessible external memory**
- **Boolean processor**
- **128 user bit-addressable locations**
- **Upward compatible with SCN8031AH/SCN8051AH**

LOGIC SYMBOL



PIN CONFIGURATION





Pin	Function	Pin	Function
1	NC	23	NC
2	T2/P1.0	24	A8/P2.0
3	T2EX/P1.1	25	A9/P2.1
4	P1.2	26	A10/P2.2
5	P1.3	27	A11/P2.3
6	P1.4	28	A12/P2.4
7	P1.5	29	A13/P2.5
8	P1.6	30	A14/P2.6
9	P1.7	31	A15/P2.7
10	RST	32	$\overline{PSEN}$
11	RXD/P3.0	33	ALE
12	NC	34	NC
13	TXD/P3.1	35	$\overline{EA}$
14	$\overline{INT0}$/P3.2	36	AD7/P0.7
15	$\overline{INT1}$/P3.3	37	AD6/P0.6
16	T0/P3.4	38	AD5/P0.5
17	T1/P3.5	39	AD4/P0.4
18	$\overline{WR}$/P3.6	40	AD3/P0.3
19	$\overline{RD}$/P3.7	41	AD2/P0.2
20	XTAL2	42	AD1/P0.1
21	XTAL1	43	AD0/P0.0
22	VSS	44	VCC

Single-chip 8-Bit Microcontroller — SCN8032AH, SCN8052AH

ORDERING CODE

SCN80□□ H □ □ □ 40 (CPxxxx)

ROM/RAM (bytes)
32 = EXT/256
52 = 8K/256

POWER CONSUMPTION
H = Reduced Power AH

OPERATING TEMPERATURE RANGE
A = −40°C to +85°C
C = 0°C to +70°C

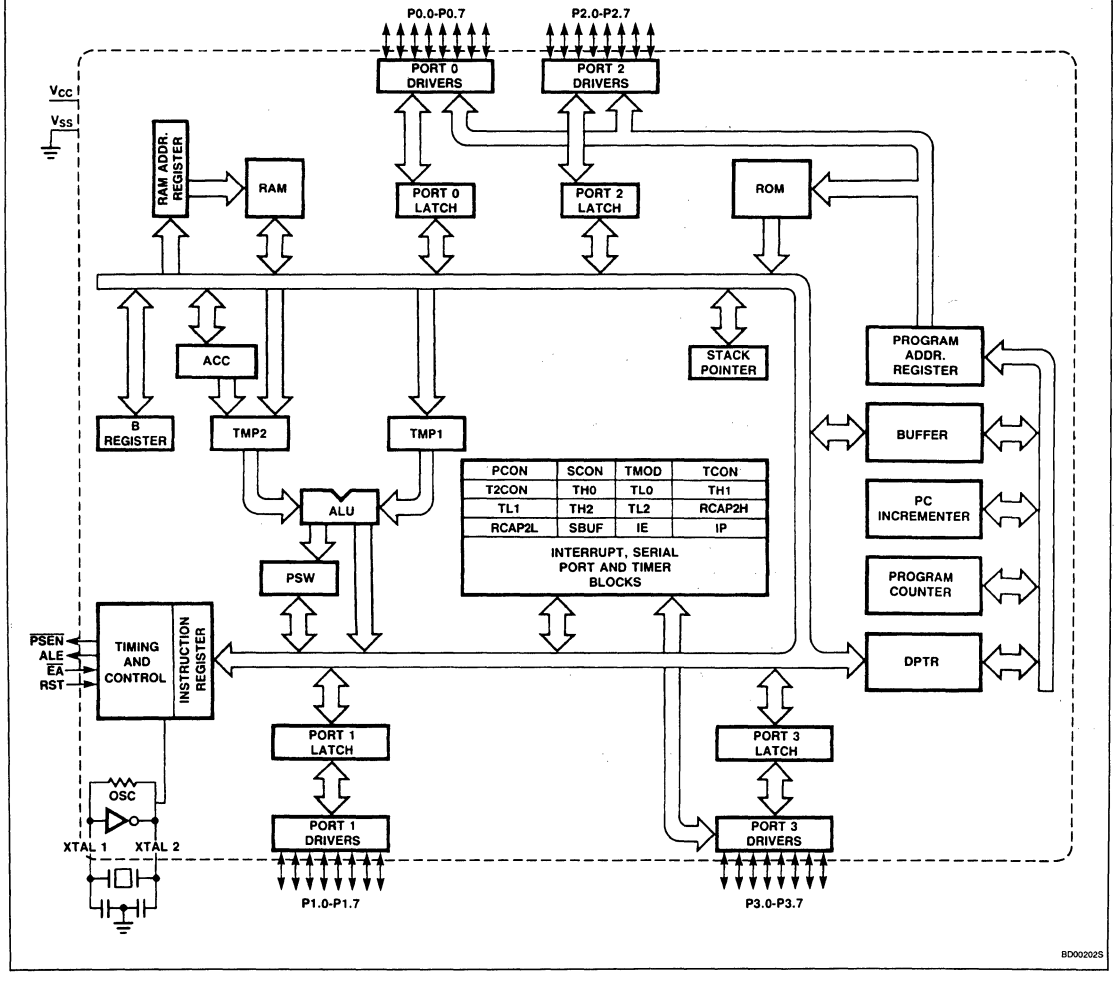
SPEED
C = 12MHz clock

CUSTOM ROM PATTERN NUMBER
Applies to masked ROM versions only. Number will be assigned by Signetics. Contact Signetics sales office for ROM pattern submission requirements.

40-PIN DIP
44-PIN PLCC

PACKAGE
N = Plastic DIP
I = Ceramic DIP
A = Plastic LCC

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. LCC	TYPE	NAME AND FUNCTION
V_{SS}	20	22	I	Circuit ground potential.
V_{CC}	40	44	I	+5V supply voltage during operation and program verification.
P0.0 – P0.7	39 – 32	43 – 36	I/O	**Port 0:** Port 0 is an 8-bit open-drain, bidirectional I/O port. It is also the multiplexed low-order address and data bus when using external memory. It is used for data output during program verification. Port 0 can sink (and in bus operations can source) eight LS TTL loads.
P1.0 – P1.7	1 – 8	2 – 9	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port. Pins P1.0 and P1.1 also correspond to the special functions T2, timer 2 counter trigger input, and T2EX, external input to timer 2. The output latch on these two special function pins must be programmed to a one for that function to operate. Port 1 is also used for the low-order address byte during program verification. Port 1 can sink/source four LS TTL loads.
	1	2	I	**T2 (P1.0):** Timer counter 2 trigger input.
	2	3	I	**T2EX (P1.1):** Timer counter 2 external count input.
P2.0 – P2.7	21 – 28	24 – 31	I/O	**Port 2:** Port 2 is an 8-bit quasi-bidirectional I/O port. It also emits the high-order address byte when accessing external memory. It is used for the high-order address and the control signals during program verification. Port 2 can sink/source four LS TTL loads.
P3.0 – P3.7	10 – 17	11, 13 – 19	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port. The output latch corresponding to a secondary function must be programmed to a one for that function to operate. Port 3 can sink/source four LS TTL loads. Each of the P3 pins also correspond to the special functions listed below:
	10	11	I	**RXD (P3.0):** Serial input port
	11	13	O	**TXD (P3.1):** Serial output port
	12	14	I	**INT0 (P3.2):** External interrupt
	13	15	I	**INT1 (P3.4):** External interrupt
	14	16	I	**T0 (P3.4):** Timer 0 external input
	15	17	I	**T1 (P3.5):** Timer 1 external input
	16	18	O	**WR (P3.6):** External data memory write strobe
	17	19	I	**RD (P3.7):** External data memory read strobe
RST	9	10	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running resets the device. A small external pulldown resistor (≈8.2KΩ) from RST to V_{SS} permits power-on reset when a capacitor (≈10μF) is also connected from this pin to V_{CC}.
ALE	30	33	O	**Address Latch Enable:** Output for latching the address into external memory during normal operation. It is activated every six oscillator periods except during an external data memory access.
$\overline{PSEN}$	29	32	O	**Program Store Enable:** Output is a control signal that enables the external program memory to the bus during external fetch operations. It is activated every six oscillator periods, except during external data memory accesses. Remains high during internal program execution.
$\overline{EA}$	31	35	I	**Instruction Execution Control:** When held at a high TTL level, the SCN8052AH executes instructions from the internal ROM when the PC is less than 8192. When held at a TTL low level, the SCN8032AH/SCN8052AH fetches all instructions from external program memory.
XTAL1	19	21	I	**Crystal 1:** Input to the inverting amplifier that forms the oscillator.
XTAL2	18	20	O	**Crystal 2:** Output of the inverting amplifier that forms the oscillator and input to the internal clock generator. Receives the external oscillator signal when an external oscillator is used.

ABSOLUTE MAXIMUM RATINGS

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +7.0	V
Power dissipation	2.0	W

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 4.5$ to $5.5V$, $V_{SS} = 0V$[4,5]

	PARAMETER	TEST CONDITIONS	LIMITS Min	LIMITS Max	UNIT
V_{IL}	Input low voltage		−0.5	0.8	V
V_{IH}	Input high voltage (except RST and XTAL2)		2.0	$V_{CC}+0.5$	V
V_{IH1}	Input high voltage to RST for reset, XTAL2	XTAL1 to V_{SS}	2.5	$V_{CC}+0.5$	V
V_{OL}	Output low voltage ports 1, 2, 3[6]	I_{OL} = 1.6mA		0.45	V
V_{OL1}	Output low voltage port 0, ALE, $\overline{PSEN}$[7]	I_{OL} = 3.2mA		0.45	V
V_{OH}	Output high voltage ports 1, 2, 3	$I_{OH} = -80\mu A$	2.4		V
V_{OH1}	Output high voltage port 0, ALE, $\overline{PSEN}$	$I_{OH} = -400\mu A$	2.4		V
I_{IL}	Logical 0 input current ports 1, 2, 3	V_{IN} = 0.45V		−800	μA
I_{IH1}	Input high current to RST for reset	$V_{IN} = V_{CC} - 1.5V$		500	μA
I_{LI}	Input leakage current to port 0, $\overline{EA}$	$0.45 < V_{IN} < V_{CC}$		±10	μA
I_{IL2}	Logical 0 input current for XTAL2	XTAL1 at V_{SS}, V_{IN} = 0.45V		−2.5	mA
I_{CC}	Power supply current[9]	$\overline{EA} = V_{CC}$		125	mA
C_{IO}	Capacitance of I/O buffer	f_C = 1MHz, $T_A = 25°C$		10	pF
$T_A = -40°C$ to $+85°C$					
V_{IH}	Input high voltage (except RST and XTAL2)		2.2	$V_{CC}+0.5$	V
V_{IH1}	Input high voltage to RST for reset, XTAL2	XTAL1 to V_{SS}	2.7		V
I_{CC}	Power supply current[9]	$\overline{EA} = V_{CC}$		175	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying voltages greater than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground. For testing, all input signals swing between 0.45V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and at output voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical supply voltages and typ ical processing parameters.
7. V_{OL} is degraded when the SCN8032AH/SCN8052AH rapidly disch arges external capacitance. This AC noise is most pronounced during emission of address data. When using external memory, locate the latch or buffer as close to the SCN8032AH/SCN8052AH as possible.

Datum	Emitting Ports	Degraded I/O Lines	V_{OL}(Peak Max)
Address	P2, P0	P1, P3	0.8V
Write data	P0	P1, P3, ALE	0.8V

8. C_L = 100pF for port 0, ALE and $\overline{PSEN}$ outputs; C_L = 80pF for all other ports.
9. All outputs disconnected.

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$[4,5,7]

PARAMETER		12MHz CLOCK		VARIABLE CLOCK $1/t_{CLCL}$ = 3.5MHz to 12MHz		UNIT
		Min	Max	Min	Max	
Program memory char (fig. 1)						
t_{LHLL}	ALE pulse width	127		$2t_{CLCL} -40$		ns
t_{AVLL}	Address set-up to ALE	43		$t_{CLCL} -40$		ns
t_{LLAX}	Address hold after ALE	48		$t_{CLCL} -35$		ns
t_{LLIV}	ALE to valid instr in		233		$4t_{CLCL} -100$	ns
t_{LLPL}	ALE to $\overline{PSEN}$	58		$t_{CLCL} -25$		ns
t_{PLPH}	$\overline{PSEN}$ pulse width	215		$3t_{CLCL} -35$		ns
t_{PLIV}	$\overline{PSEN}$ to valid instr in		125		$3t_{CLCL} -125$	ns
t_{PXIX}	Input instr hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	Input instr float after $\overline{PSEN}$		63		$t_{CLCL} -20$	ns
t_{PXAV}	Address valid after $\overline{PSEN}$	75		$t_{CLCL} -8$		ns
t_{AVIV}	Address to valid instr in		302		$5t_{CLCL} -115$	ns
t_{PLAZ}	$\overline{PSEN}$ to address float		0		0	ns
External data memory char (fig. 2 and 3)						
t_{RLRH}	$\overline{RD}$ pulse width	400		$6t_{CLCL} -100$		ns
t_{WLWH}	$\overline{WR}$ pulse width	400		$6t_{CLCL} -100$		ns
t_{LLAX}	Address hold after ALE	48		$t_{CLCL} -35$		ns
t_{RLDV}	$\overline{RD}$ to valid data in		250		$5t_{CLCL} -165$	ns
t_{RHDX}	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	Data float after $\overline{RD}$		97		$2t_{CLCL} -70$	ns
t_{LLDV}	ALE to valid data in		517		$8t_{CLCL} -150$	ns
t_{AVDV}	Address to valid data in		585		$9t_{CLCL} -165$	ns
t_{LLWL}	ALE to $\overline{WR}$ or $\overline{RD}$	200	300	$3t_{CLCL} -50$	$3t_{CLCL} +50$	ns
t_{AVWL}	Address to $\overline{WR}$ or $\overline{RD}$	203		$4t_{CLCL} -130$		ns
t_{WHLH}	$\overline{WR}$ or $\overline{RD}$ high to ALE high	43	123	$t_{CLCL} -40$	$t_{CLCL} +40$	ns
t_{DVWX}	Data valid to $\overline{WR}$ transition	23		$t_{CLCL} -60$		ns
t_{QVWH}	Data set-up before $\overline{WR}$	433		$7t_{CLCL} -150$		ns
t_{WHQX}	Data hold after $\overline{WR}$	33		$t_{CLCL} -50$		ns
t_{RLAZ}	Address float after $\overline{RD}$		0		0	ns
External clock (fig. 4)						
t_{CLCL}	Oscillator period			83.3	286	ns
t_{CHCX}	High time			20		ns
t_{CLCX}	Low time			20		ns
t_{CLCH}	Rise time				20	ns
t_{CHCL}	Fall time				20	ns

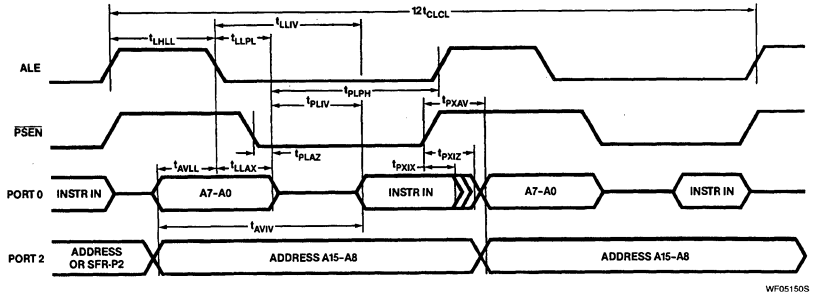


Figure 1. Program Memory Read Cycle

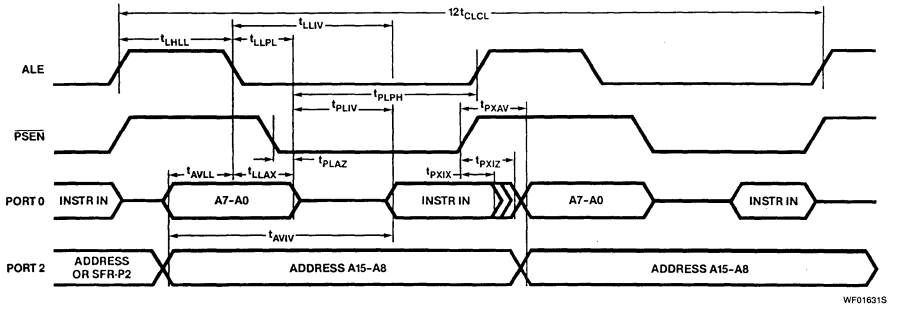


Figure 2. Data Memory Read Cycle

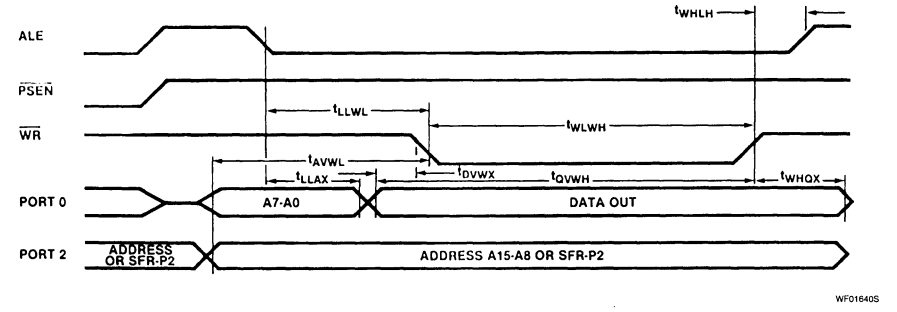


Figure 3. Data Memory Write Cycle

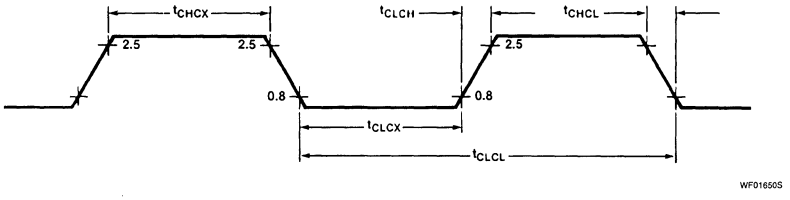


Figure 4. External Clock Drive XTAL2

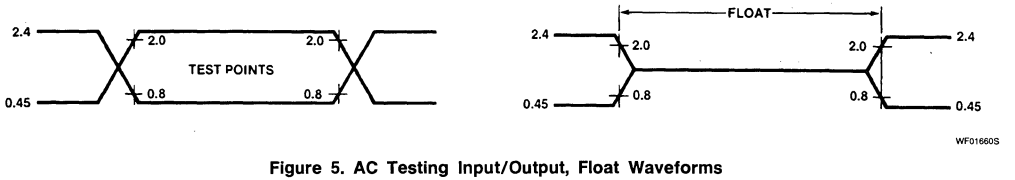


Figure 5. AC Testing Input/Output, Float Waveforms

NOTES:
1. AC testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0".
2. Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".
3. For timing purposes, the float state is defined as the point at which a P0 pin sinks 2.4mA or sources 400μA at the voltage test levels.

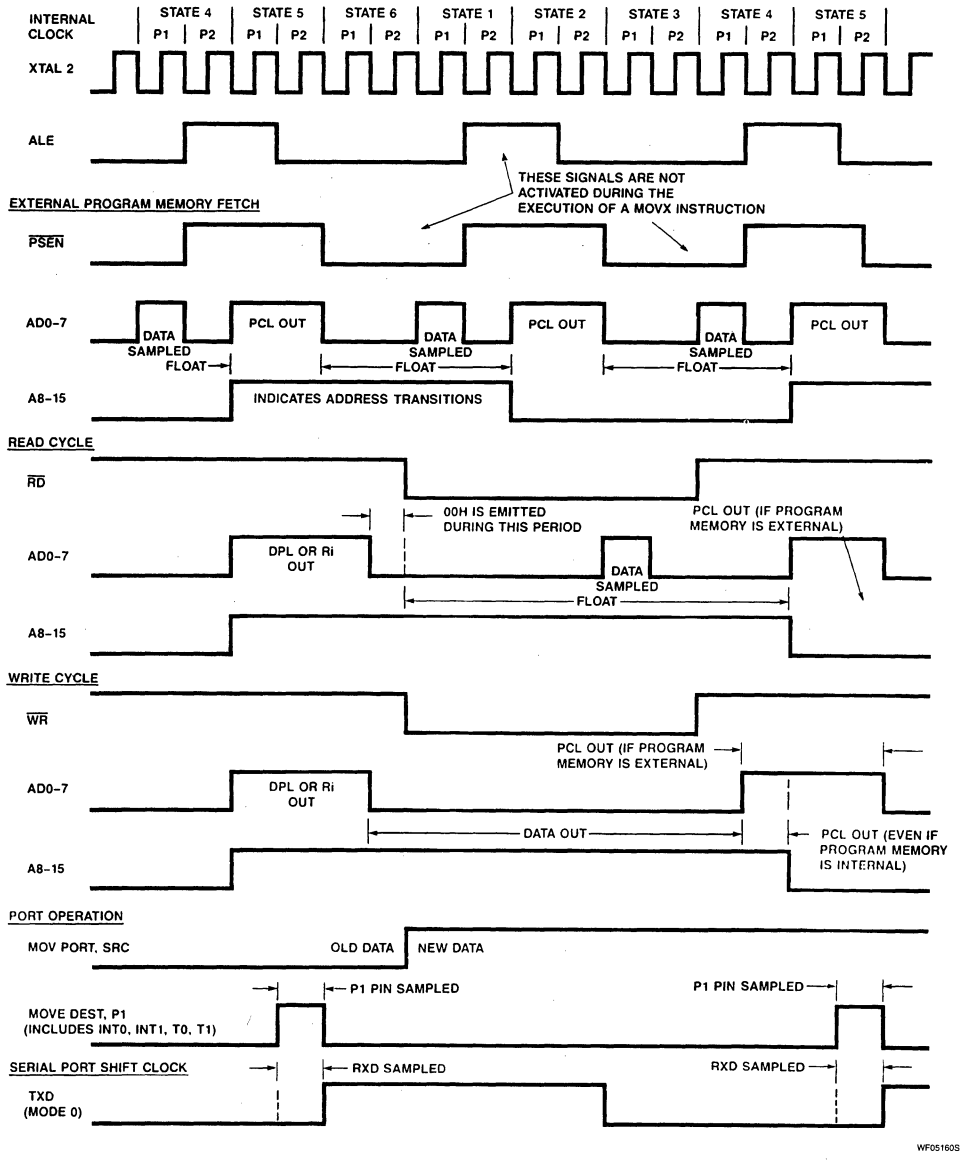


Figure 6. Timing Waveforms

NOTE:
All internal timing is referenced to the internal time states shown at the top of the page. This waveform represents the signal on the X2 input of the oscillator. This diagram represents when these signals are actually clocked within the chip. However, the time it takes a signal to propagate to the pins is in the range of 50-150ns. Propagation delays are dependent on many variables, such as temperature and pin loading. Even the different signals vary. Typically though, $\overline{RD}$ and $\overline{WR}$ have propagation delays of approximately 50ns and the other timing signals approximately 85ns. At room temperature, fully loaded, these differences in propagation delays between signals have been integrated into the timing specs.

Table 1. INSTRUCTION SET

MNEMONIC		DESCRIPTION	BYTE	CYCLES
Arithmetic Operations				
ADD	A,Rn	Add register to accumulator	1	1
ADD	A,direct	Add direct byte to accumulator	2	1
ADD	A,@Ri	Add indirect RAM to accumulator	1	1
ADD	A,R#data	Add immediate data to accumulator	2	1
ADDC	A,Rn	Add register to accumulator with carry	1	1
ADDC	A,direct	Add direct byte to A with carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with carry flag	1	1
ADDC	A,#data	Add immediate data to A with carry flag	2	1
SUBB	A,Rn	Subtract register from A with borrow	1	1
SUBB	A,direct	Subtract direct byte from A with borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A w/borrow	1	1
SUBB	A,#data	Subtract immed. data from A w/borrow	2	1
INC	A	Increment accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement Indirect RAM	1	1
INC	DPTR	Increment data pointer	1	2
MUL	AB	Mulitiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal adjust accumulator	1	1
Logical Operations				
ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive-OR register to accumulator	1	1
XRL	A,direct	Exclusive-OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive-OR indirect RAM to A	1	1
XRL	A,#data	Exclusive-OR immediate data to A	2	1
XRL	direct,A	Exclusive-OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear accumulator	1	1
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate A left through the carry flag	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate A right through carry flag	1	1
SWAP	A	Swap nibbles within the accumulator	1	1
Data Transfer				
MOV	A,Rn	Move register to accumulator	1	1
MOV	A,direct	Move direct byte to accumulator	2	1
MOV	A,@Ri	Move indirect RAM to accumulator	1	1
MOV	A,#data	Move immediate data to accumulator	2	1
MOV	Rn,A	Move accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct, #data	Move immediate data to direct byte	3	2

Table 1. INSTRUCTION SET (Continued)

MNEMONIC		DESCRIPTION	BYTE	CYCLES
Data Transfer (Continued)				
MOV	@Ri,A	Move accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data16	Load data pointer with a 16-bit constant	3	2
MOVC	A,@A + DPTR	Move code byte relative to DPTR to A	1	2
MOVC	A,@A + PC	Move code byte relative to PC to A	1	2
MOVX	A,@Ri	Move external RAM (8-bit addr) to A	1	2
MOVX	A,@DPTR	Move external RAM (16-bit addr) to A	1	2
MOVX	@Ri,A	Move A to external RAM (8-bit addr)	1	2
MOVX	@DPTR,A	Move A to external RAM (16-bit addr)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with accumulator	1	1
XCH	A,direct	Exchange direct byte with accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with A	1	1
XCHD	A,@Ri	Exchange low-order digit ind. RAM w/A	1	1
Boolean Variable Manipulation				
CLR	C	Clear carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set carry flag	1	1
SETB	bit	Set direct bit	2	1
CPL	C	Complement carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to carry flag	2	2
ANL	C,bit	AND complement of direct bit to carry	2	2
ORL	C,bit	OR direct bit to carry flag	2	2
ORL	C,bit	OR complement of direct bit to carry	2	2
MOV	C,bit	Move direct bit to carry flag	2	1
MOV	bit,C	Move carry flag to direct bit	2	2
Program and Machine Control				
ACALL	addr11	Absolute subroutine call	2	2
LCALL	addr16	Long subroutine call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute jump	2	2
LJMP	addr16	Long jump	3	2
SJMP	rel	Short jump (relative addr)	2	2
JMP	@A + DPTR	Jump indirect relative to tho DPTR	1	2
JZ	rel	Jump if accumulator is zero	2	2
JNZ	rel	Jump if accumulator is not zero	2	2
JC	rel	Jump if carry flag is set	2	2
JNC	rel	Jump if no carry flag	2	2
JB	bit,rel	Jump if direct bit set	3	2
JNB	bit,rel	Jump if direct bit not set	3	2
JBC	bit,rel	Jump if direct bit is set & clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & jump if not equal	3	2
CJNE	A,#data, rel	Comp. immed. to A & jump if not equal	3	2
CJNE	Rn,#data, rel	Comp. immed. to reg. & jump if not equal	3	2
CJNE	@Ri,#data, rel	Comp. immed. to ind. & jump if not equal	3	2
DJNZ	Rn,rel	Decrement register & jump if not zero	2	2
DJNZ	direct,rel	Decrement direct & jump if not zero	3	2
NOP		No operation	1	1

Notes on data addressing modes:

Rn	–Working register R0-R7
direct	–128 internal RAM locations, any I/O port, control or status register
@Ri	–Indirect Internal RAM location addressed by register R0 or R1
#data	–8-bit constant included in instruction
#data16	–16-bit constant included as bytes 2 & 3 of instruction
bit	–128 software flags, any I/O pin, control or status bit

Notes on program addressing modes:

addr16	–Destination address for LCALL & LJMP may be anywhere within the 64-kilobyte program memory address space.
addr11	–Destination address for ACALL & AJMP will be within the same 2-kilobyte page of program memory as the first byte of the following instruction.
rel	–SJMP and all conditional jumps include an 8-bit offset byte. Range is +127 – 128 bytes relative to first byte of the following instruction.

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A, #data
25	2	ADD	A,data addr
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr,code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A, #data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Continued)

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr,#data
44	2	ORL	A,#data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr,#data
54	2	ANL	A,#data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr,A
63	3	XRL	data addr,#data
64	2	XRL	A,#data
65	2	XRL	A,data addr
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data
79	2	MOV	R1,#data
7A	2	MOV	R2,#data
7B	2	MOV	R3,#data
7C	2	MOV	R4,#data
7D	2	MOV	R5,#data
7E	2	MOV	R6,#data

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Continued)

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
7F	2	MOV	R7,#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A + PC
84	1	DIV	AB
85	3	MOV	data addr,data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR,#data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A + DPTR
94	2	SUBB	A,#data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0
97	1	SUBB	A,@R1
98	1	SUBB	A,R0
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,/bit addr
A1	2	AJMP	code addr
A2	2	MOV	C,bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,data addr
A7	2	MOV	@R1,data addr
A8	2	MOV	R0,data addr
A9	2	MOV	R1,data addr
AA	2	MOV	R2,data addr
AB	2	MOV	R3,data addr
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,/bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A,#data,code addr
B5	3	CJNE	A,data addr,code addr
B6	3	CJNE	@R0,#data,code addr
B7	3	CJNE	@R1,#data,code addr
B8	3	CJNE	R0,#data,code addr
B9	3	CJNE	R1,#data,code addr
BA	3	CJNE	R2,#data,code addr
BB	3	CJNE	R3,#data,code addr
BC	3	CJNE	R4,#data,code addr
BD	3	CJNE	R5,#data,code addr
BE	3	CJNE	R6,#data,code addr
BF	3	CJNE	R7,#data,code addr

Table 2. INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Continued)

HEX CODE	NUMBER OF BYTES	MNEMONIC	OPERANDS
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr,code addr
D6	1	XCHD	A,@R0
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,code addr
D9	2	DJNZ	R1,code addr
DA	2	DJNZ	R2,code addr
DB	2	DJNZ	R3,code addr
DC	2	DJNZ	R4,code addr
DD	2	DJNZ	R5,code addr
DE	2	DJNZ	R6,code addr
DF	2	DJNZ	R7,code addr
E0	1	MOVX	A,@DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A,@R0
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,data addr
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	data addr,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A

SCN8400 Series
Single-Chip 8-Bit Microcontroller

Product Specification

Microprocessor Products

DESCRIPTION

The SCN84XX family of microcontrollers is fabricated in NMOS. The family consists of five devices. Each version has 20 quasi-bidirectional I/O port lines, one serial I/O line, one single-level vectored interrupt, an 8-bit timer ovent counter and on-board clock oscillator and clock circuits. Two 20-pin versions, the SCN8422 and SCN8442, are also available.

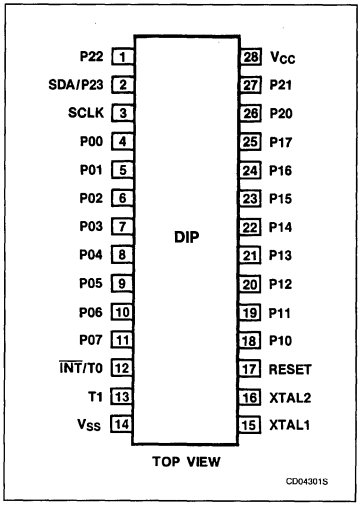
This microcontroller family is designed to be an efficient controller as well as an arithmetic processor. The instruction set is based on that of the SCN8048. The microcontrollers have extensive bit handling abilities and facilities for both binary and BCD arithmetic.

For detailed information refer to Philips' "Users Manual Single-Chip Microcomputer."

FEATURES

- **I^2C serial I/O that can be used in single or multimaster systems (serial I/O data and clock via P23 and SCLK lines, respectively)**
- **8-bit CPU, ROM, RAM and I/O in a single 28-lead DIP package**
- **2K, 4K or 6K ROM bytes plus a ROM-less version**
- **64 or 128 RAM bytes**
- **20 quasi-bidirectional I/O port lines**
- **Two testable inputs: one of which can be used to detect zero cross-over; the other is also the external interrupt input**
- **Single level vectored interrupts: external, timer/event counter, serial I/O**
- **8-bit programmable timer/event counter**
- **Internal oscillator, generated with inductor, crystal, ceramic resonator or external source**
- **Over 80 instructions (based on SCN8048) all of 1 or 2 cycles**
- **Single 5V power supply (± 10%)**
- **8-bit 10mA LED-driver on port 1**

PIN CONFIGURATION



ORDERING CODE

SCN84 □□ A □□ □□□ (CPXXXX)

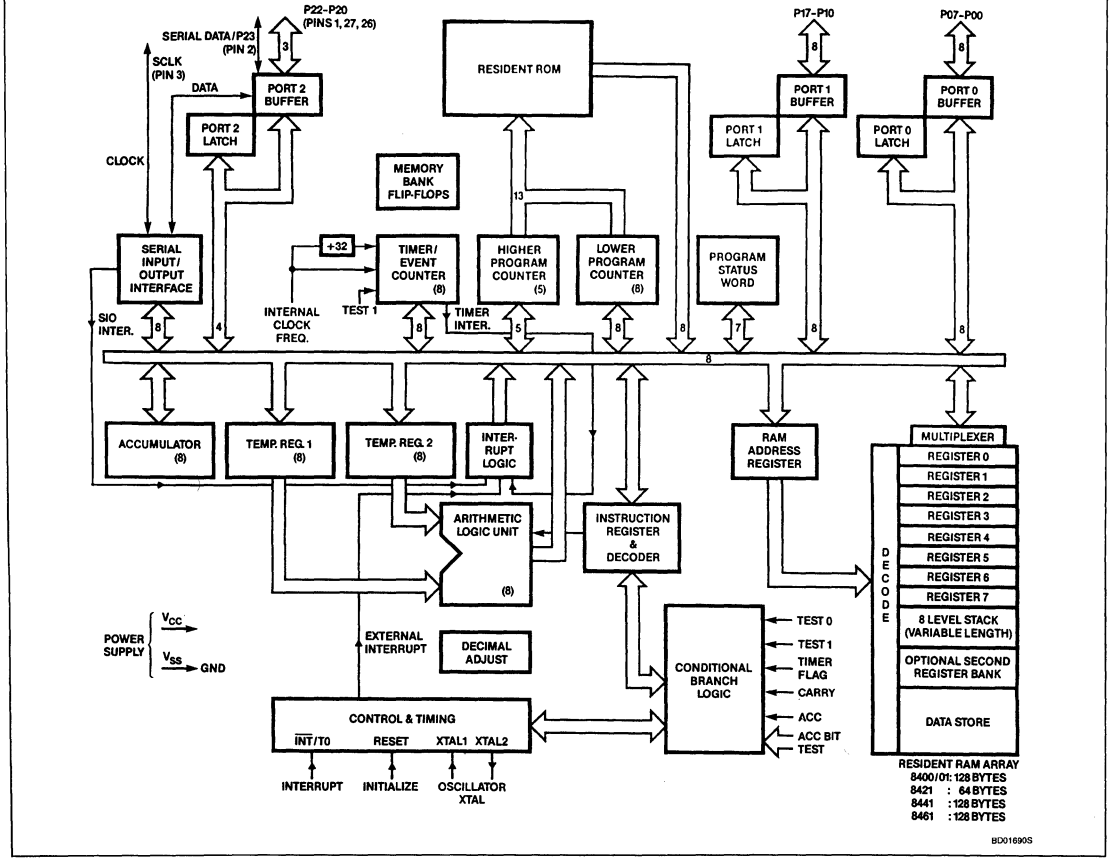
MEMORY ROM/RAM
00 = 0/128 x 8*
01 = 0/128 x 8*
21 = 2K x 8/64 x 8
41 = 4K x 8/128 x 8
61 = 6K x 8/128 x 8

CUSTOM ROM PATTERN AND I/O PORT NUMBER
Applied to masked ROM versions only. Numbers will be assigned by Signetics. Contact Signetics sales office for ROM pattern and I/O port option submission requirements.

PACKAGE
N28 = 28 Pin DIP
A28 = 28 PLCC

TEMPERATURE/FREQUENCY
C6 = 0°C to 70°C/6MHz
A6 = −40°C to 85°C/6MHz
E5 = −40°C to 110°C/5MHz

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{SS}	14	I	**Ground**: 0V reference.
V_{CC}	28	I	**Power Supply**: +5 volts.
P00 – P07	4 – 11	I/O	**Port 0**: 8-bit quasi-bidirectional I/O port.
P10 – P17	18 – 25	I/O	**Port 1**: 8-bit quasi-bidirectional I/O port.
P20 – P23	26, 27, 1, 2	I/O	**Port 2**: 4-bit quasi-bidirectional I/O port; P23 is the serial data I/O in serial I/O mode.
SCLK	3	I/O	**Serial Clock**: Bidirectional clock for serial I/O.
$\overline{INT}$/T0	12	I	**External Interrupt**: Input (sensitive to a negative-going edge min low > 7 clock pulses, min high > 4 clock pulses), testable using the JT0 or JNT0 instructions.
T1	13	I	**Test 1**: Input pin, testable using the JT1 or JNT1 instructions. It can be designated as event counter input using the STRT CNT instruction. It can also be used to detect zero cross-over of slowly moving AC inputs.
RESET	17	I	**Reset**: Input to initialize the processor (active high).
XTAL1	15	I	**Crystal 1**: Connection to timing component (crystal) that determines the frequency of the internal oscillator. It is also the input for an external clock source.
XTAL2	16	I	**Crystal 2**: Connection to other side of the timing component.

EPROM SOCKET CONFIGURATION FOR SCN8400 AND SCN8401

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{SS}	14, 22	I	**Ground**: 0V reference.
V_{CC}	1, 26 – 28	I	**Power Supply**: +5 volts.
A0 – A12	10 – 3, 25, 24, 23, 21, 2	O	**Address Outputs**
D0 – D7	11 – 13, 15 – 19	I	**Data Inputs**
$\overline{PSEN}$	20	O	**Program Store Enable**

FUNCTIONAL DESCRIPTION

Piggyback Version SCN8400/8401

The piggyback version is a special package that has standard pinning on the bottom. An EPROM can be mounted on top in an additional socket. The total package height is greater than the standard DIP package. Emulation of the 20-pin SCN8422/8442 can also be facilitated with the SCN8401.

Figure 1 shows the EPROM socket pinout on top of the SCN8400/8401 package. The socket accepts 2764 and 2732 EPROMs.

All 28-Pin Versions — Program and Data Memory

The program memory (ROM) is mask-programmed at the factory. Figure 3 shows the program memory map. Program memory is arranged in banks of 2K bytes, that are selected by SEL MB instructions.

The data memory (RAM) consists of 64 or 128 bytes (8-bit words). All locations are indirectly addressable using RAM pointer registers and up to 16 designated locations can be addressed directly. The memory also includes an 8-level program counter stack addressed by a 3-bit stack pointer. Figure 4 shows the data memory map.

On-Chip Peripheral Functions

In addition to the CPU and memories, an interrupt system, I/O facilities, and an 8-bit timer/event counter are integrated on-chip to assist the CPU in repetitious, complicated, or time-critical tasks. The I/O facilities include the I/O pins, parallel ports and a serial I/O port, consisting of a data line SDA shared with a parallel port line (P23), and a dedicated clock line SCLK.

I/O Facilities

The SCN84XX family has 23 I/O lines arranged as:

- Two parallel ports of 8 lines (P00 – P07, P10 – P17).
- A parallel port of 4 lines (P20 – P23).
- A serial I/O consisting of a data line shared with a parallel port line (P23) and a separate clock line SCLK. This supports the I^2C serial bus protocol and CBUS.
- An external interrupt and test input INT/T0, which when used as a test input can be tested by the conditional jump instructions JT0 or JNT0.
- A test input T1, which can altor program sequences when tested by conditional jump instructions JT1 or JNT1. T1 can also be used as an input to the timer/event counter or to detect zero cross-over of slowly moving AC signals.

All parallel port lines are available in three optional output configurations (except P23 – option 1 only. See figure 5.):

Option 1: Open drain output without pull-up transistor.

Option 2: Open drain output with pull-up transistor.

Option 3: Push-pull output with pull-up transistor. Standard output.

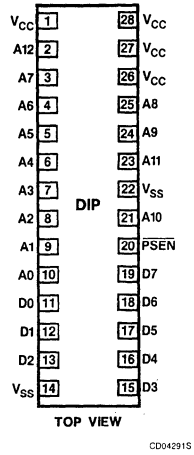


Figure 1. SCN8400/8401 Top Socket for EPROM

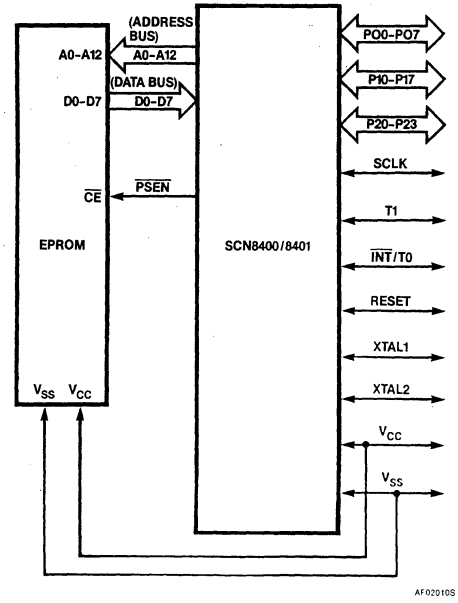


Figure 2. EPROM Connection to SCN8400/8401 Piggyback Package

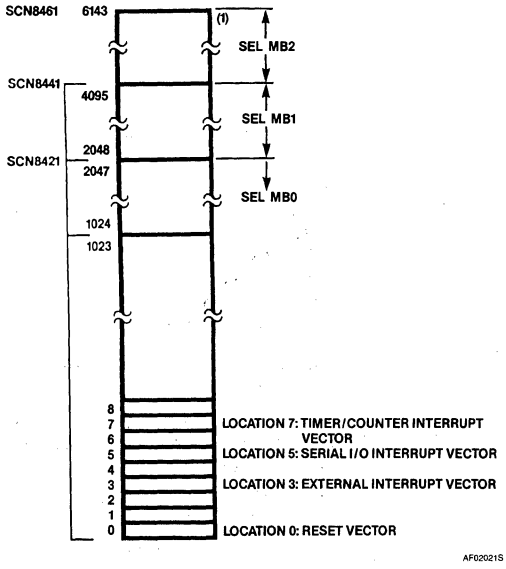


Figure 3. Program Memory Map

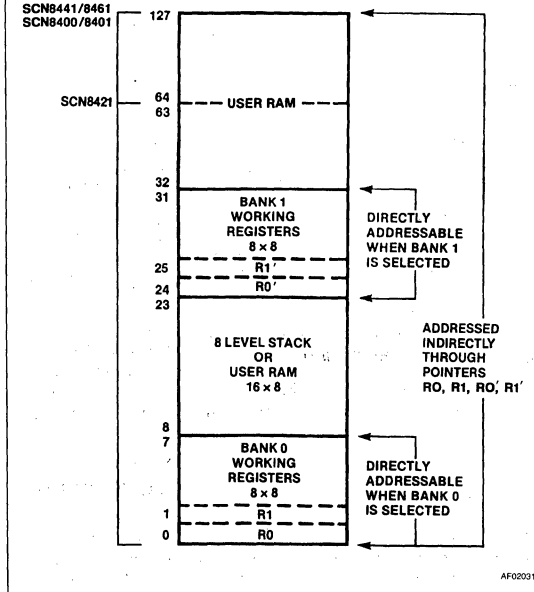
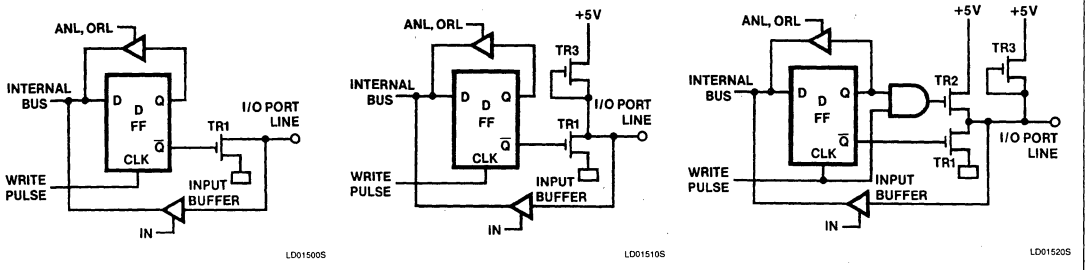


Figure 4. Data Memory Map



Option 1
Open Drain Output Without Pull-Up Transistor

Option 2
Open Drain Output With Pull-up Transistor

Option 3
Push-pull Output With Pull-up Transistor

Figure 5. Quasi-Bidirectional I/O Interface Types

If the inputs and outputs on a port are mixed (mixed-mode), the inputs should be options 1 or 2 but not option 3. This prevents cross-currents via TR2 and an external connection to ground, while switching the output on the same port and in parallel, masking the inputs with logic ones.

Serial I/O Interface

The SCN84XX family serial I/O interface has been designed to eliminate the heavy processing load imposed on a normal microcontroller performing serial data transfer. Whereas a normal microcontroller must regularly monitor the serial data bus for the presence of data, the serial I/O interface detects, receives and converts the serial data stream into a parallel format without interrupting execution of the current program. An interrupt is sent to the microcontroller only when a complete byte is received. Then, the microcontroller reads the data byte in one instruction. Likewise, for transmission, the serial I/O interface performs parallel to serial conversion and subsequent serial output of the data, and the microcontroller is only interrupted in the execution of its programmed tasks when a complete byte has been transmitted. The design of the serial I/O interface allows any number of SCN84XX family devices and peripheral circuits with I^2C bus compatibility to be interconnected by the two-line serial bus. This is achieved by allocating a specific 7-bit address to each device and ensuring that a device reacts only to a message preceded by its own address or the general call address.

Address recognition is performed by the interface hardware so that the microcontroller only needs to be interrupted when a valid address is received. This saves significant processing time and memory space compared to a conventional microcontroller with a software serial interface. When the address facility is not required, for instance in a system with only two microcontrollers, direct data transfer is possible. In multimaster systems, an automatically invoked arbitration procedure prevents two or more devices transmitting simultaneously.

Figure 6 shows the serial I/O interface. The clock line of the serial bus has exclusive use of pin 3 (SCLK) while the data line shares pin 2 (serial data) with the I/O line P23 of port 2. When the serial I/O is enabled, P23 is disabled as a parallel port line (P23 and SCLK only open drain).

The microcontroller and interface communicate via the internal microcontroller bus and the serial interrupt request line. Data and information controlling the operation of the interface are stored in four registers:

- Data shift register S0
- Serial I/O interface status word S1
- Serial clock control word S2
- Address register

Data Shift Register S0

S0 is the shift register that converts serial data to parallel format and vice versa. A pending interrupt is generated only after a complete byte has been transmitted, or after a complete data byte, specific or general call address has been received. The most significant bit is transmitted first.

Status Word S1

S1 provides information about the state of the interface and stores interface control information from the microcontroller. The four most significant bits are common to both read and write instructions. Four read only status bits and four write only control bits occupy the four least significant bits.

MST and TRX — These bits determine the operating mode of the serial I/O interface (table 1).

Table 1. SERIAL I/O INTERFACE OPERATING MODES

MST	TRX	Mode
0	0	Slave receiver
1	0	Master receiver
0	1	Slave transmitter
1	1	Master transmitter

BB: Bus Busy — This bit indicates the bus status.

PIN: Pending Interrupt Not — PIN = 0 indicates that there is an interrupt pending. This causes a serial interrupt request when the serial interrupt mechanism is enabled.

ESO: Enable Serial Output — The ESO flag enables/disables the serial I/O interface:
ESO = 1 enables
ESO = 0 disables

BC0, BC1 and BC2 — These bits set the number of bits received or transmitted in a serial data stream.

NOTE:
Bits ESO, BC0, BC1 and BC2 are write only.

AL: Arbitration Lost — The AL flag is set via the hardware when the serial I/O interface, as a master transmitter, loses the bus arbitration procedure.

ASS: Addressed as Slave — This flag is set via the hardware when the interface detects either its own address or the general call address as the first byte of a transfer, and if the interface has been programmed to operate in the address recognition mode.

AD0: Address Zero — This flag is set via the hardware after the general call address is detected when the interface is operating in the address recognition mode.

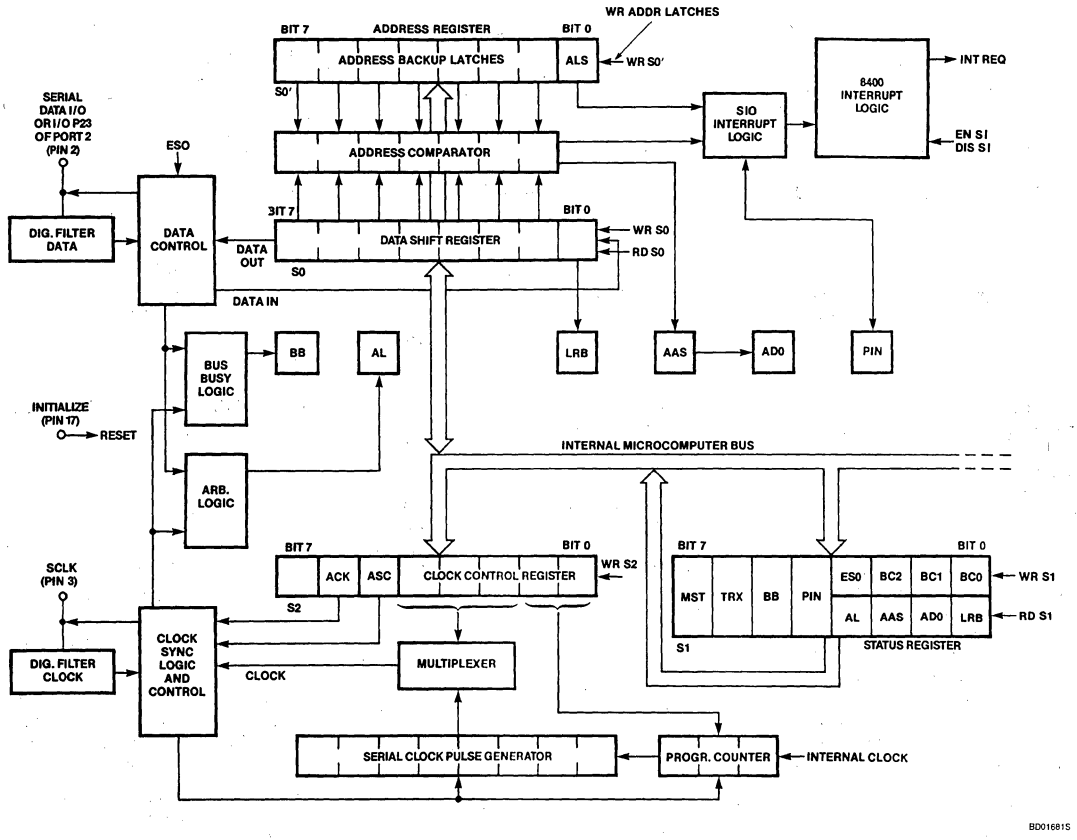


Figure 6. Serial I/O Interface

LBR: Last Bit Received — This contains either the last data bit received or, for a transmitting device in the acknowledge mode, the acknowledge from the receiving device.

NOTE:
Bits AL, AAS, AD0 and LBR are read only.

Clock Control Register S2
Bits 0 to 4 of S2 are used to set the frequency of the serial clock signal. When a 4.43MHz crystal is used, the frequency of the serial clock can be varied between 114kHz and 720Hz. An asymmetrical clock with a high to low ratio of 3 to 1 is produced by setting bit 5. The asymmetrical clock allows a microcontroller more time per clock period for sampling the data line, making the timing of this action less critical. Bit 6 is used to activate the acknowledge mode of the serial I/O. S2 is a write-only register.

Address Register
The address register contains the 7-bit address backup latches and the bit (ALS) used to enable/disable the address recognition mode. Only when ESO = 0 can the address register be written using the MOV S0,A and MOV S0,#data instructions.

Serial I/O Interrupt Logic
The interrupt logic is enabled by the EN SI instruction and disabled by DIS SI. When the interrupt logic is enabled, a pending interrupt results in a serial I/O interrupt to the controller, causing a jump to location 5 in the ROM. When the logic is disabled, the presence of an interrupt is still indicated by the PIN bit in register S1. Therefore, an interrupt can still be serviced but a vectored interrupt will not occur.

Interrupt System

External events and real-time on-chip peripherals require servicing by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, three single-level nested interrupts are provided.

Each interrupt vectors to a separate location in the program memory for its service program. Each source can be individually enabled or disabled. When more than one interrupt occurs simultaneously, their priority will be: (1) external, (2) serial I/O, and (3) timer/event counter. An additional external interrupt can be created using the timer/event counter interrupt.

Test Input T1

The T1 input line can be used as:

- A test input for branch instructions
- An input for zero voltage cross-over detection
- An external input to the event counter

An internal pull-up transistor is provided as a ROM mask option. This is useful when the input is from a switch or standard TTL output.

When T1 is used as a test input, the JT1 or JNT1 instructions test for a high or a low, respectively. The T1 input has a self-biasing circuit that can detect when an AC signal crosses zero (within ±100mV when coupled through a 1μF capacitor — typical design values are not guaranteed). The maximum input voltage is 3V (peak-to-peak), the maximum frequency is 1kHz. Zero cross-over detection used in conjunction with the timer/event counter interrupt is useful in thyristor control of power equipment. The operation of T1 as an input to the timer/event counter is described under the heading Timer/Event Counter.

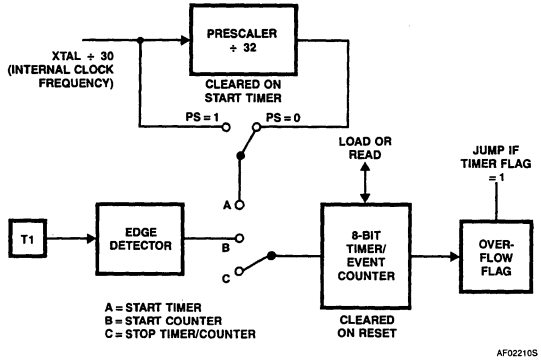
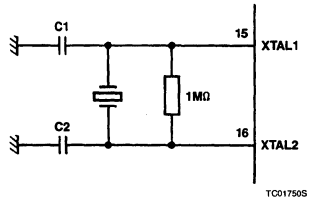


Figure 7. Timer/Event Counter

3

Table 2. DIFFERENCES BETWEEN THE MAB8021, SCN8400, AND SCN8400 SERIES MICROCONTROLLERS

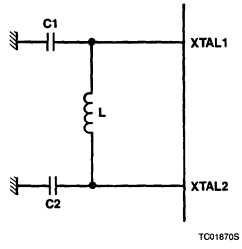
	8021	8048	8400, 8401, 8421, 8441, 8461
Parallel I/O lines	8 + 8 + 4	8 + 8 + 8	8 + 8 + 4
Single inputs	1	3	2
Serial I/O	No	No	Yes, 2-line multi-transmitter
Timer	8 bit	8 bit	8 bit
Prescaler	Modulo 32	Modulo 32	Modulo 1 and modulo 32
Machine cycle time (μs) for clock (MHz)	10 3	2.5 6	5 6
Instruction set	8021	8048	8048 with omissions 5 new serial I/O instructions 2 new register instructions 2 new control instructions 1 new cond. branch instruction
Interrupts	None	2 external timer/event counters	3 external serial I/O timer/event counters
No. of pins (DIP)	28	40	28



C1 = C2 = 27pF

C1 may be trimmed
Crystal – AT cut
or
Ceramic resonator

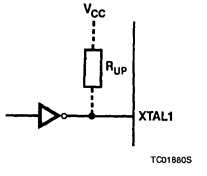
a. Quartz Crystal or Ceramic Resonator Mode



LC Oscillator Timing

Frequency	C1 = C2	L
3.0MHz	33pF	100μH
4.0MHz	33pF	56μH
4.4MHz	33pF	47μH
5.0MHz	33pF	33μH
6.0MHz	33pF	22μH

b. LC Pi-Network



- Drive XTAL1
- Leave XTAL2 open
- Driver may be high-speed CMOS or if TTL t_r, t_f < 10ns

c. External Drive

Figure 8

High Current Outputs

Ten pins are provided that can sink high currents (typical values):

P23 (serial data), pin 2 — 5mA at 0.45V (open drain)

SCLK, pin 3 — 5mA at 0.45V (open drain)

P10 – 17 — 10mA at 1V (except 8400)

P10 – P17 can be connected in parallel if their logic outputs are always the same.

Timer/Event Counter

An 8-bit binary up-counter is provided. This can count external events, machine cycles divided by 32, or machine cycles directly. When used as a timer, the input to the counter is either the overflow or input of a 5-bit prescaler. When used as an event counter, low to high transitions on T1 (pin 13) are counted. The maximum rate at which the counter may be incremented is once every machine cycle (200kHz for a 5μs machine cycle). Figure 7 illustrates the timer/event counter.

OSCILLATOR CIRCUITRY

Clock frequency is determined by using the internal oscillator or by connecting an external clock to XTAL1. Where the internal oscillator is used the frequency is set by a crystal between XTAL1 and XTAL2, or by a ceramic resonator or an inductor, each with two associated capacitors, between XTAL1 and XTAL2 (see figure 8). A machine cycle consists of 10 states, each state being 3 oscillator periods. The common 6MHz crystal gives a 5μs machine cycle. The SCN84XX family has dynamic logic, and therefore, for adequate refreshing the oscillator frequency must be at least 1MHz.

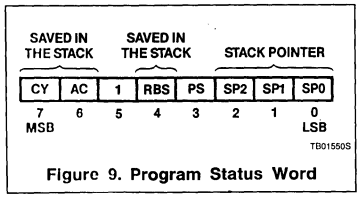


Figure 9. Program Status Word

Table 3. CONDITIONAL BRANCHES

TEST	JUMP CONDITION	JUMP INSTRUCTION
Accumulator	0 or non-zero	JZ, JNZ
Accumulator bit test	1	JB0 to JB7
Carry flag	0 or 1	JNC, JC
Timer overflow flag	1	JTF
Test input $\overline{INT}$	0 or 1	JNI, JI
Test input T1	0 or 1	JNT1, JT1
Test flag 0	1	JF0
Test flag 1	1	JF1
Register	Non-zero	DJNZ

PROGRAM STATUS WORD

The program status word (PSW) is an 8-bit word in the CPU which stores information about the current status of the microcontroller (figure 9). The PSW bits are:

Bits 0, 1, 2 — Stack pointer bits (SP0, SP1, SP2).

Bit 3 — Prescaler select (PS): 0 = divide-by-32 1 = no prescaling.

Bit 4 — Working register bank select (RBS):
0 = register bank 0
1 = register bank 1

Bit 5 — Not used

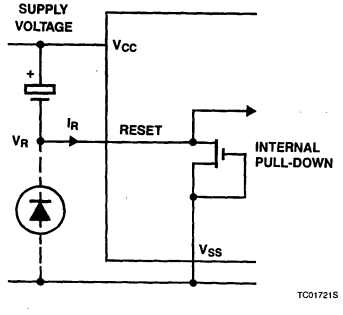
Bit 6 — Auxiliary carry (AC): Half-carry bit is generated by an ADD instruction and used by the decimal adjust instruction DA A.

Bit 7 — Carry (CY): The carry flag indicates that the previous operation has resulted in an overflow of the accumulator.

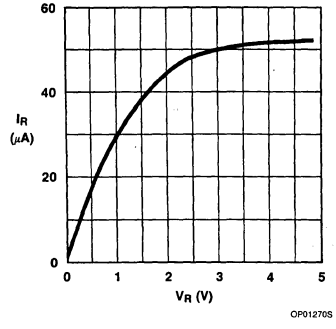
All bits can be read and written using the MOV A,PSW and MOV PSW,A instructions, respectively.

Bits 6 and 7 can be set and cleared by CPU operation. Bit 4 is changed by the SEL RB instruction, bit 3 by the MOV PSW,A instruction, and bits 0, 1 and 2 by the CALL, RET or RETR instructions and when an interrupt occurs. Bits 4, 6 and 7 are stored in the program counter stack during subroutine and interrupt calls. These bits are restored to the PSW with RETR (return and restore) instruction.

NOTE:
The RET instruction has no restore feature and should not be used at the end of an interrupt because this would leave any further interrupts disabled.



a. Typical Power-On Reset Circuitry



b. Typical Input Characteristics (Not Guaranteed)

Figure 10

The SCN84XX family has arithmetic, logical and branching capabilities. The DA A, SWAP A, and XCHD instructions simplify BCD arithmetic and the handling of nibbles. The MOVP A,@A instruction permits efficient table look-up from the current ROM page.

The conditional branch logic within the processor enables several conditions, internal and external to the processor, to be tested by the user's program. Table 3 lists the conditional branch instructions used to change the program execution sequence. The DJNZ instruction decrements a designated register and branches if the contents are not zero. This instruction makes the register an efficient program loop counter. The JMPP @A instruction allows multiway branches to destinations indirectly addressed by the contents of the accumulator.

RESET

A positive-going signal on the RESET input:

- Sets the program counter to zero.
- Selects location 0 of memory bank 0, and register bank 0.
- Sets the stack pointer to zero ('000'B); pointing to RAM address 8.
- Disables the interrupts (external, timer and serial I/O).
- Stops the timer/event counter, then sets it to zero.
- Sets the timer prescaler to divide-by-32.
- Resets the timer flag.
- Sets all ports to logic 1 (input mode).
- Sets the serial I/O to slave receiver mode and disables serial I/O.

The external power-on-reset circuit can consist of a capacitor connected between V_{CC} and the RESET pin. A diode can be added between the RESET pin and ground to ensure reset if the supply voltage falls momentarily. Figures 10a and 10b show a typical reset circuit and input characteristics of the RESET pin.

RESET has to be active high for more than two machine cycles after the power supply and clock have stabilized.

3

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER		RATING	UNIT
Operating ambient temperature[2]	SCN84XXAC	0 to +70	°C
	SCN84XXAA	−40 to +85	°C
	SCN84XXAE	−40 to +110	°C
Storage temperature		−65 to +150	°C
Input voltage on any pin with respect to ground (V_{SS})[3]		−0.5 to +7.0	V
Maximum input/output current		10	mA
Total power dissipation		1	W

DC ELECTRICAL CHARACTERISTICS $V_{CC} = 5.0V \pm 10\%$, $V_{SS} = 0V$[4,5]

PARAMETER			TEST CONDITIONS	LIMITS		UNIT
				Min	Max	
I_{CC}	Supply current	SCN84XXAC	0 to +70°C	−	85	mA
		SCN84XXAA	−40 to +85°C	−	100	mA
		SCN84XXAE	−40 to +110°C	−	100	mA
V_{IL}	Input voltage low (except P23 and SCLK)			−0.5	0.8	V
V_{IL1}	Input voltage low (P23 and SCLK)			−0.5	1.5	V
V_{IH}	Input voltage high (all inputs except (XTAL1, P23 and SCLK)			2	V_{CC}	V
V_{IH1}	Input voltage high (XTAL1, P23 and SCLK)			3.0	V_{CC}	V
V_{OL}	Output voltage low (P00 − P07) (P10 − P17)		I_{OL} = 1.6mA	−	0.45	V
V_{OL11}	Output voltage low (P10 − P17 for 8400)		I_{OL11} = 7mA	−	2.5	V
V_{OL12}	Output voltage low (P10 − P17 for 8401/21/41/61)		I_{OL12} = 10mA	−	1.0	V
V_{OL2}	Output voltage low (P20 − P22)		I_{OL2} = 1.6mA	−	0.45	V
V_{OL3}	Output voltage low (P23, SCLK)		I_{OL3} = 5mA	−	0.45	V
V_{OH}	Output voltage high (all outputs unless open drain)[6]		I_{OH} = −50μA	2.4	−	V
I_{OL}	Output leakage current		$V_{SS} < V_I < V_{CC}$	−	±10	μA

AC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V± 10%, V_{SS} = 0V [4,5]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
f_{XTAL} Frequency	SCN84XXAC, SCN84XXAA		1	6	MHz
	SCN84XXAE		1	5	MHz
t_{CY}	SCN84XXAC, SCN84XXAA		5	30	μs
	SCN84XXAE		6	30	μs

AC ELECTRICAL CHARACTERISTICS (SCN8400 and SCN8401) V_{CC} = 5.0V± 10% [4,5]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
t_{CC}	Control pulse duration $\overline{PSEN}$ (9 clock periods – CP)		1500	9000	ns
t_{AS}	Address to $\overline{PSEN}$ L setup (1CP)		167	–	ns
t_{DS}	Data to $\overline{PSEN}$ H setup (1CP + 120ns)		600	–	ns
t_{DR}	Data hold time		0	–	ns
t_{AD}	Address to data-in (10CP – t_{DS})		–	1070	ns
t_{PC}	Time from $\overline{PSEN}$ L to C1 (3CP)		500	–	ns
t_{IP0}	Time from $\overline{INTA}$ L to $\overline{PSEN}$ (3CP)		500	–	ns
t_{IP1}	Time from $\overline{INTA}$ H to $\overline{PSEN}$ (6CP)		1000	–	ns
t_{HS}	$\overline{HALT}$ setup to $\overline{PSEN}$ (15CP)		2500	–	ns
t_{HH}	$\overline{HALT}$ hold time from $\overline{PSEN}$ (3CP)		500	–	ns

T1 ZERO-CROSS CHARACTERISTICS V_{CC} = 5V± 10%, V_{SS} = 0V, C_L = 80pF [4,5]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{ZX}	Zero-cross detection input (T1) peak-to-peak	AC coupled, C = 0.2μF	1	3	V
A_{ZX}	Zero-cross accuracy	50Hz sine wave	–	±135	mV
F_{ZX}	Zero-cross detection input frequency (T1)		0.05	1	kHz

NOTES:

1. Stresses above those listed under absolute maximum ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other conditions above those indicated in the operation section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltage larger than the rated maximum.
3. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and are checked at 0.8V and 2.0V as appropriate.
6. P10/SDA and P11/SCL have open-drain outputs.

INSTRUCTION SET

The instruction set consists of over 80 one and two byte instructions and is based on the SCN8048 instruction set. New instructions include those for serial I/O operation and memory bank selection. Program code efficiency is high because all RAM locations on a 256 byte page require only a single byte address.

Table 4 gives the instruction set of the SCN8400 Series. Table 5 shows additional SCN84XX family instructions (including the five for serial I/O operation) that are not a part of the SCN8048 instruction set. Table 6 shows SCN8048 instructions omitted from the SCN8400 Series instruction set. Table 7 gives the SCN8400 Series instruction map.

The following symbols and abbreviations are used.

SYMBOL	DESCRIPTION
A	Accumulator
AC	Auxiliary carry flag
addr	Program memory address (11 bits)
Bb	Bit designation (b = 0 – 7)
BS	Bank switch
C	Carry flag
CLK	Clock signal
CNT	Event counter
D	Nibble designation (4 bits)
DBF	Program memory bank flip-flop
data	Number or expression (8 bits)
F0, F1	Flags 1 and 1
I	Interrupt
$\overline{INT}$	External interrupt

SYMBOL	DESCRIPTION
P	In-page operation designation
Pp	Port designation (p = 1, 2 or 4 – 7)
PSW	Program status word
Rr	Register designation (r = 0, 1 or 0 – 7)
SP	Stack pointer
T	Timer
TF	Timer flag
T0, T1	Test 0 and 1 inputs
#	Immediate data prefix
@	Indirect address prefix
$	Current value of program counter
←	Is replaced by
↔	Is exchanged with

3

Table 4. SCN8400 SERIES INSTRUCTION SET

MNEMONIC	FUNCTION		DESCRIPTION	OPCODE (HEX)	BYTES/ CYCLES	NOTES
Accumulator						
ADD A, Rr	(A) ← (A) + (Rr)	r = 0 – 7	Add register contents to A	6*	1/1	1
ADD A, @Rr	(A) ← (A) + ((R0)) (A) ← (A) + ((R1))		Add RAM data, addressed by Rr, to A	60 61	1/1	1
ADD A, #data	(A) ← (A) + data		Add immediate data to A	03 data	2/2	1
ADDC A, Rr	(A) ← (A) + (Rr) + (C)	r = 0 – 7	Add carry and register contents to A	7*	1/1	1
ADDC A, @Rr	(A) ← (A) + ((R0)) + (C) (A) ← (A) + ((R1)) + (C)		Add carry and RAM data, addressed by Rr, to A	70 71	1/1	1
ADDC A, #data	(A) ← (A) + data + (C)		Add carry and immediate data to A	13 data	2/2	1
ANL A, Rr	(A) ← (A) AND (Rr)	r = 0 – 7	AND Rr with A	5*	1/1	
ANL A, @Rr	(A) ← (A) AND ((R0)) (A) ← (A) AND ((R1))		AND RAM data, addressed by Rr, with A	50 51	1/1	
ANL A, #data	(A) ← (A) AND data		AND immediate data with A	53 data	2/2	
ORL A, Rr	(A) ← (A) OR (Rr)	r = 0 – 7	OR Rr with A	4*	1/1	
ORL A, @Rr	(A) ← (A) OR ((R0)) (A) ← (A) OR ((R1))		OR RAM data, addressed by Rr, with A	40 41	1/1	
ORL A, #data	(A) ← (A) OR data		OR immediate data with A	43 data	2/2	
XRL A, Rr	(A) ← (A) XOR (Rr)	r = 0 – 7	XOR Rr with A	D*	1/1	
XRL A, @Rr	(A) ← (A) XOR ((R0)) (A) ← (A) XOR ((R1))		XOR RAM, addressed by Rr, with A	D0 D1	1/1	
XRL A, #data	(A) ← (A) XOR data		XOR immediate data with A	D3 data	2/2	
INC A	(A) ← (A) + 1		Increment A by 1	17	1/1	
DEC A	(A) ← (A) – 1		Decrement A by 1	07	1/1	
CLR A	(A) ← 0		Clear A to zero	27	1/1	
CPL A	(A) ← not(A)		One's complement A	37	1/1	
RL A	$(A_{n+1}) \leftarrow (A_n)$ $(A_0) \leftarrow (A_7)$	n = 0 – 6	Rotate A left	E7	1/1	

Table 4. SCN8400 SERIES INSTRUCTION SET (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	OPCODE (HEX)	BYTES/ CYCLES	NOTES
Accumulator (Continued)					
RLC A	$(A_{n+1}) \leftarrow A_n$ n = 0 – 6 $(A_0) \leftarrow (C) \leftarrow (A_7)$	Rotate A left through carry	F7	1/1	2
RR A	$(A_n) \leftarrow (A_{n+1})$ n = 0 – 6 $(A_7) \leftarrow (A_0)$	Rotate A right	77	1/1	
RRC A	$(A_n) \leftarrow (A_{n+1})$ n = 0 – 6 $(A_7) \leftarrow (C), (C) \leftarrow (A_0)$	Rotate A right through carry	67	1/1	2
DA A		Decimal adjust A	57	1/1	2
SWAP A	$(A_{4-7}) \leftrightarrow (A_{0-3})$	Swap nibbles of A	47	1/1	
Data moves					
MOV A, Rr	(A) ← (Rr) r = 0 – 7	Move register contents to A	F*	1/1	
MOV A, @Rr	(A) ← ((R0)) (A) ← ((R1))	Move RAM data, addressed by Rr, to A	F0 F1	1/1	
MOV A, #data	(A) ← data	Move immediate data to A	23 data	2/2	
MOV Rr, A	(Rr) ← (A) r = 0 – 7	Move accumulator contents to register	A*	1/1	
MOV @Rr, A	((R0)) ← (A) ((R1)) ← (A)	Move accumulator contents to RAM location addressed by Rr	A0 A1	1/1	
MOV Rr, #data	(Rr) ← data	Move immediate data to Rr	B* data	2/2	
MOV @Rr, #data	((R0)) ← data ((R1)) ← data	Move immediate data to RAM location addressed by Rr	B0 data B1 data	2/2	
XCH A, Rr	(A) ↔ (Rr) r = 0 – 7	Exchange accumulator contents with Rr	2*	1/1	
XCH A, @Rr	(A) ↔ ((R0)) (A) ↔ ((R1))	Exchange accumulator contents with RAM data addressed by Rr	20 21	1/1	
XCHD A, @Rr	$(A_{0-3}) \leftrightarrow ((R0_{0-3}))$ $(A_{0-3}) \leftrightarrow ((R1_{0-3}))$	Exchange lower nibbles of A and RAM data addressed by Rr	30 31	1/1	
MOV A, PSW	(A) ← (PSW)	Move PSW contents to accumulator	C7	1/1	
MOV PSW, A	$(PSW_3) \leftarrow (A_3)$	Move accumulator bit 3 to PSW_3	D7	1/1	3
MOVP A, @A	$(PC_{0-7}) \leftarrow (A), (A) \leftarrow ((PC))$	Move indirectly addressed data in current page to A	A3	1/2	
Flags					
CLR C	(C) ← 0	Clear carry bit	97	1/1	2
CPL C	(C) ← not(C)	Complement carry bit	A7	1/1	2
Register					
INC Rr	(Rr) ← (Rr) + 1 r = 0 – 7	Increment register by 1	1*	1/1	
INC @Rr	((R0)) ← ((R0)) + 1 ((R1)) ← ((R1)) + 1	Increment RAM data, addressed by Rr, by 1	10 11	1/1	
DEC Rr	(Rr) ← (Rr) – 1 r = 0 – 7	Decrement register by 1	C*	1/1	
DEC @Rr	((R0)) ← ((R0)) – 1 ((R1)) ← ((R1)) – 1	Decrement RAM data, addressed by Rr, by 1	C0 C1	1/1	
Branch					
JMP addr	$(PC_{8-10}) \leftarrow addr_{8-10}$ $(PC_{0-7}) \leftarrow addr_{0-7}$ $(PC_{11-12}) \leftarrow$ MBFF 0 – 1	Unconditional jump within a 2K bank	• 4 address	2/2	
JMPP @A	$(PC_{0-7}) \leftarrow ((A))$	Indirect jump within a page	B3	1/2	

Table 4. SCN8400 SERIES INSTRUCTION SET (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	OPCODE (HEX)	BYTES/ CYCLES	NOTES
Branch (Continued)					
DJNZ Rr, addr	(Rr) – (Rr) – 1 r = 0 – 7 if (Rr) not zero (PC_{0-7}) ← addr	Decrement Rr by 1 and jump if not zero to addr	E* address	2/2	
DJNZ @Rr, addr	((R0)) ← ((R0)) – 1 if ((R0)) not zero (PC_{0-7}) ← addr	Decrement RAM data, addressed by Rr, by 1 and jump if not zero to addr	E0	2/2	
	((R1)) ← ((R1)) – 1 if ((R1)) not zero (PC_{0-7}) ← addr		E1		
JBb addr	if b = 1: (PC_{0-7}) ← addr b = 0 – 7	Jump to addr if Acc. bit b = 1	▲ 2 address	2/2	
JC addr	if C = 1: (PC_{0-7}) ← addr	Jump to addr if C = 1	F6 address	2/2	
JNC addr	if C = 0: (PC_{0-7}) ← addr	Jump to addr if C = 0	E6 address	2/2	
JZ addr	if A = 0: (PC_{0-7}) ← addr	Jump to addr if A = 0	C6 address	2/2	
JNZ addr	if A ≠ 0: (PC_{0-7}) ← addr	Jump to addr if A is not zero	96 address	2/2	
JT0 addr	if T0 = 1: (PC_{0-7}) ← addr	Jump to addr if T0 = 1	36 address	2/2	
JNT0 addr	if T0 = 0: (PC_{0-7}) ← addr	Jump to addr if T0 = 0	26 address	2/2	
JT1 addr	if T1 = 1: (PC_{0-7}) ← addr	Jump to addr if T1 = 1	56 address	2/2	
JNT1 addr	if T1 = 0: (PC_{0-7}) ← addr	Jump to addr if T1 = 0	46 address	2/2	
JTF addr	if TF = 1: (PC_{0-7}) ← addr	Jump to addr if Timer Flag = 1	16 address	2/2	4
JNTF addr	if TF = 0: (PC_{0-7}) ← addr	Jump to addr if Timer Flag = 0	06 address	2/2	
Timer/event counter					
MOV A, T	(A) ← (T)	Move timer/event counter contents to accumulator	42	1/1	
MOV T, A	(T) ← (A)	Move accumulator contents to timer/event counter	62	1/1	
STRT CNT		Start event counter	45	1/1	
STRT T		Start timer	55	1/1	
STOP TCNT		Stop timer/event counter	65	1/1	
EN TCNTI		Enable timer/event counter interrupt	25	1/1	
DIS TCNTI		Disable timer/event counter interrupt	35	1/1	
Control					
EN I		Enable external interrupt	05	1/1	
DIS I		Disable external interrupt	15	1/1	
SEL RB0	(RBS) ← 0	Select register bank 0	C5	1/1	5
SEL RB1	(RBS) ← 1	Select register bank 1	D5	1/1	5
SEL MB0	(MBFF0) ← 0, (MBFF1) ← 0	Select program memory bank 0	E5	1/1	
SEL MB1	(MBFF0) ← 1, (MBFF1) ← 0	Select program memory bank 1	F5	1/1	
SEL MB2	(MBFF0) ← 0, (MBFF1) ← 1	Select program memory bank 2	A5	1/1	
SEL MB3	(MBFF0) ← 1, (MBFF1) ← 1	Select program memory bank 3	B5	1/1	

Table 4. SCN8400 SERIES INSTRUCTION SET (Continued)

MNEMONIC	FUNCTION	DESCRIPTION	OPCODE (HEX)	BYTES/ CYCLES	NOTES
Subroutine					
CALL addr	((SP)) ← (PC), ($PSW_{4,6,7}$) (SP) ← (SP) + 1 (PC_{9-10}) ← $addr_{8-10}$ (PC_{0-7}) ← $addr_{0-7}$ (PC_{11-12}) ← MBFF 0 – 1	Jump to subroutine	▲ 4 address	2/2	6
RET	(SP) ← (SP) – 1 (PC) ← ((SP))	Return from subroutine	83	1/2	6
RETR	(SP) ← (SP) – 1 ($PSW_{4,6,7}$) + (PC) ← ((SP))	Return from interrupt and restore bits 4, 6, 7 of PSW	93	1/2	6
Parallel input/output					
IN A, P_P	(A) ← (P0) (A) ← (P1) (A) ← (P2)	Input port p data to accumulator	08 09 0A	1/2	7
OUTL P_P, A	(P0) ← (A) (P1) ← (A) (P2) ← (A)	Output accumulator data to port p	38 39 3A	1/2	
ANL P_P, #data	(P0) ← (P0) AND data (P1) ← (P1) AND data (P2) ← (P2) AND data	AND port p data with immediate data	98 99 9A	2/2	
ORL P_P, #data	(P0) ← (P0) OR data (P1) ← (P1) OR data (P2) ← (P2) OR data	OR port p data with immediate data	88 89 8A	2/2	
OUTL P0, A	(P0) ← (A)	Output accumulator data to port ϕ	90	1/2	
Serial input/output					
MOV A, S_n	(A) ← (S0) (A) ← (S1)	Move serial I/O register contents to accumulator	0C 0D	1/2	8
MOV S_n, A	(S0) ← (A) (S1) ← (A) (S2) ← (A)	Move accumulator contents to serial I/O register	3C 3D 3E	1/2	
MOV S_n, #data	(S0) ← data (S1) ← data (S2) ← data	Move immediate data to serial I/O register	9C 9D 9E	2/2	
EN SI		Enable serial I/O interrupt	85	1/1	
DIS SI		Disable serial I/O interrupt	95	1/1	
NOP		No operation	00	1/1	

NOTES:
1. PSW CY, AC (affected)
2. PSW CY (affected)
3. PSW PS (affected)
4. Execution of JTF and JNTF instructions resets the timer flag (TF).
5. PSW RBS (affected)
6. PSW SP_0, SP_1, SP_2 (affected)
7. (A) = 1111 P23, P22, P21, P20.
8. (S1) has a different meaning for read and write operation, see serial I/O interface.

*– 8, 9, A, B, C, D, E, F
●– 0, 2, 4, 6, 8, A, C, E
▲– 1, 3, 5, 7, 9, B, D, F

Table 5. SCN8400 SERIES INSTRUCTIONS NOT IN THE SCN8048 INSTRUCTION SET

SERIAL I/O	REGISTER	CONTROL	CONDITIONAL BRANCH

MOV A, S_n
MOV S_n, A
MOV S_n, #data
EN SI
DIS SI
	DEC @Rr
	DJNZ @Rr, addr
		SEL MB2
		SEL MB3
			JNTF addr

Table 6. SCN8048 INSTRUCTIONS NOT IN THE SCN8400 SERIES INSTRUCTION SET

DATA MOVES	FLAGS	BRANCH	CONTROL

MOVX A, @Rr
MOVX @R, A
MOVP3 A, @A
MOVD A, P
MOVD P, A
ANLD P, A
ORLD P, A
	CLR F0
	CPL F0
	CLR F1
	CPL F1
		*JNI addr
		JF0 addr
		JF1 addr
		
		*replaced by
		JT0 JNT0
			ENT0 CLK

Table 7. SCN8400 SERIES INSTRUCTION MAP

FIRST HEXADECIMAL CHARACTER OF OPCODE ↓	SECOND HEXADECIMAL CHARACTER OF OPCODE →															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP			ADD A, #data	JMP page 0	EN I	JNTF addr	DEC A	IN A, Pp 0 1 2				MOV A, S_n 0 1			
1	INC @Rr 0 1		JB0 addr	ADDC A, #data	CALL page 0	DIS I	JTF addr	INC A	INC Rr 0 1 2 3 4 5 6 7							
2	XCH A, @Rr 0 1			MOV A, #data	JMP page 1	EN TCNTI	JNT0 addr	CLR A	XCH A, Rr 0 1 2 3 4 5 6 7							
3	XCHD A, @Rr 0 1		JB1 addr		CALL page 1	DIS TCNTI	JT0 addr	CPL A	OUTL Pp, A 0 1 2				MOV S_n, A 0 1 2			
4	ORL A, @Rr 0 1		MOV A, T	ORL A, #data	JMP page 2	STRT CNT	JNT1 addr	SWAP A	ORL A, Rr 0 1 2 3 4 5 6 7							
5	ANL A, @Rr 0 1		JB 2 addr	ANL A, #data	CALL page 2	STRT T	JT1 addr	DA A	ANL A, Rr 0 1 2 3 4 5 6 7							
6	ADD A, @Rr 0 1		MOV T, A		JMP page 3	STOP TCNT		RRC A	ADD A, Rr 0 1 2 3 4 5 6 7							
7	ADDC A, @Rr 0 1		JB3 addr		CALL page 3			RR A	ADDC A, Rr 0 1 2 3 4 5 6 7							
8				RET	JMP page 4	EN SI			ORL Pp, #data 0 1 2							
9	OUTL P0, A		JB4 addr	RETR	CALL page 4	DIS SI	JNZ addr	CLR C	ANL Pp, #data 0 1 2				MOV S_n, #data 0 1 2			
A	MOV @Rr, A 0 1			MOVP A, @A	JMP page 5	SEL MB2		CPL C	MOV Rr, A 0 1 2 3 4 5 6 7							
B	MOV @Rr, #data 0 1		JB 5 addr	JMPP @A	CALL page 5	SEL MB3			MOV Rr, #data 0 1 2 3 4 5 6 7							
C	DEC @Rr 0 1				JMP page 6	SEL RB0	JZ addr	MOV A, PSW	DEC Rr 0 1 2 3 4 5 6 7							
D	XRL A, @Rr 0 1		JB6 addr	XRL A, #data	CALL page 6	SEL RB1		MOV PSW, A	XRL A, Rr 0 1 2 3 4 5 6 7							
E	DJNZ @Rr, addr 0 1				JMP page 7	SEL MB0	JNC addr	RL A	DJNZ Rr, addr 0 1 2 3 4 5 6 7							
F	MOV A, @Rr 0 1		JB7 addr		CALL page 7	SEL MB1	JC addr	RLC A	MOV A, Rr 0 1 2 3 4 5 6 7							

Signetics

Microprocessor Products

Section 4
Microsystems Products

INDEX

Signetics

SMVME0400
VMEbus Power Supply

Product Specification (Brief)

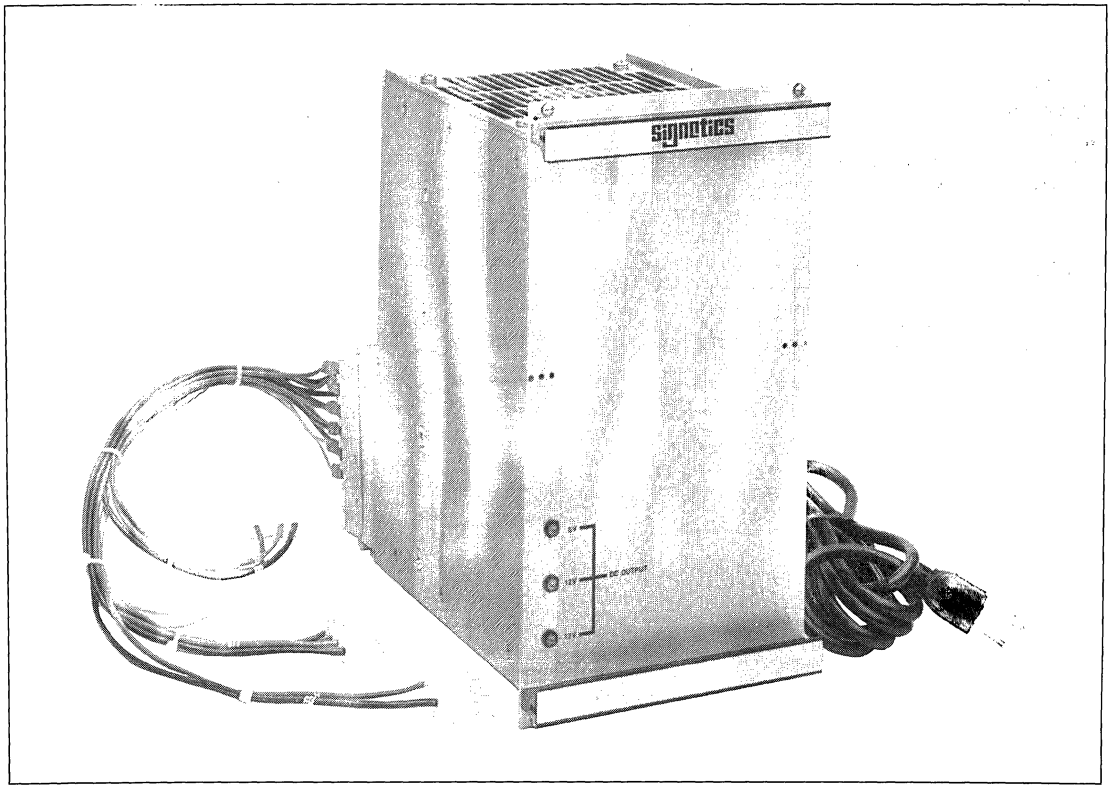
Microsystems Products

DESCRIPTION

The Signetics SMVME0400 is a switching power supply which provides +5VDC (at 30A), +12VDC (at 3.0A) and −12VDC (at 1.0A), and fully conforms to VMEbus specifications. The power supply consists of the power supply box itself (which contains an AC interface and DC boards, an AC filter, a temperature sensing switch for over-temperature protection, and feedback circuits for overload and overvoltage protection) and cable harness with connectors to connect the supply to a VMEbus backplane. The SMVME0400 Power Supply, when integrated into the Signetics SMVME0510 Card Cage, provides a perfect base for most VMEbus system applications.

FEATURES

- **VMEbus Compatible**
- **Selectable Input: 115VAC or 230VAC at 50 or 60Hz**
- **Output:**
  +5VDC at 30A
  +12VDC at 3.0A
  −12VDC at 1.0A
- **Overload Protection**
- **Overvoltage Protection**
- **Over Temperature Protection**



4

VMEbus Power Supply — SMVME0400

BASIC OPERATION

The incoming AC voltage is rectified and filtered to high-voltage low-current DC, which is chopped at high frequency, stepped down to low-voltage high-current AC, and again rectified for the DC output.

OVERLOAD PROTECTION

Overload protection is based on automatic fold-back current limiting. The output recovers to normal when overload is removed.

OVERVOLTAGE PROTECTION

Overload protection circuitry releases the output voltage to rated output voltage or less within 500μsec. Overvoltage protection is not triggered by turn-on, turn-off, power-fail or overload recovery. The circuitry may be reset by interrupting input power.

OVERTEMPERATURE PROTECTION (THERMAL SHUTDOWN)

The temperature-sensing switch interrupts AC power input if operating temperature is exceeded.

ORDERING INFORMATION

SMVME0400SD	VMEbus 200W Power Supply
SMMAN0400	VMEbus 200W Power Supply Manual

SMVME0400 VMEbus POWER SUPPLY SPECIFICATIONS

- Input Power: 100 to 130VAC RMS or 187 to 256VAC RMS single phase — 47 to 63Hz 380 watts (max)
- Output Power:

Voltage (VDC)	Max. Cur. (amps)	Total Reg. (VDC)	Noise & Ripple (mVDC p-p)
+5	30.0	+5.125 ± 0.125	75
+12	3.0	+12.20 ± 0.40	100
−12	1.0	−12.20 ± 0.40	100

  Total power output 200 watts.
- Overload Limit: +5V 33A (min) with all other outputs fully loaded; 63A (max) with all other outputs inloaded. +12V 110% of rated output current
- Overvoltage Limit: 125% ± 5% of nominal voltage
- Operating Temperature: 0°C to 50°C
- Power Hold-Up Time (Rated Load): 20ms (min) after loss of AC input
- Output Rise Time (10% to 90% of rated value): 30ms (min) after AC turn-on
- Dimensions (L x W x H): 10.3 x 5.6 x 6.5 in. (26.16 x 14.22 x 16.51cm)
- Weight: 11 lb (5kg)
- DC Power Harness and Line Cord Included

Signetics

SMVME0500/0510
Card Cage Assembly

Product Specification (Brief)

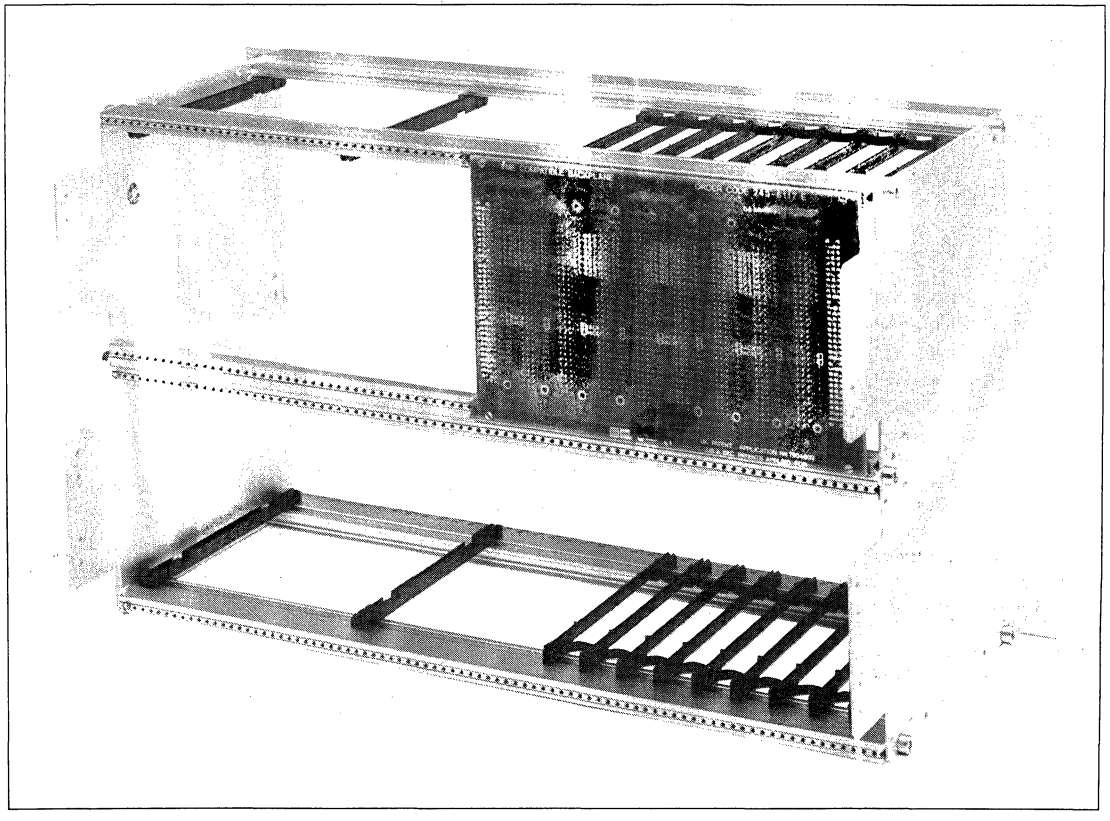
Microsystems Products

DESCRIPTION

The Signetics 9-slot VMEbus card cage assemblies offer the OEM a reliable, low noise mounting and interconnect facility for configuring VMEbus board based systems to meet the needs of nearly any application. The card cages allow standard board front panels to be locked in place with screws by utilizing the tapped strips on the front of the cage and are provided with side flanges to simplify mounting. The SMVME0500 Card Cage Assembly provides backplanes for both J1/P1 and J2/P2 connectors allowing designs utilizing the VMEbus in its fully expanded configuration (A32, D32). The SMVME0510 is built to standard 19 inch rack size and provides the J1/P1 backplanes and mounting rails for the SMVME0400 VMEbus Power Supply.

FEATURES

- **Meets all VMEbus specifications**
- **Accommodates up to nine double Eurocard size boards**
- **Compatible with standard VMEbus front panels and card ejectors**
- **Side flanges for easy mounting**
- **Tapped strips in front permit securing board front panels to cage**
- **SMVME0500 is provided with both J1/P1 and J2/P2 backplanes**
- **SMVME0510 is standard 19 inch rack mountable and includes the J1/P1 backplane and mounting rails for the SMVME0400 Power Supply**



4

Card Cage Assembly — SMVME0500/0510

BACKPLANE FEATURES

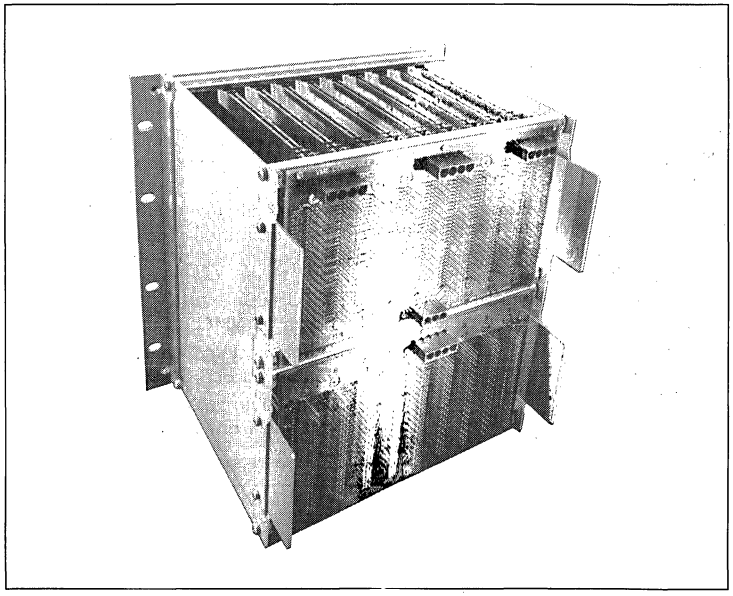
- Meets all VMEbus specifications
- J1/P1 backplane implemented on a five layer board with Signal-Ground-V_{CC}-Ground-Signal designed to ensure uniform impedance
- J2/P2 backplane implemented on a two layer board
- 64 pins of user-defined I/O on J2/P2 connector
- Ground lines between adjacent signals to minimize crosstalk
- Terminated at each end as defined in VMEbus specification
- All holes plated through
- Pressfit DIN 41612 connectors with wire wrap tails at each connector position

J1/P1 BACKPLANE SPECIFICATIONS

- Five layer epoxy glass construction Nominal thickness 2.4mm Dialectric to BS4584EP-GC-CU-3 (FR4)
- Copper thickness 35μm finish-plated Copper 50μm average
- Tin lead 8μm maximum
- Total 93μm outside layers only
- J1/P1 backplane power connections 1 – 3 pin polarized "mate-n-lock" (AMP P/N 350776-1) 3 – 4 pin polarized "mate-n-lock" (AMP P/N 350779-1)
- Power handling capacity of 27 amps 5 volts for J1/P1 backplane

J2/P2 BACKPLANE SPECIFICATIONS

- Two layer epoxy glass construction Nominal thickness 2.4mm
- Copper thickness 35μm finish-plated Copper 50μm average
- Tin lead 8μm maximum
- Total 93μm outside layers
- J2/P2 backplane power connections 1 – 4 pin polarized "mate-n-lock" (AMP P/N 350779-1)
- Power handling capacity of 9 amps 5 volts for J2/P2 backplane



CARD CAGE MECHANICAL SPECIFICATIONS

	SMVME0500	SMVME0510
• Overall Dimensions		
Height (with wire wrap pins)	278.9mm (10.98 in.)	278.9mm (10.98 in.)
Width (with mounting flanges)	268.5mm (10.575 in.)	482.6mm (19.0 in.)
Depth	18.2mm (7.15 in.)	18.2mm (7.15 in.)
• Number of Card Slots	9	9
• Card Format	Double Eurocard (160mm x 233.4mm)	same
• Weight	5 lbs	5 lbs
• Shipping Weight	7 lbs	7 lbs

ORDERING INFORMATION

SMVME0500	VMEbus 10.575 in. Card Cage Assembly, 9-slot, including J1/P1 and J2/P2 backplane
SMVME0510	VMEbus 19 in. Card Cage Assembly, 9-slot, including J1/P1 backplane and mounting rails for SMVME0400 Power Supply
SMVME0600	VMEbus Backplane Assembly, 9-slot, J2/P2 PC board with connectors and terminators
SMVME0700	VMEbus Backplane Assembly, 9-slot, J1/P1 PC board with connectors and terminators
SMMAN0010	VMEbus Architectural Specification
SMMAN05X0	VMEbus Card Cage Assembly User Manual

Signetics

SMVME1000
VMEbus Monitor Board

Product Specification (Brief)

Microsystems Products

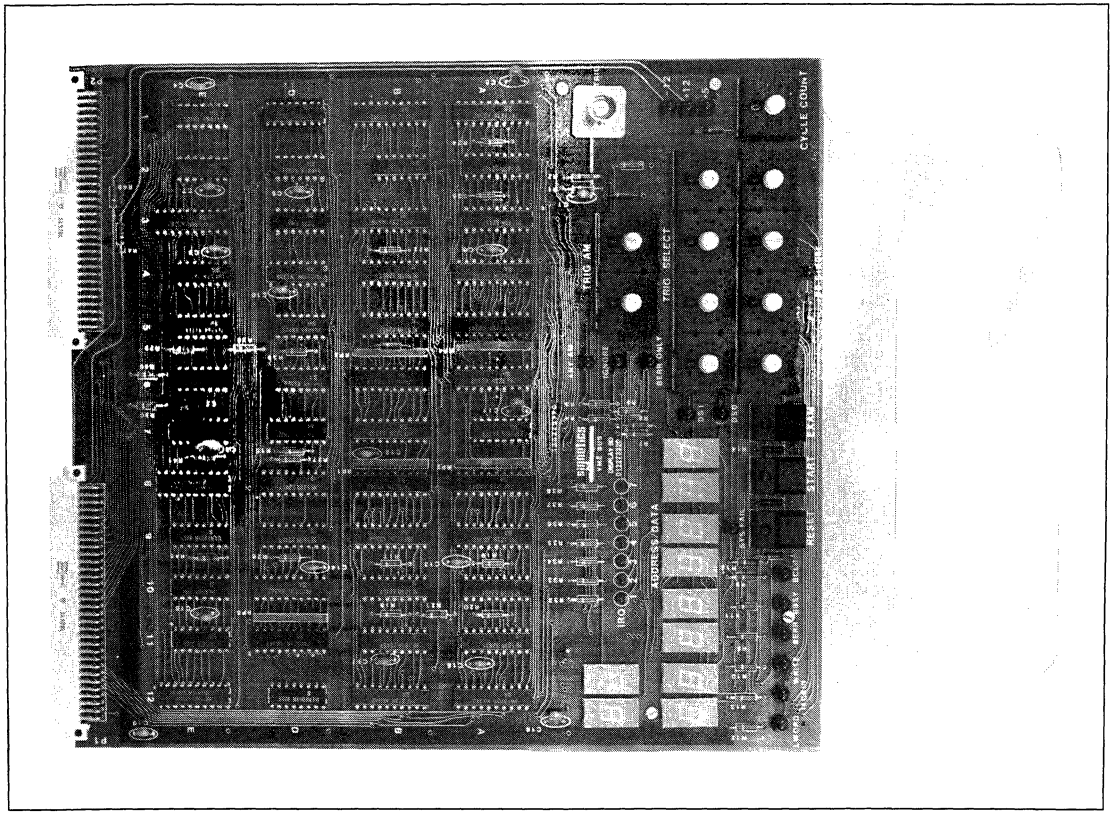
DESCRIPTION

The VMEbus Monitor Board is a low-cost diagnostic aid which provides a means of capturing specific bus transactions and displaying the bus signal activity relevant to the captured cycle. A single module, the Bus Monitor has been extended to allow convenient access to operator controls and displays. Address, address modifier, and data bus values are displayed as seven-segment hexadecimal characters. Light emitting diodes indicate other relevant bus conditions.

Comprised of operator controls, comparators, latches, display drivers and displays, the Signetics VMEbus Monitor Board provides an easily understood method of program tracing and hardware troubleshooting. A passive observer, the module is not capable of generating VMEbus signals other than SYSRESET. A reset push button is provided for convenience.

FEATURES

- **Monitors and Displays VMEbus Transfers**
  - **Easy-to-read seven-segment displays**
  - **LED display of relevant backplane signals**
- **Triggers on Address or Data Value**
  - **Hexadecimal rotary switches provide easy, accurate selection of trigger value**
  - **Trigger cycle may be qualified as READ ONLY, WRITE ONLY, or READ/WRITE**
- **Subsequent Cycle Capturing**
- **Operator Control of Bus Error Response**
- **Trigger Output for External Test Equipment**



4

VMEbus Monitor Board — SMVME1000

OPERATOR CONTROLS

The Signetics VMEbus Monitor Board provides hexadecimal rotary switches for trigger value selection. Two of these switches allow address modifier selection. Address modifier values greater than '3F' are used to control the compare circuitry, the display, and the effect BERR has on the VMEbus Monitor Board.

A one-of-eight rotary switch controls which subsequent cycle is to be captured after reaching the trigger cycle.

A three-position slide switch provides a means of qualifying the trigger cycle as READ ONLY, WRITE ONLY, or READ/WRITE.

Controls include a RESET push button, a START push button, and a push button labeled "EXAM". Depressing the EXAM button causes data bus values to be enabled to the comparators and the displays. When released, the address bus values are displayed and compared.

LATCHES

Each address bus signal and data bus signal is connected to a latch input. These latches store the bus states of the cycle being captured and the contents are displayed as determined by the position of the EXAM push button. Latches are also provided for VMEbus signals which are displayed via light-emitting diodes.

COMPARATOR

The Signetics VMEbus Monitor Board is capable of comparing short, standard, and extended address combinations to values selected by the operator. Address modifiers may be compared to selected values or ignored. Data bus comparison may be performed on words or long words. The result of a successful comparison is an enable for immediate or subsequent cycle capture. If BERR is enabled and asserted, an immediate extended capture is performed regardless of the state of the comparator. It is also possible to ignore BERR.

DISPLAYS

The Signetics VMEbus Monitor Board provides a direct hexadecimal readout of captured values via common cathode seven-segment displays. A two-speed scan circuit reduces the power required for display illumination and increases character brightness when a trigger cycle is encountered.

SPECIFICATIONS

Dimensions

233.4mm x 248mm

Power

1.2A typical (2.1A max) @ +5VDC
20mA @ +12VDC
20mA @ −12VDC

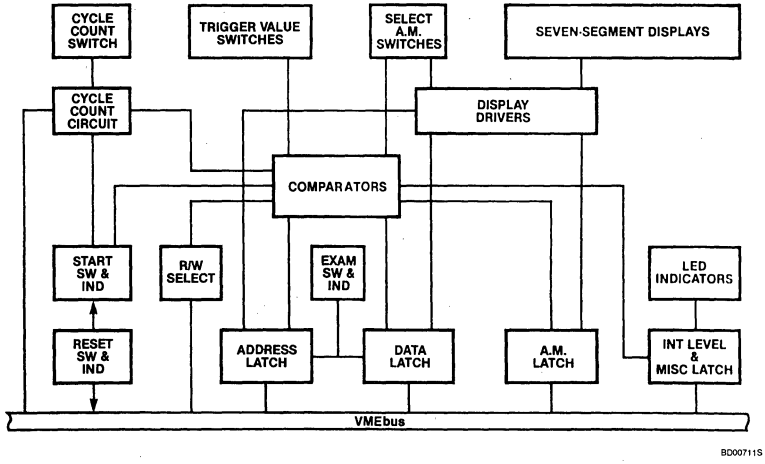
Temperature

0°C to +55°C operating

Humidity

8% to 90% non-condensing

The Signetics VMEbus Monitor Board is a slave device.



Signetics VMEbus Monitor Board SMVME1000

Signetics

SMVME1200/1201
VMEbus Prototype Board

Product Specification (Brief)

Microsystems Products

DESCRIPTION

The SMVME1200 is a high performance prototyping module designed to function as a VMEbus master or slave or dual ported master and slave. The prototyping board is designed to relieve today's schedule=pressured designers from getting deeply involved in the sophisticated details of VMEbus arbitration, control and interrupt=handler timings. This board will also help designers avoid VMEbus protocol violations and take advantage of advanced VMEbus features such as address look-a-head without having to become true "bus experts."

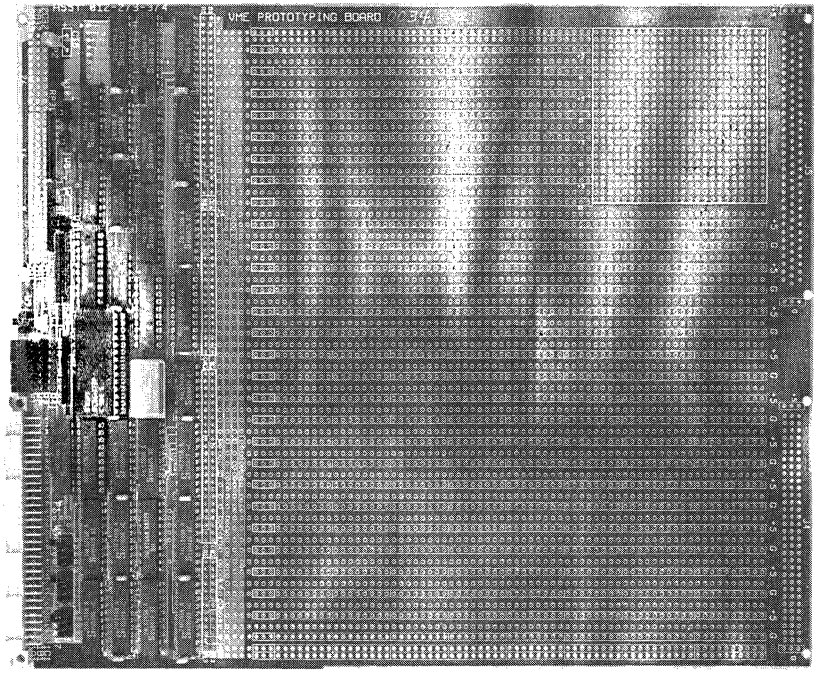
A hole grid pattern is provided to allow prototype logic design using wire wrap sockets. SMVME1200 prototype board circuits cover VMEbus Interface, Interrupter, and Interrupt Handler functions.

TWO FORM FACTORS

The prototyping module is available in two sizes, a standard double high VMEbus size and an extended 280mm version. The extended version full of wire wrapped circuitry can be roughly shrunk down to a standard VMEbus form factor using modern layout techniques.

FEATURES

- **Complete VMEbus Interface with VMEbus Controller SCB68172**
- **VMEbus System Controller Functions: Single Level Bus Arbiter**
- **Board Configurations:**
  - **Master Only**
  - **Slave Only**
  - **Dual Ported Master/Slave Processor Type**
  - **Dual Ported Master/Slave DMA Type**
- **Jumper Selectable Release of VMEbus Mastership, RWD or ROR**
- **Software Controlled Interrupt Generation for Off-Board Resources on any one of the Seven Jumper Selectable Levels**
- **Interrupt Handler for Up to Seven Levels of Off-Board or On-Board Interrupt Requests**



4

VMEbus Prototype Board — SMVME1200/1201

VMEBUS INTERFACE

This board supports jumper selectable level of bus request and bus grant daisy chain lines. The board provides VMEbus handshake, address, data, and control buffers, and direction control of these buffers. The usage of these buffers depends upon the selected type of board configuration. The possible choices of the board's configurations are given below:

- Board used as a master only.
- Board used as a slave only.
- Board used as a dual ported processor type master/slave.
- Board used as a dual ported DMA type master/slave.

The last two configurations support local slaves isolated from VMEbus and shared slaves dual ported between local controller and VMEbus. The board supports a jumper selectable VMEbus release mechanism for release on request or release when done.

INTERRUPTER

The board is designed to act as a VMEbus Interrupter. It is able to generate an interrupt request for VMEbus slaves under software control. This request can be made on any one of seven jumper selectable levels supported by VMEbus.

INTERRUPT HANDLER

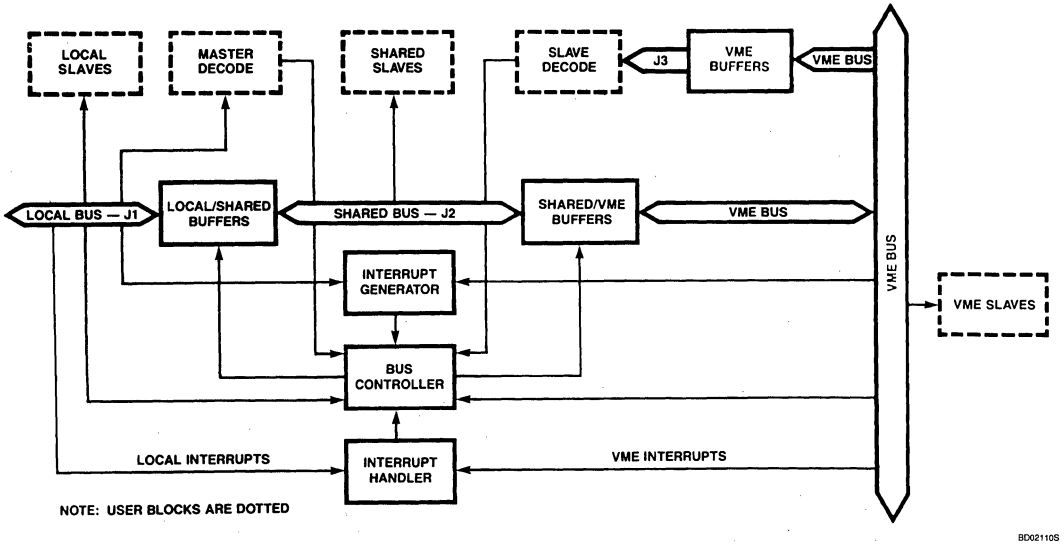
The board handles up to seven different levels of incoming interrupts from VMEbus. In addition to these seven, up to seven levels of local onboard interrupt requests. The higher level of interrupt source will have higher priority whether it is a bus interrupt source or local source. The local interrupt sources have priority over bus interrupt sources within the same level. These interrupt requests can be auto-vectored or the interrupting device can supply the vector.

USER CONNECTORS

Local Bus Connector J1 provides signals for user circuit to function as either local master or local slave for VMEbus based systems. Shared Bus Connector J2 Signals allow user circuits to share dual ported memory between local master/slave and other master/slave boards over the VMEbus. Connector J3 provides user signals to activate local slave circuit on the prototype board.

PROTOTYPING AREA

The board is of a four layer construction with internal power and ground planes. The prototyping area has a hole pattern suitable for 0.3, 0.6 and 0.9 inch DIPS with .1" pin spacing. A special grid is provided for up to four PGA (Pin Grid Array) type packages. Two connector patterns are provided for "D" shell type connectors and on 96-pin type C DIN connector or equivalent header.



NOTE: User blocks are dotted.

VMEbus Prototype Board SMVME1200/1201

CONFIGURATIONS

The personality of the SMVME1200 board is defined by setting jumpers. Master only, slave only, dual ported master/slave processor type, and dual ported master/slave DMA type configurations are provided.

SPECIFICATIONS

USAGE	1 per VMEbus card cage
DIMENSIONS	• 160mm x 233.3mm • 280mm x 233.3mm
*POWER	+5V @ 1 AMP
TEMPERATURE	0 to 55°C oporating
HUMIDITY	8% to 80% non-condonsing

*User circuit power limitations per VMEbus specifications.

ORDERING INFORMATION

SMVME1200	VMEbus Prototype Module (280mm x 233.3mm)
SMVME1201	VMEbus Prototype Module (160mm x 233.3mm)
SMMAN1200/1201	VMEbus Prototype Module User Manual

4

Signetics

SMVME1500
VMEbus System Controller

Product Specification (Brief)

Microsystems Products

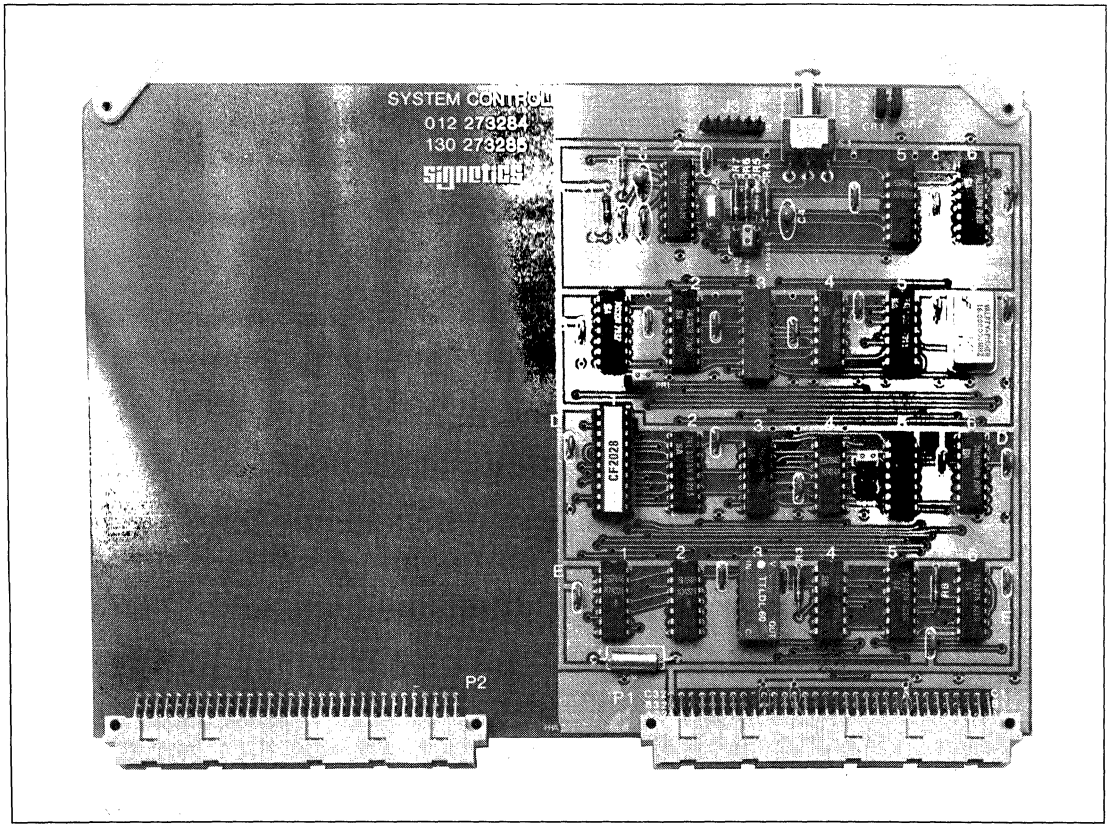
DESCRIPTION

The VMEbus System Controller Module is one of the fundamental building blocks in the Signetics Modular Board product family. It consolidates, in a single module, all of the bus arbitration, system utilities, and system monitors which the user finds needlessly replicated in other offerings. This implementation allows system designers the flexibility to utilize multiple Central Processing Modules, as their application dictates, without conflict or costly duplication of circuitry. The System Controller Module is particularly well suited to high-performance microcomputer systems which require multiprocessing and intelligent DMA facilities.

This product has been specifically designed to provide system level functions for VMEbus based systems while complementing the proven reliability of the DIN connectors and DIN form factor. The System Controller Module provides user selectable options which give the system designer a high degree of flexibility in optimizing the system configuration for a variety of 8-, 16-, and 32-bit applications.

FEATURES

- **Provides total bus arbitration**
  - **Selectable for Priority and Round-Robin modes on all four arbitration levels**
- **Supports orderly system shutdown**
  - **Provides early power fail notification**
- **Monitors system operation**
  - **Implements selectable watchdog timer and detects improper address conditions**
- **Provides all system utilities**
  - **Generates 16MHz system clock and 4MHz serial bus clock**
- **VMEbus compatible**



VMEbus System Controller — SMVME1500

BUS ARBITRATION

The System Controller Module includes all the circuitry necessary to support total bus arbitration in a VMEbus card cage. The arbitration protocol is user selectable for either Priority or Round-Robin modes of operation where the Priority mode grants preference to specific bus masters and the Round-Robin mode uniformly distributes bus usage among several bus masters.

The Priority mode of operation is applicable in a multiple user system where some tasks need more access to the system resources than do others.

The Round-Robin mode of operation is applicable in a multiple user system where all tasks need relatively equal access to the resources of the system.

In either mode a new bus master typically gains control of the system bus within 100ns of release by its predecessor.

AC POWER FAIL AND RESET

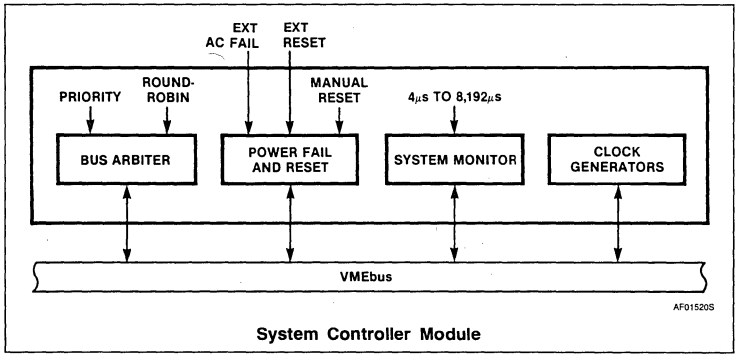
In order for systems without battery backup to support an orderly shutdown, advance notification of AC power loss is necessary. The System Controller Module accepts an AC power loss signal as an input and generates the necessary VMEbus signals to allow an orderly and recoverable system shutdown to be implemented. System reset is coordinated with the AC FAIL signal or can be manually initiated by a switch located on the module or from an external source such as a front panel.

SYSTEM OPERATION MONITORS

Application software errors may occur in systems from time to time. Those software errors often result in "system hangup" due to the software accidentally attempting to access a non-existent module or attempting to access a module in an incorrect manner. To aid in recovering from these errors the System Controller Module contains a watchdog timer and address verification circuitry. The watchdog timer is user selectable for a range of timeout values from 4μs to 8,192μs in 12 steps by powers of 2 which, if exceeded, will notify the bus master of an error condition. The address verification function monitors the bus and notifies the bus master of an error condition if an illegal address combination occurs.

SYSTEM UTILITIES

The System Controller Module supplies the independent 16MHz system clock which is available for general clocking functions by other modules in the card cage. This product also supplies the serial bus clock which is required by systems utilizing the serial data communication channel for intermodule information exchange.



System Controller Module

SPECIFICATIONS

USAGE	—	One per VMEbus card cage, mounted in SLOT 1
DIMENSIONS	—	160mm x 233.4mm
POWER	—	0.5A typ (0.8A max) @ +5Vdc
TEMPERATURE	—	0°C to 55°C operating
HUMIDITY	—	8% to 80% non-condensing
ALLOCATION	—	100ns typ, 130ns max
SPEED		
CLOCKS	—	16MHz 50% duty cycle, 4MHz 25% duty cycle
INDICATORS	—	BUS ERROR, SYSTEM FAIL
OPTIONS (static)	—	BUS ARBITRATION: Priority/Round-Robin TOUT: 4μs to 8,192μs

ORDERING INFORMATION

SMVME1500	VMEbus System Controller Module
SMMAN3015	VMEbus System Controller Module User Manual

4

Signetics

SMVME1600
VMEbus Quad I/O Module

Product Specification (Brief)

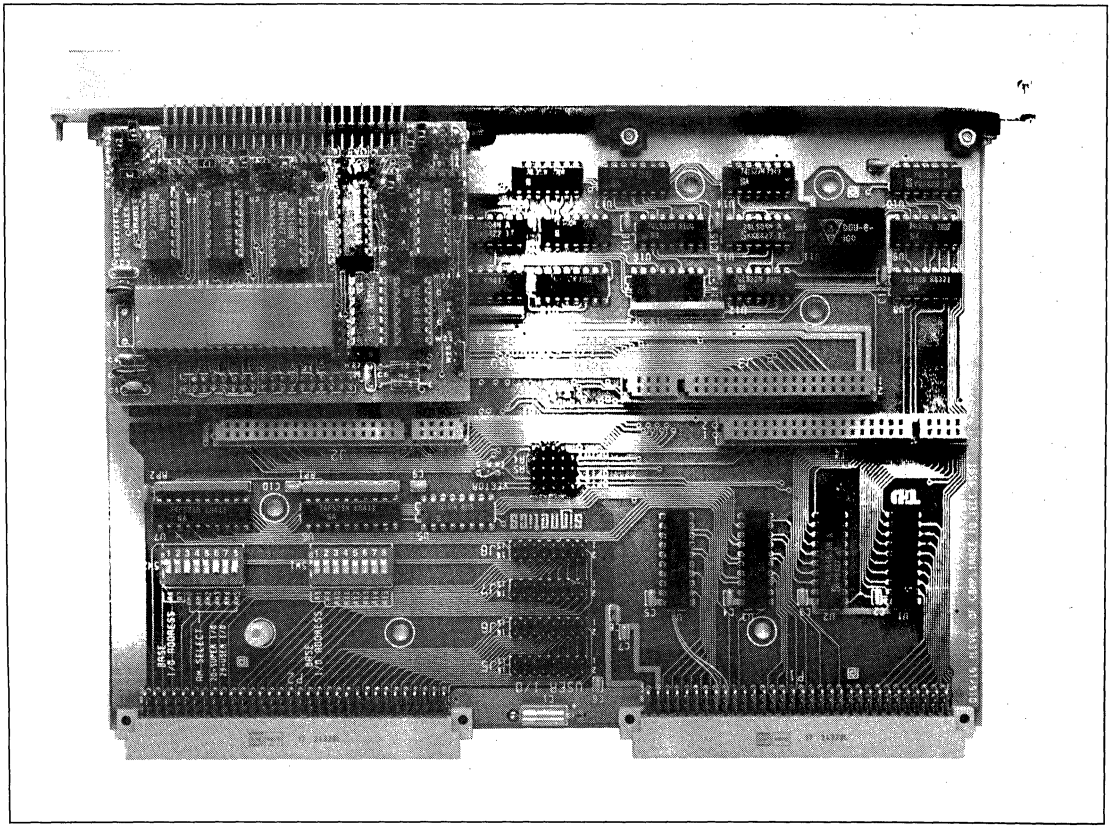
Microsystems Products

DESCRIPTION

The SMVME1600 Quad I/O Module is a part of the Signetics Modular Board product family. It is designed to allow the user to interface up to four IEEE-P959 input/output modules to VMEbus based systems or sub-systems. User system design is enhanced for specific applications by selection of appropriate I/O modules capable of 8- or 16-bit data transfers to and from the VMEbus. Base I/O address and address modifier selections are done by switch setting. Interrupt vector source selection is also jumper selectable. MWAIT signal with up to 300ns latency delay is also supported. VMEbus P2 connector can also be used for user I/O cables. BERR signal is asserted if an uninstalled module is accessed.

FEATURES

- **Connectors for up to four ISBX™ or BLX™ modules**
- **8- or 16-bit transfers**
- **Single or double wide I/O modules**
- **VME slave interface**
- **Memory-mapped I/O**
- **Supervisor and/or User modes**
- **64 bytes address space per module**
- **Programmable interrupt level or vector**



ISBX is a trademark of Intel Corporation.
BLX is a trademark of National Somiconductor Corporation.

VMEbus Quad I/O Module SMVME1600

MODULE INTERFACE
Up to four IEEE-P959 (ISBX or BLX) I/O modules can transfer 8 or 16 bits of data to and from VMEbus.

ADDRESS SELECTION
The base I/O address is switch selectable. Address modifier code is also switch selectable for User, Supervisor and User/Supervisor modes.

INTERRUPTS
The interrupt vector for each of the four IEEE-P959 modules is software programmable and is presented during the interrupt acknowledge cycle on the VMEbus. For unused modules the interrupt may be disabled. Interrupt level IREQ1 through IREQ7 is dynamically configurable.

MWAIT LATENCY
Up to 300ns delay in the assertion of MWAIT signal before data transfer from IEEE-P959 modules.

I/O CONNECTORS
User I/O pins can be connected on the VMEbus P2 connector for modules 2 and 4. Module 1 and 3 connections are from the front of the VME1600 board.

POSSIBLE APPLICATIONS
- Single/dual channel RS-232C interface
- Centronics parallel printer interface
- ISBX series modules
- BLX series modules
- Real-time clock
- IEEE-488 Listener/Talker and controller
- Display modules
- A/D, D/A converter

SPECIFICATIONS

CONFIGURATION	DTB Slave A16, D16, DYN
USAGE	One or more per VMEbus card cage
DIMENSIONS	Width: 233.4mm (9.19″) Height: 160.0mm (6.3″)
POWER	+5VDC A Max ±12VDC *
OPERATING TEMPERATURE RANGE	0°C to 55°C
HUMIDITY	10% to 90%, non-condensing

***NOTE:** Provided for User Modules.

ORDERING INFORMATION

SMVME1600	VMEbus Quad I/O Module
SMVME1610	Serial Interface Module
SMVME1620	Parallel Interface Module
SMMAN1600	VMEbus Quad I/O Module User Manual

***NOTE:** Provided for User Modules.

4

Signetics

SMVME1610
VMEbus Serial Module

Product Specification (Brief)

Microsystems Products

DESCRIPTION

SMVME1610 provides two independent full-duplex asynchronous receiver/transmitter channels on a single module in IEEE959 form factor. Programmable baud rates up to 38,400 baud are available from SCN68681 resident baud rate generator.

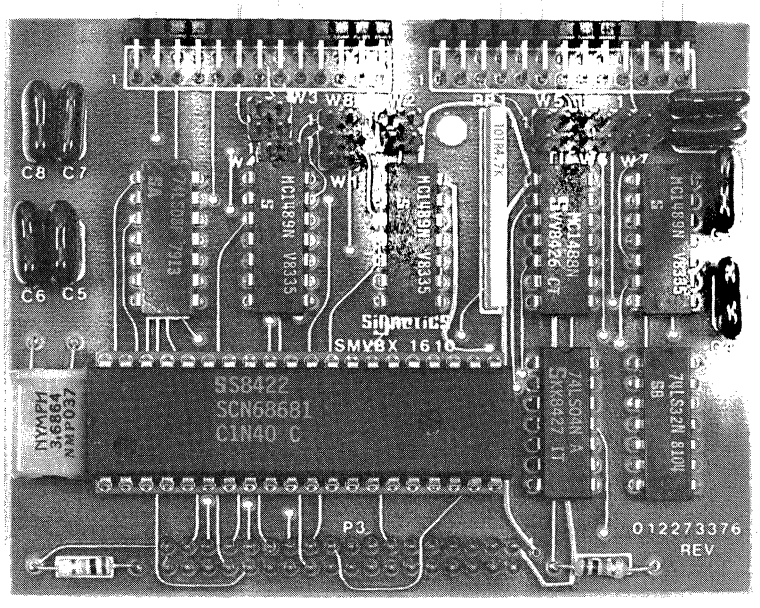
The operating mode and data format of each channel can be programmed independently. Each transmitter and receiver can select one of eighteen fixed baud rates, 16x clock derived from a programmable counter timer, or external 1x or 16x clock. Baud rate generator can operate directly from a crystal or external clock.

Data format is programmable and includes 5 to 8 data bit length plus parity (odd, even, forced parity, and no parity) with 1, 1.5, or 2 stop bits. False start bit detection, parity, framing, overrun, and line break detection are also provided by SCN68681. Automatic wake up mode, automatic echo, local or remote loopback channel features are also available.

The module is designed to operate in conjunction with SMVME1600SD VMEbus compatible I/O module. Each serial communication channel utilizes RS232-C compatible signal level drivers and receivers.

FEATURES

- **IEEE959 compatible**
- **Dual Asynchronous Receiver/ Transmitter SCN68681**
- **Two RS232-C Level Serial Channels**
- **Operates from +5VDC, ± 12VDC voltage sources**
- **Programmable baud rates up to 38,400**
- **Compatible with SMVME1600SD**



VMEbus Serial Module SMVME1610

SPECIFICATIONS

CONFIGURATION

DTB Slave using address lines A01 through A04 of VMEbus, D8 I(1).
Provides MWAIT* signal upon asserting MCS0* until DTACK* is generated by the SCN68681.
Interrupt is generated through MINT0.

MECHANICAL

Standard Single-Wide IEEE-P959 expansion module

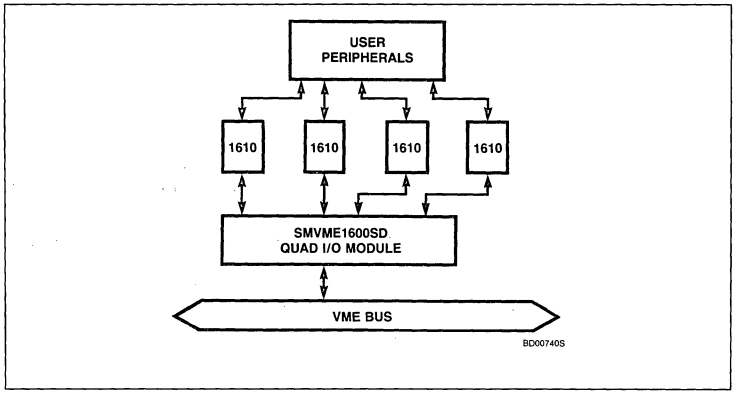
Width: 3.70 inch Depth: 2.85 inch

ELECTRICAL

	Current Typ.	Max.
+5.0VDC	170mA	250mA
+12.0VDC	19mA	25mA
-12.0VDC	-18mA	-23mA

ENVIRONMENTAL

Recommended temperature:
Operating: 0°C to 55°C ambient max.
Nonoperating: -40°C to 85°C ambient max.



ORDERING INFORMATION

SMVME1610	VMEbus Serial Module
SMMAN1610	Serial Module User Manual
SMVME1600	VMEbus Quad I/O Module
SMMAN1600	VMEbus Quad I/O Module User Manual

4

Signetics

SMVME1620
VMEbus Parallel Module

Product Specification (Brief)

Microsystems Products

DESCRIPTION

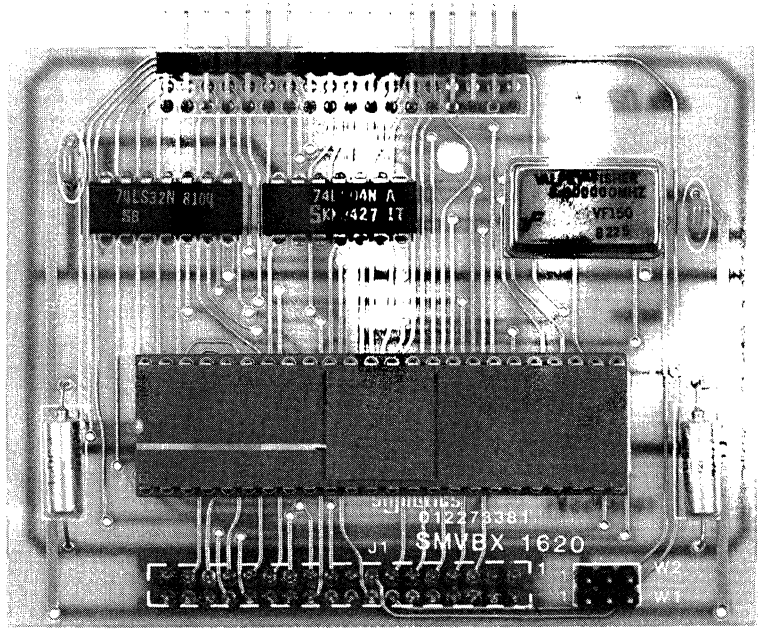
SMVME1620 provides double buffered parallel interface ports and a SCN68230 resident timer, in IEEE959 module. Bi-directional 8-bit or 16-bit I/O ports along with 24-bit counter and 5-bit prescaler are also available.

In unidirection mode, SCN68230 resident data direction register determines which pins are inputs or outputs. However, in bidirectional mode data direction register is ignored and four handshake pins dynamically determine data direction. The handshake pins are programmable and provide flexibility to interface a wide range of peripherals. The SCN68230 ports allow use of vectored or autovectored interrupts. A Direct Memory Access (DMA) pin is also available. The timer can be clocked by system or external clock. The counter may be used for time ticks or elapsed time measurements through periodic interrupt generation.

SMVME1620 is designed to operate in conjunction with SMVME1600SD VMEbus compatible module.

FEATURES

- **IEEE959 form factor**
- **Parallel interface/timer VLSI SCN68230**
- **Dual 8-bit or one 16-bit bidirectional port**
- **24-bit programmable timer**
- **Operates from +5VDC voltage source**
- **Compatible with SMVME1600SD**



VMEbus Parallel Module

SMVME1620

SPECIFICATIONS

CONFIGURATION

DTB Slave using address lines A01 through A05 of VMEbus, D8 I(1).
Provides MWAIT* signal upon asserting MCS0* until DTACK* is generated by the SCN68230.
Interrupt is generated through MINT0 for the parallel port and on MINT1 for the timer.

MECHANICAL

Standard Single-Wide IEEE-P959 expansion module

Width: 3.70 inch
Depth: 2.85 inch

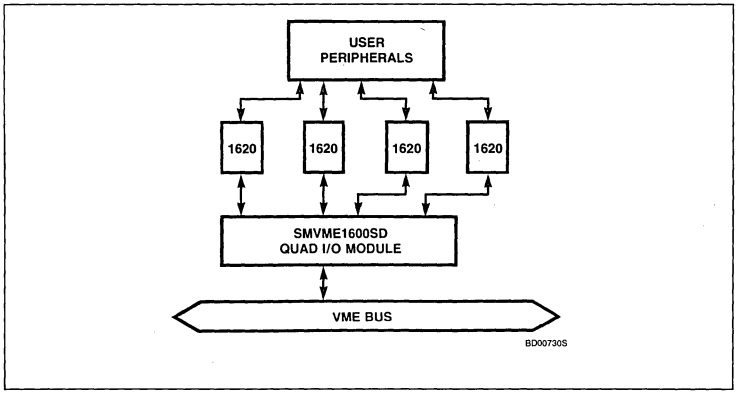
ELECTRICAL

	Current	
	Typ.	Max.
+5.0VDC	150mA	200mA

ENVIRONMENTAL

Recommended temperature:

Operating:	0°C to 55°C ambient max.
Nonoperating:	−40°C to 85°C ambient max.



ORDERING INFORMATION

SMVME1620	VMEbus Parallel Module
SMMAN1620	Parallel Module User Manual
SMVME1600	VMEbus Quad I/O Module
SMMAN1600	VMEbus Quad I/O Module User Manual

4

Signetics

SMVME2000
VMEbus CPU Module

Product Specification (Brief)

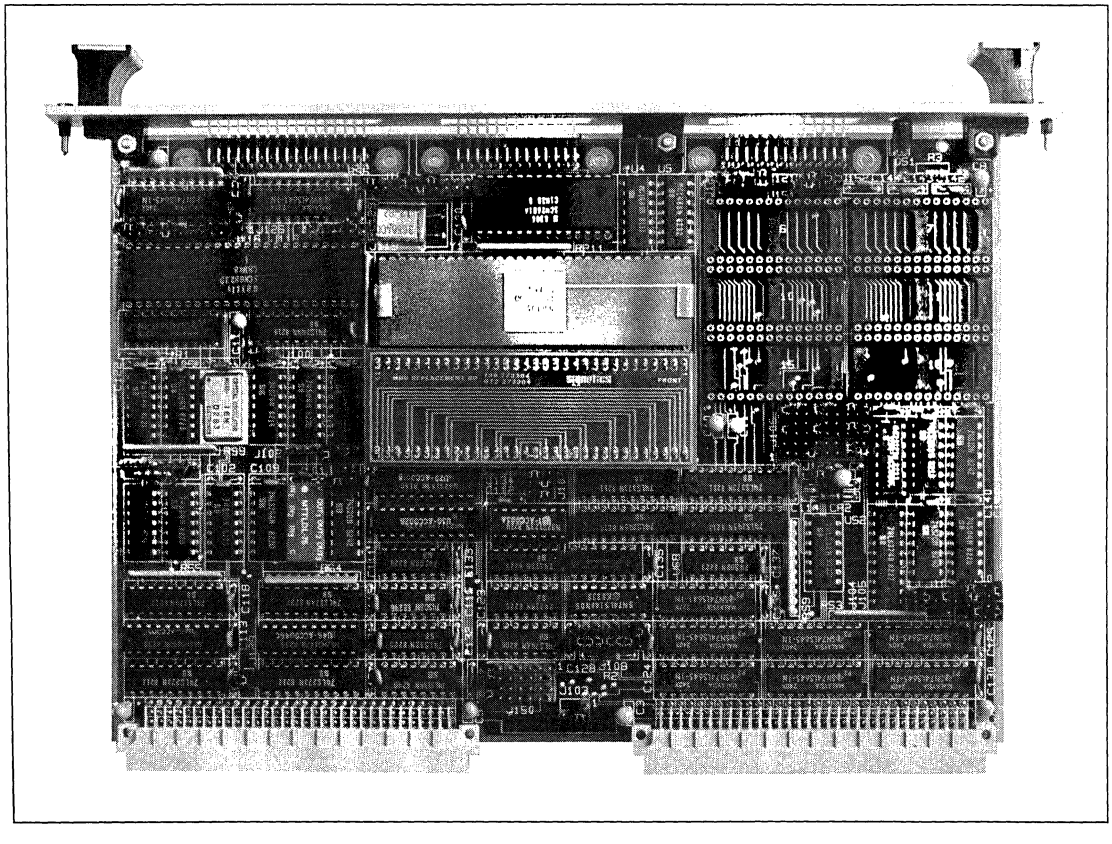
Microsystems Products

DESCRIPTION

The VMEbus Central Processor Module is one of the high-performance building blocks in the Signetics Modular Board product family. It embodies, in a single module, an SCN68000 based computing function capable of supporting high-speed data processing and control environments. This module is ideally suited for a wide range of applications. As a ROM based, single board computer, it is usable in a standalone application requiring no card cage or supporting modules. This provides the user with a compact computing nucleus for dedicated applications. The flexibility extends upward to include memory management for use in more general purpose, disk-based applications requiring multiprogrammed/multitasking solutions. All local I/O devices employ interrupt protocols for programming ease and fast response. This module is designed utilizing multilayer printed circuit techniques with internal power and ground planes in order to minimize noise and provide a reliable, industrial-grade product. The Central Processor Module is easily configured to support a wide variety of applications.

FEATURES

- **8MHz SCN68000**
- **Optional memory management SCN68451**
- **On-board memory: up to 48KB ROM PROM EPROM 8KB RAM**
- **Peripheral I/O ports: 2 RS-232 SCN2681/1-parallel (SCN68230)**
- **Programmable real time clock: 1 24-bit timer**
- **Seven levels of interrupt**
- **VMEbus compatible**



VMEbus CPU Module — SMVME2000

OPTIONAL MEMORY MANAGEMENT

Memory address translation and access protection are provided by the SCN68451 Memory Management Unit (MMU) for the entire addressing range of the SCN68000 CPU. The 16-megabyte address range is partitioned into 32 segments, where each segment can range from 256 bytes to 16 megabytes. The MMU can be bypassed for applications where a single, continuous address space is more appropriate.

CPU

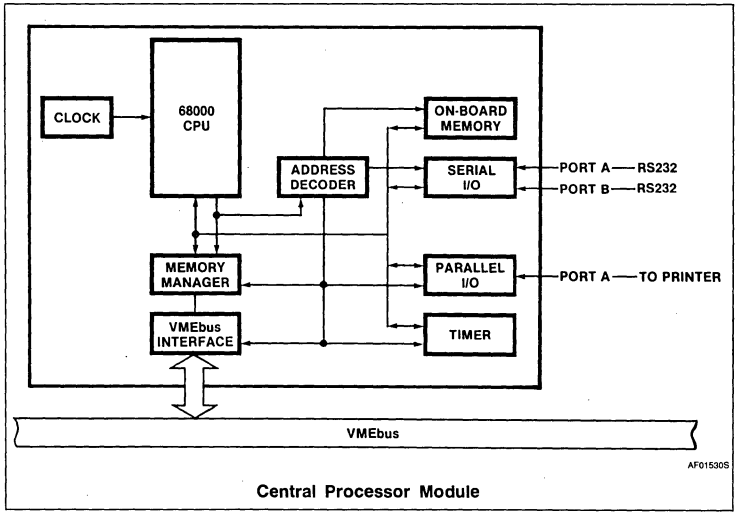
The Processor Module is available with an SCN68000 8MHz CPU. The CPU is configured to fully support up to 1,537 individual interrupting devices. This configuration will support the majority of the applications requiring high-performance processing.

ON-BOARD MEMORY

The On-Board Memory is implemented as a local resource to the SCN68000 CPU and, therefore, is protected from other VMEbus modules. Three pairs of sockets are provided. Each pair of sockets are 28-pin JEDEC standard and have sufficient address and data lines for 16K words maximum. VMEbus battery backup is selectable for each socket pair to support CMOS RAMs and the user can select the memory timing to appropriately match the memory devices used.

ADDRESS DECODER

A decoder PROM selects the on-board resources (Memory, Serial I/O, Parallel I/O, Timer). The PROM is socketed to allow the user the option of repositioning these devices in the address space.



Central Processor Module

I/O PORTS AND TIMER

Two serial ports are provided and supported by RS-232 drivers and receivers. A Signetics SCN2681 DUART is employed. This device provides two independent, full duplex, asynchronous receiver/transmitter channels which are individually programmable for operating mode, data format, and baud rates from 50 to 38.4K.

A parallel port has been provided using an SCN68230 PI/T which is a general purpose, programmable parallel interface device. This interface is user configurable to select up to 16 bidirectional lines under local or external control. This port defaults to support a Centronics printer interface. The SCN68230 also provides the 24-bit programmable timer function.

The timer is program selectable in numerous modes to provide flexible support as a real-time clock and all I/O ports are implemented as local resources.

SYSTEM INTERRUPTS

The VMEbus CPU Module can respond to any or all of the seven priority levels of incoming interrupts from the VMEbus and on-board devices.

ORDERING INFORMATION

SMVME2000	VMEbus CPU Module
SMVME2010	VMEbus CPU Module with memory management unit
SMMAN3020	VMEbus CPU Module User Manual

SPECIFICATIONS

CONFIGURATION	DTB Master: A24, D16
USAGE	One (standalone); One or more per VMEbus card cage
DIMENSIONS	160mm x 233.4mm
POWER	2.5A typ (3.0A max) @ +5VDC
	40mA typ (50mA max) @ +12VDC
	20mA typ (30mA max) @ −12VDC
TEMPERATURE	0°C to 55°C operating
HUMIDITY	8% to 80% non-condensing
CPU	SCN68000 8MHz
OPTIONS (static)	DTB REQUESTER: any one of levels 1 through 4
	INT REQUESTER: any one of levels 1 through 7
	INT HANDLER: all levels 1 through 7
	MEMORY: RAM/ROM, type, speed, standby

4

Signetics

SMVME21XX Series
VMEbus Distributed Multiprocessing Engine

Product Specification (Brief)

Microsystems Products

DESCRIPTION

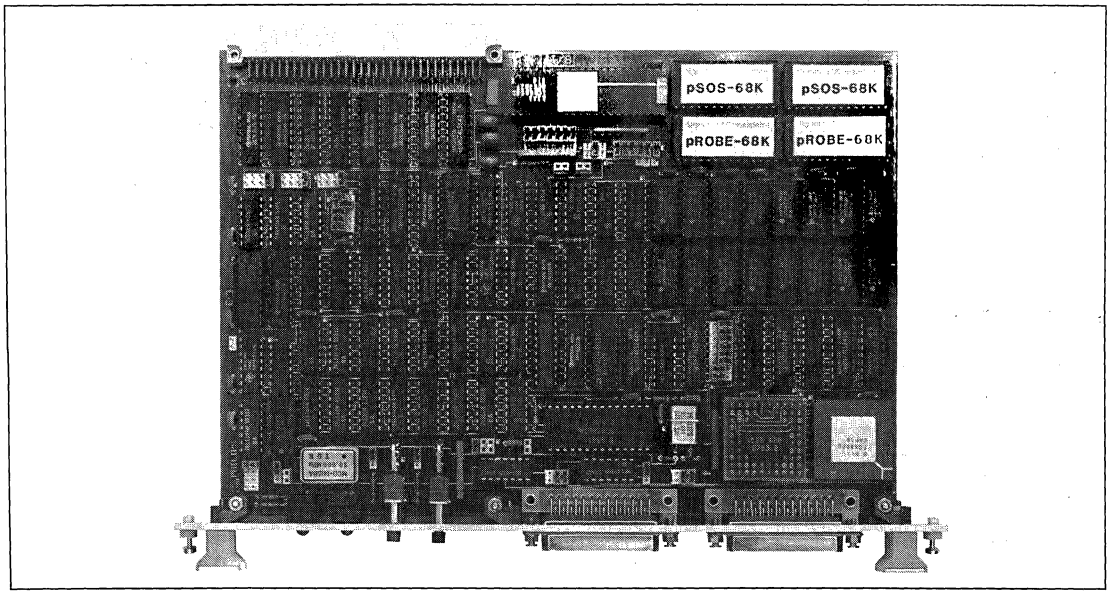
The SMVME21XX series of VMEbus Central Processor Modules are part of the high-performance building blocks in the Signetics Modular Board product family. The series is also the first of continuing generations of distributed multiprocessing engines. The modules form a complete computing nucleus consisting of SCN68000 MPU, SCN68451 MMU, and instruction and data memory, as well as peripheral I/O and real-time clock. The hierarchical buses architecture implemented within the modules confines execution traffic to the local on-board bus. This relieves the VME system bus of instruction and data manipulation traffic, thereby allowing multiple application processors, not just intelligent I/O processors, to run in parallel. The SMVME21XX series modules also provide some additional key features to support this class of multiprocessing. A VMEbus interrupt requester with dynamic IRQ signal line selector and vector allows application tasks residing in different processors to communicate with each other signaling events and exchanging messages. System interrupt handling can be localized or fully distributed between the application processors. Dual ported, master/slave operation of the modules allows messages, new instructions, and data to be directly written into the local memory without any intermediate buffering within global memory. A processor I.D./Flag register is provided to allow distributed operating systems to identify the unit number of the processor during system initialization.

The SMVME21XX series of modules also provide many benefits for the more classical systems architectures as well. The large on-board memory provides a low latency path to memory allowing the user to extract the full performance of the SCN68000 MPU. The series not only provides lower total system costs by combining two boards into a single board, but also provides savings in VMEbus slots.

The modules are reliable, industrial-grade products using multilayer printed circuit boards with internal power and ground planes to minimize noise.

FEATURES

- **Complete, self-contained computing nucleus**
- **Master/slave VMEbus operation**
- **Dual ported, on-board memory: 128KB DRAM**
- **SCN68000 MPU at 8MHz or 10MHz**
- **Optional SCN68451 for memory management or protection**
- **On-board VMEbus interrupt requester to signal events and messages between processors**
- **Process I.D./Flag register for distributed operating systems**
- **Four on-board 28-pin ROM/EPROM sockets for up to 128KB of operating system kernels**
- **Two asynchronous RS-232 communication ports: SCN2681**
- **24-bit real-time clock**



VMEbus Distributed Multiprocessing Engine — SMVME21XX Series

THE MPU

The processor module comes with an SCN68000 8MHz or 10MHz MPU. The module is configured to fully support all interrupt levels and vectors, which makes it ideal for the majority of high-performance processing applications.

OPTIONAL MEMORY MANAGEMENT

An optional SCN68451 Memory Management Unit (MMU) provides address translation and access protection for the entire address range of the SCN68000 MPU.

The MMU contains 32-segment descriptors which can be used to partition the 16MB address range into segments containing from 256 bytes to 16MB. The MMU can be bypassed for applications where a single continuous address space is more appropriate.

I/O PORTS AND TIMER

The module has two serial ports supported by RS-232 drivers and receivers. An SCN2681 DUART provides two independent, full duplex, asynchronous receiver/transmitter channels, individually programmable for operating mode, data format, and baud rates from 50 to 38,400 baud.

An SCN68230 PI/T provides the 24-bit programmable timer function. The timer is program selectable in numerous modes to serve as a real-time clock.

ON-BOARD MEMORY

The 128KB dynamic RAM is implemented as a dual-accessed resource. It can be accessed by the on-board MPU and via the VMEbus. A jumper-selectable base address, for access from the VMEbus, is provided. The memory array is designed to accept 256K DRAM providing 512KB of on-board DRAM for future memory expansion.

Four JEDEC sockets provide up to 128KB of EPROM. The user can select the access time appropriate to the EPROM used.

PROCESSOR I.D./FLAG REGISTER

The processor I.D./Flag is an 8-bit, jumper-selectable input port that can be read by the processor. During the initialization phase of a multiprocessor system, this information can be used to determine the unit number of the processor. The register can also be utilized for other purposes such as auto configuring software.

VMEBUS INTERFACE

Data Transfer Bus

The modules provide a VMEbus master/slave controller. On-board circuitry resolves deadlock situations that may occur. An A24, D16 master is capable of generating both A24 and A16 supervisor and non-privileged address modifiers.

Arbitration Bus

The bus requester operating is jumper selectable for release on request or release when done operating, as well as being operable on any of the four levels.

Interrupt Bus

The module series can respond to any or all seven VMEbus interrupt request lines, Interrupt Handler 1 – 7 static. An Interrupt Requester 1 – 7 dynamic is also provided whereby software can select both the IRQ signal line and the vector.

EXCEPTION HANDLING

All on-board interrupt sources are prioritized and vectored. AC FAIL and a front panel abort switch provide a level 7 auto vectored interrupt. Bus error can be generated from MMU faults, local bus time-out, or VMEbus errors. SYSFAIL circuitry is provided for both module diagnostics and broadcast to the VMEbus, as well as for reception of SYSFAIL from other failed devices on the bus. Module reset is provided by both SYSRESET and a local reset from a front panel switch.

SPECIFICATIONS

Configuration

DTB Master: A24, D16 ROR/RWD, NTSLT, RMW
DTB Slave: A24, D16 (DRAM), NBLT, RMW
IH 1 – 7 static
IR 1 – 7 dynamic

Usage

One (standalone), one or more per VMEbus

Dimensions

160 x 233.4mm^2

Power

+5VDC, 3.1A max
+12VDC, 50mA max
−12VDC, 50mA max

Operating Temperature

0°C to 55°C

Humidity

10% to 90% non-condensing

ORDERING INFORMATION

SMVME2100	8MHz 68000/128KB DRAM CPU Module
SMVME2101	10MHz 68000/128KB DRAM CPU Module
SMVME2110	8MHz 68000/128KB DRAM with MMU CPU Module
SMVME2111	10MHz 68000/128KB DRAM with MMU CPU Module
SMVME2120	8MHz 68000/512KB DRAM CPU Module
SMMAN21XX	User Manual

4

Signetics

SMVME31XX
VMEbus 256KB/1MB Memory Module

Product Specification (Brief)

Microsystems Products

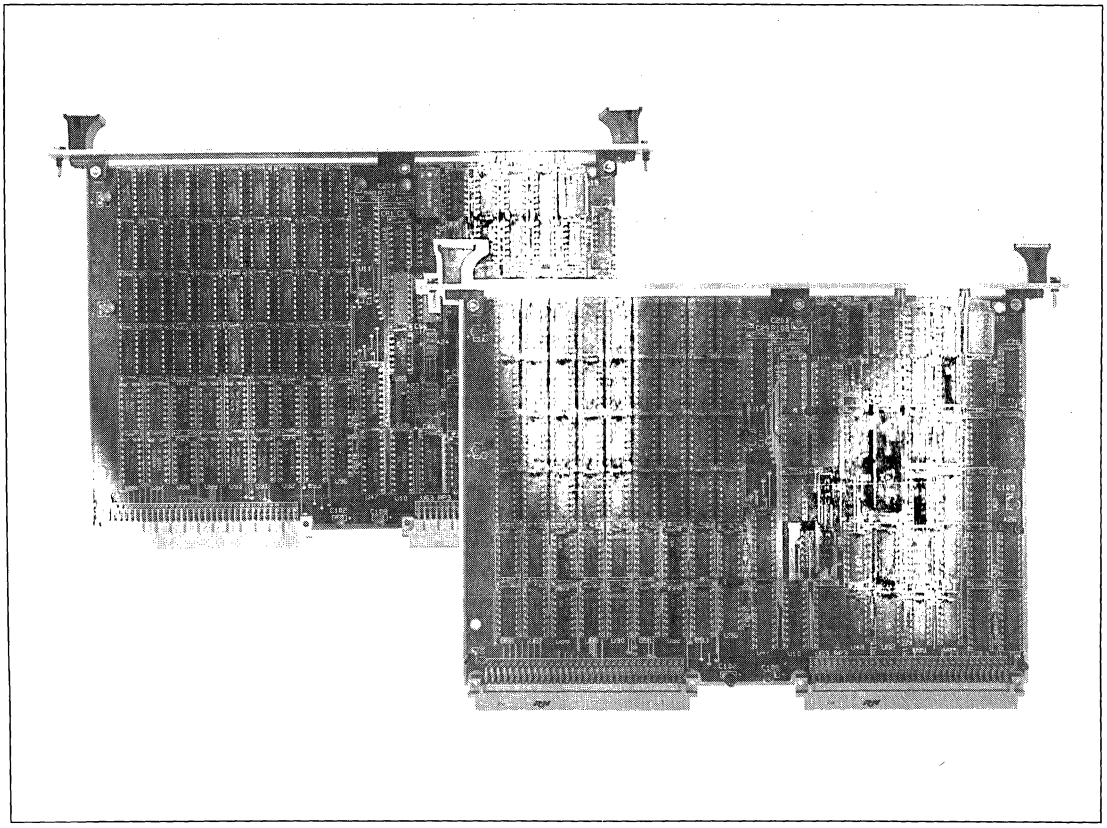
DESCRIPTION

The VMEbus Memory Module is one of the high-performance building blocks in the Signetics Modular Board product family. It provides, in a single module, 256KB/1MB bytes of dynamic random access memory plus parity protection. This module was specifically designed to support 8-bit, 16-bit, and 32-bit data transfers automatically. This feature allows total software flexibility in mixed CPU systems and negates the need to replace memory modules in future system upgrades.

The Memory Module employs modern multilayer, printed circuit techniques with internal power and ground planes in order to minimize noise and provide a reliable, industrial-grade product.

FEATURES

- **Large Capacity 256/1024KB plus parity**
- **Flexible data widths**
  - **Supports 8-, 16-, and 32-bit transfers**
- **Protection mechanisms**
  - **Selectable for supervisor access only**
- **Modern architecture**
  - **Supports 2-way interleaving**
- **VMEbus compatible**



VMEbus 256KB/1MB Memory Module SMVME31XX

MEMORY ARRAY AND PARITY

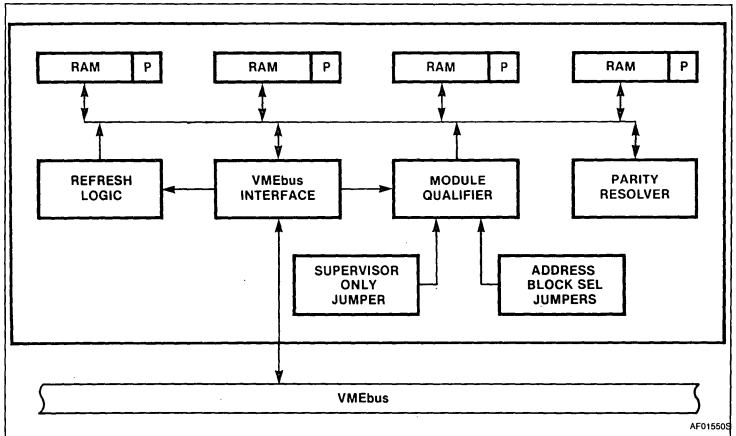
The 256KB/1024KB of dynamic RAM are partitioned into four 64KB/256KB arrays. Parity is generated or checked whenever an array is accessed on an array by array basis. In the D8 mode (byte transfer) only the appropriate array is accessed. In the D16 mode (word transfer) only the appropriate array pair is accessed. In the D32 mode (long word transfer) all arrays are accessed. In every case, the data byte from the accessed array is interfaced to its appropriate byte slot on the VMEbus data lines. Any parity error encountered will result in a bus error (BERR) being returned to the master.

PROTECTION MECHANISMS

In order to support protected software methodologies, the user may select the supervisor only mode of operation. In this mode the module can only be accessed when the VMEbus address modifier field indicates a supervisor is initiating the access. All non-supervisor accesses will result in a bus error (BERR) return. The Memory Module may alternately be selected to the supervisor-user mode of access. In this mode the Memory Module will respond to either a supervisor or a user initiated access.

INTERLEAVING

The user may select 2-way interleaving in either the D16 or D32 mode. The effect of interleaving is to decrease the effective memory cycle time. Interleaving can provide a 50% to 100% memory throughput improvement depending on the application. The interleave is accomplished by steering sequential memory accesses to pairs of Memory Modules on an alternating basis. This allows a master (processor or DMA device) access to one of the module pairs while the other module is completing the present cycle and performing refresh. This feature requires Memory Modules using this mode to exist in pairs. The Memory Module is selectable for either D16 or D32 operation which is included specifically for the interleaving mode. Other Memory Modules residing in the VMEbus card cage which do not have the interleaving mode enabled are not affected.



Memory Module

SPECIFICATIONS

CONFIGURATION	—	DTB Slave: A24, D8, D16, D32
USAGE	—	One or more per VMEbus card cage
DIMENSIONS	—	160mm x 233.4mm
POWER	—	3.1A typ (4.0A max) @ +5VDC
TEMPERATURE	—	0°C to 55°C operating
HUMIDITY	—	8% to 80% non-condensing
SPEED	—	195ns WRITE, 265ns READ, 340ns cycle
ERROR	—	BUS ERROR (BERR) on parity fail
OPTIONS (static)	—	ACCESS: SUPV only/SUPV-USER
	—	ADDRESS BLOCK: 256KB boundaries
	—	Interleave: non/2-way
	—	MODE: D16/D32

ADDRESS BLOCK

The Memory Module provides user selection on any 256KB/1024KB boundary in the 16MB address space.

REFRESH LOGIC

The memory arrays are refreshed in one of two modes. The primary mode is to perform a refresh immediately after a memory access cycle (often referred to as "hidden refresh"). In the backup mode, if no memory access occurs for 8 microseconds, the module then "forces" a refresh by self-initiating a refresh cycle.

ORDERING INFORMATION

SMVME3100	VMEbus 256KB Memory Module
SMVME3110	VMEbus 1MB Memory Module
SMMAN31XX	VMEbus 256KB/1MB Memory Module User Guide

4

Signetics

SMVME3300
VMEbus RAM, ROM and EPROM Module

Product Specification (Brief)

Microsystems Products

DESCRIPTION

The SMVME3300 Memory Module is one of the high-performance building blocks in Signetics' Modular Board product family, with a 512KB capacity. It provides two arrays of eight JEDEC sockets for RAM, ROM or EPROM. Since each array may contain a different type of memory, the access-time can be adjusted, in steps of 62.5ns, to that required for the memory in use. Further, an address field is available for each array and the address range can be adapted according to the device used.

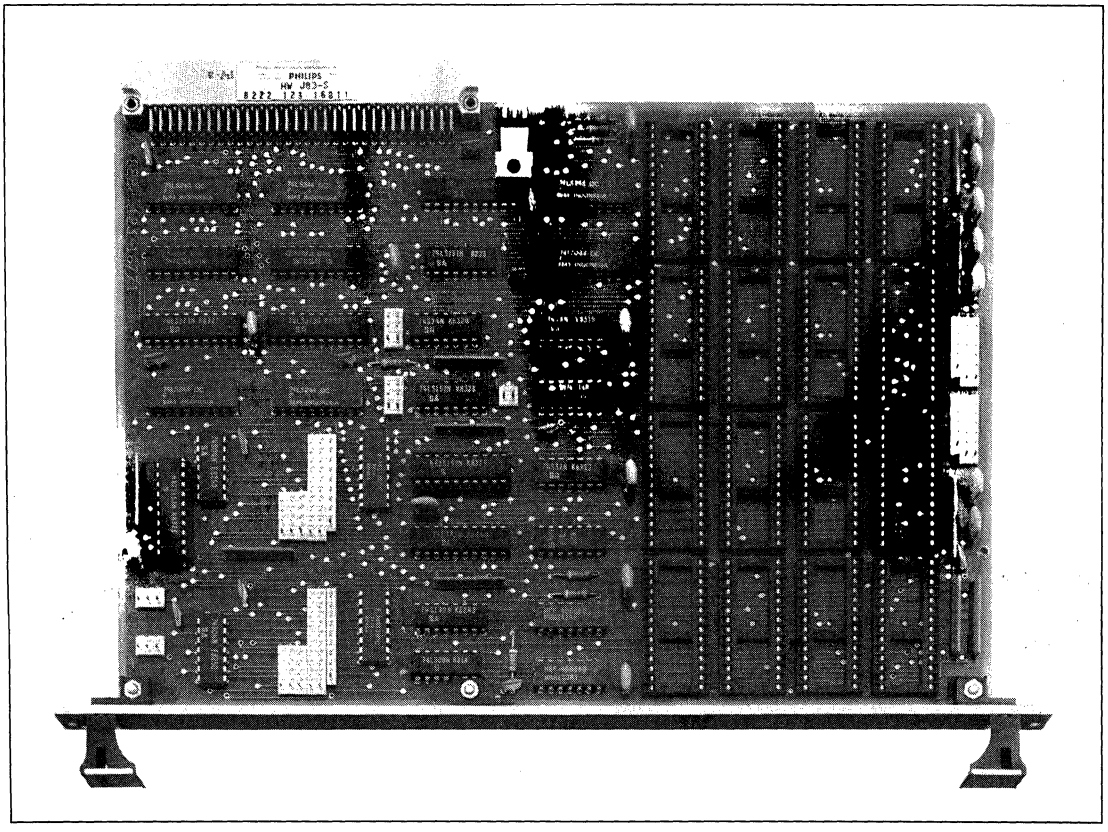
An external battery backup may be connected when using CMOS RAM. BERR is generated when an illegal access occurs.

The SMVME3300 recognizes the address modifier signals of the VMEbus. Address modifier codes are jumper selectable.

The SMVME3300 is a reliable industrial standard product using modern multilayer, printed circuit techniques with internal power and ground planes to minimize noise.

FEATURES

- **Sockets for RAM, ROM and EPROM**
- **Two arrays of eight JEDEC sockets**
- **Large capacity: up to 512KB**
- **DTACK adjustment (per array)**
- **Selectable write-protect function (per array)**
- **Suitable for many types of memory**
- **Battery backup capability for CMOS memory**
- **VMEbus compatible**



VMEbus RAM, ROM and EPROM Module SMVME3300

ADDRESS DECODER
The capacity of each memory can be from 2KB to 32KB; consequently, the address decoder has a corresponding address range for each socket array.

DTACK LOGIC
The DTACK timing of each array is variable. This allows the access time to be selected to suit the device. The DTACK timing is derived from the system clock of the system controller and can be adjusted in 62.5ns steps to a maximum of 875ns, via a jumper field.

ADDRESS MODIFIER
An FPLA decodes the address modifier signals of the VMEbus, giving the option to install a new set of modifiers. For each array, 8 modifiers can be selected via jumpers (seven selectable modifiers and one "don't care").

PROTECTION
To prevent software errors causing a 'write' to a read-only memory, the user may select a write-protect function. If such an error occurs, a bus error (BERR) instead of a DTACK signal will be sent. The BERR signal is also generated when the memory is accessed in D32 (long-word) mode.

BATTERY BACKUP
External battery backup can be provided for CMOS memories.

SOCKET CONFIGURATION
Any memory with JEDEC pinning can be used by inserting appropriate jumpers.

ORDERING INFORMATION

SMVME3300	RAM, ROM, EPROM Module
SMMAN3300	RAM, ROM, EPROM User Manual

SPECIFICATIONS

Configuration
DTB Slave: A24, D16

Usage
One or more per VMEbus card cage

Dimensions
160 x 233.4mm^2

Power
+5VDC, 1.5A max (with empty memory sockets)

Operating Temperature Range
0°C to 55°C

Humidity
10% to 90%, non-condensing

Speed
Device dependent
Access time: 875ns (max), in steps of 62.5ns

Error
Bus error (BERR) on write-protect
Violation and long-word access

Battery Backup
External via VMEbus

Address Modifier
Enable/disable
Seven selectable modifiers
Programmable via FPLA

Address Decoder
Variable memory boundaries
Selectable start address

4

Signetics

SMVME4300
VMEbus Disk Controller Module

Product Specification (Brief)

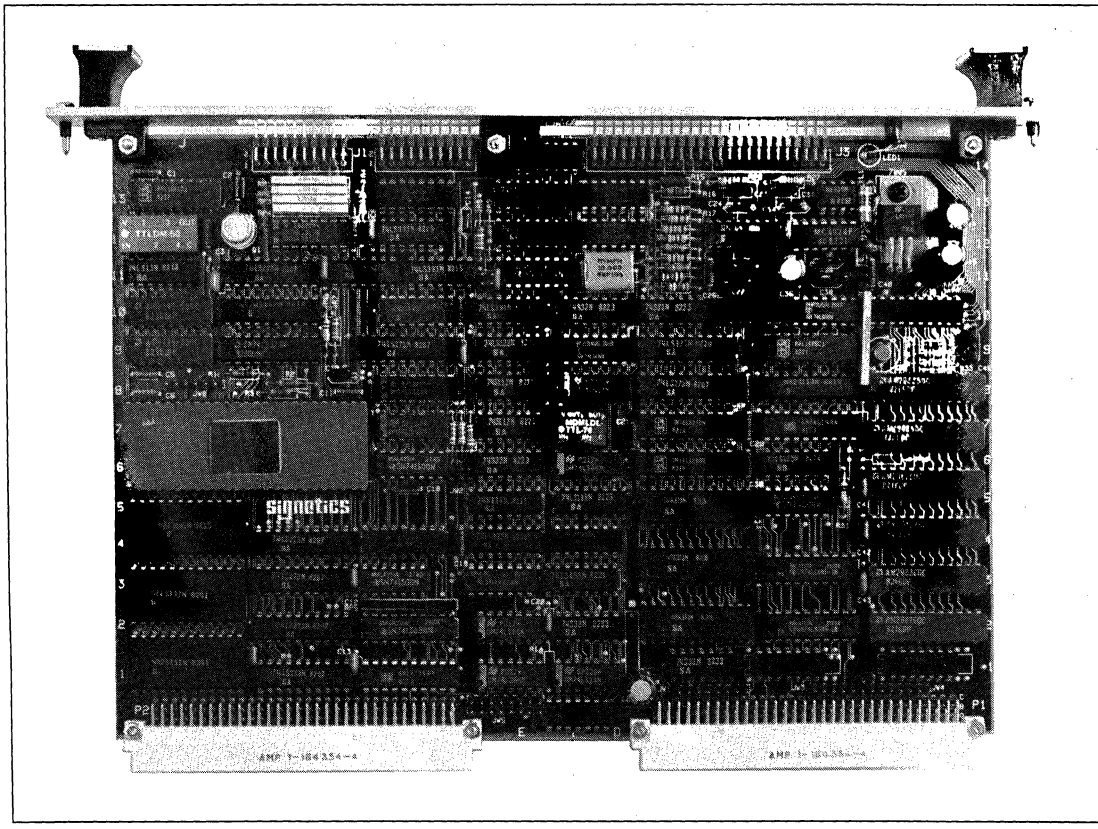
Microsystems Products

DESCRIPTION

The VMEbus Disk Controller Module is another addition to the high-performance building blocks in the Signetics Modular Board product family. It contains in a single module, a high-speed microprocessor supervised disk controller which interfaces to the industry's most popular hard and floppy disk drives. The Disk Controller Module provides a high-performance data channel between multiple disk drives and the VMEbus system main memory. Intended as an intelligent peripheral interface, the Disk Controller includes a straightforward, high-level interface to the system software by utilizing device control block (DCB) and direct memory access (DMA) protocols. Designed utilizing multilayer printed circuit techniques with internal power and ground planes to minimize noise, the Signetics Disk Controller Module provides a reliable, industrial-grade product.

FEATURES

- **Mixed media mass storage control**
  - **Supports up to four 5¼ in. and 8 in. Winchester and floppy disk drives per module**
- **Bus master DMA between main memory and disk drives controlled by an on-board Signetics 8X305 bipolar microprocessor**
- **Flexible formats**
  - **IBM standard or user defined**
- **Error recovery mechanisms**
  - **Error correction and detection built in**
- **VMEbus compatible**



VMEbus Disk Controller Module — SMVME4300

MICROCONTROLLER

Intelligence for the Disk Controller Module is provided by a Signetics 8X305 low-power Schottky microcontroller operating at 200ns per instruction. The 8X305 and associated support devices provide all data handling, DMA, disk drive, and error correction and control functions.

DISK DRIVE INTERFACE

Up to four disk drives in mixed configuration are supported. Of the four drives, up to two drives can be Winchester type 5¼ in. or 8 in. hard disks. Floppy drives, 5¼ in. or 8 in. single or double density, single or double sided are supported in any mix but hard drives can only be mixed if they provide the same data transfer rate.

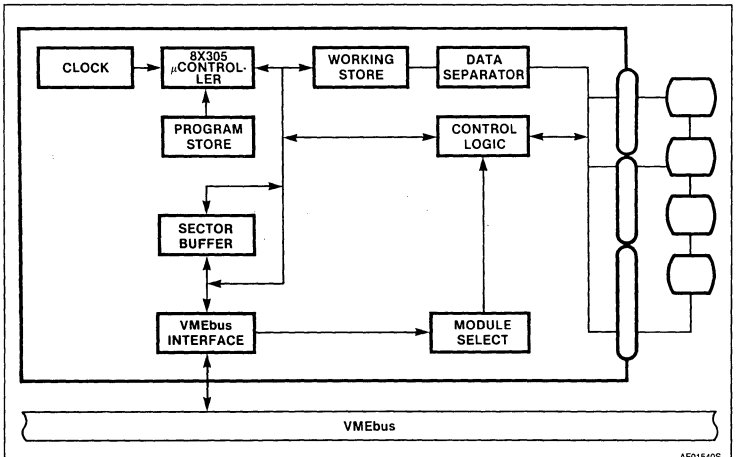
All head positioning (up to 1024 cylinders/drive) and head selection (up to 16 heads/drive) are automatically provided by the Disk Controller in response to a high-level command from the VMEbus system software. Serial data rates up to 8Mb/sec are supported.

DATA INTERFACE

Data is automatically transferred between any of the four disk drives and main memory for read and write operations. The read/write buffers may be located anywhere in the 16 megabyte main memory address space. Transfers can occur in the word (16-bit) or byte (8-bit) mode with odd byte boundary discontinuities automatically compensated for. Data is transferred directly for floppy type disks and buffered for hard disks in a 1024 byte on-board buffer. Multiple sector read/write is supported with automatic head switching and cylinder positioning as necessary.

ERROR DETECTION CORRECTION

The user may choose either CRC-16 or ECC-32 to provide data integrity validation. For CRC-16 a data error will be flagged and status returned to the system software. For ECC-32, burst errors of up to 11 bits/sector are automatically corrected and indicated in the return status at the completion of the operation.



DISK CONTROLLER MODULE

SPECIFICATIONS

CONFIGURATION	– DTB Master: A24, D16 DTB Slave: A24, D8
USAGE	– One or more per VMEbus card cage
DIMENSIONS	– 160mm x 233.4mm
POWER	– 2.6A typ (3.0A max) @ +5VDC – 20mA typ (30mA max) @ +12VDC – 20mA typ (30mA max) @ –12VDC
TEMPERATURE	– 0°C to 55°C operating
HUMIDITY	– 8% to 80% non-condensing
MICRO-CONTROLLER	– 8X305 5MHz
DRIVE INTERFACE	– Floppy Disk: Shugart 800, 850, 400, 450, 460 – Hard Disk: Shugart 1000, 600 Seagate ST506/512 Quantum Q 2000
SERIAL DATA RATE	– 8Mb/sec (max)
OPTIONS (static)	– DTB requester any one of levels 1 through 4 INT requester any one of levels 1 through 7 Module address: 2KB boundaries

COMMAND SET

Read sector(s) with overlapped seek
Write sector(s) with overlapped seek
Write deleted data
Format a track
Read a track
Restore and Recalibrate (to track 0)
Read record ID
Read sector (transparently)

ORDERING INFORMATION

SMVME4300	VMEbus Disk Controller Module
SMMAN3040	VMEbus Disk Controller Module User Manual

4

Signetics

SMVME5100/5101
Asynchronous Communication Module

Product Specification (Brief)

Microsystems Products

DESCRIPTION

The SMVME5100 4-channel Asynchronous Communication Module is one of the high-performance building blocks in the Signetics Modular Board product family. It provides four universal data communication lines capable of handling a variety of asynchronous communication protocols.

The board contains two SCN68681 DUARTs to handle four asynchronous data communication lines. These DUARTs are software programmable for full or half duplex data transfers, transmission speed, character length and parity.

Each line is RS-232 compatible. The SMVME5100 provides an additional isolated RS-422 interface for use in multidrop lines or for use in industrial environments.

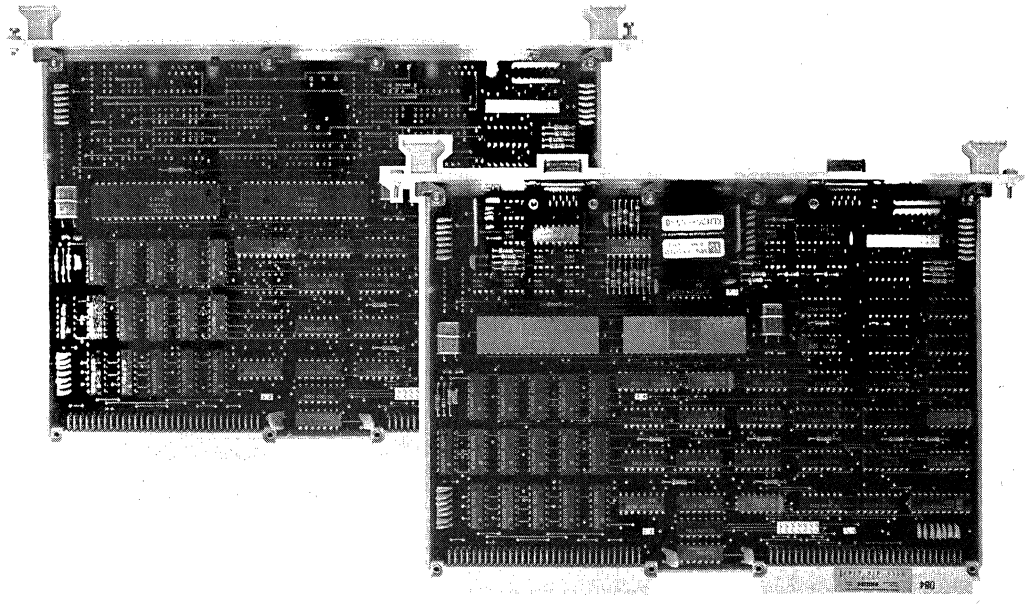
A full modem interface is provided to connect any type of asynchronous modem. Interface circuits are available to communicate with auto-answering modems.

The I/O devices employ interrupt protocols for programming ease and fast response.

This module is designed utilizing multilayer printed circuit techniques with internal power and ground planes in order to minimize noise and provide a reliable, industrial-grade product.

FEATURES

- **Four independent full duplex channels of asynchronous communications using SCN68681**
- **RS-232/V.24 with full modem interface and isolated RS-422/V.11 multidrop configuration**
- **Programmable by channel**
  - **Baud rates 50 to 38.4K baud**
  - **Data formats**
  - **Channel mode**
- **Two general purpose 16-bit timers**
- **Quadruple buffered receiver data registers**
- **VMEbus compatible**



Asynchronous Communication Module — SMVME5100/5101

VMEbus INTERFACE

DATA TRANSFER BUS
The SMVME5100/5101 is an A16, D8 DTB slave. The board can be located on any 128-byte boundary within the 64KB VMEbus Short Address Space.

PRIVILEGED ACCESS
The SMVME5100/5101 can be placed in either the Supervisor Only Short Address Space, or the Non-Privileged Short Address Space. Selection is static via jumpers.

INTERRUPT REQUESTER
A single interrupt requester may be connected to any one of seven VMEbus interrupt request lines IRQ1 – IRQ7 via jumpers. Two interrupt vectors, one per SCN68681, are provided which are activated upon the occurrence of any one of eight events per DUART. An interrupt mask register, interrupt status register and auxiliary control register are provided by each DUART. Interrupt capability upon change-of-state of the modem interface signals is also provided.

EXCEPTION HANDLING
Bus error is generated upon illegal accesses, D16 and D32. SYSRESET initializes the DUARTs. Uninitialized interrupt vector $OF is placed into the interrupt vector register. SYSFAIL disables RS-422 drivers on the SMVME5101 only.

SERIAL PORTS
The serial ports are implemented using two SCN68681 DUARTs.

GENERAL
Each DUART provides a dual full duplex asynchronous receiver/transmitter. Each receiver is quadruple buffered. Automatic wake-up mode is provided for multidrop applications. Line-break detection and generation are provided per channel. A multifunction programmable 16-bit timer/counter is also provided by each DUART.

PROGRAMMABLE DATA FORMATS
Five to eight data bits plus parity are software programmable. Odd, even, or no parity is selectable as well as 1, 1.5 or 2 stops bits.

PROGRAMMABLE BAUD RATE
Is selectable for each receiver and transmitter separately. Selection can be made from 18 fixed rates ranging from 50 to 38.4K baud or derived from the internal counter/timer.

PROGRAMMABLE CHANNEL MODES
Can be selected to be full duplex, half duplex, auto echo, or local and/or remote loopback. CCITT modem test loop circuitry and indicators are provided.

DATA COM ERRORS
Parity, framing, false start bit and overrun errors are detected and reported. Line-break detection and generation are also provided.

FULL MODEM INTERFACE
The SMVME5100/5101 provides all the necessary circuits to interface with most asynchronous modems as well as auto-answering types. In addition to the standard RS-232 Transmit and Receive Data Lines, Request to Send, Clear to Send, Data Set Ready, Data Carrier Detect, Connect Data Set to Line, Calling Indicator, Test Loops 2 and 3 and Test Indicator signals are provided per channel. Handshaking is performed automatically and input circuits can be programmed to generate interrupts.

SMVME5101
The SMVME5101 provides four RS-232 channels with all of the above features. These circuits are implemented on the designer defined pins of the J2/P2 VMEbus connector.

SMVME5100
The SMVME5100 provides four RS-232 channels with all of the above features plus the following features:

ISOLATED RS-422
Four channels of optically isolated RS-422 circuits are implemented on two 9-pin D-shell connectors on the front panel of the board. The design is suitable for use in multidrop applications or industrial environments.

INTERFACE TYPE SELECTION
Is performed by jumpers per channel.

TRANSMITTER DISABLING
Can be performed under software control by RTS, or SYSFAIL can force off all transmitters.

LOCAL COPY
In half duplex or multidrop connectors the receivers can be tied to their respective transmitters by jumper selection.

4

SPECIFICATIONS

CONFIGURATION	DTB Slave I/O, A15, D8 (odd byte)
USAGE	one or more per VMEbus
DIMENSIONS	160 x 233.4mm
POWER REQUIREMENTS	+5VDC: 2.1A typ., 3.0A max.
	+12VDC: 0.2A typ., 0.25A max.
	−12VDC: 0.2A typ., 0.25A max.
TEMPERATURE	0°C to 55°C operating
HUMIDITY	10% to 90% non-condensing
SPEED	1000nsec WRITE
	1100nsec READ
	1100nsec Interrupt
ERROR	Bus Error on long-word (LWORD = low, or DSO and DSI = low)
OPTIONS (status)	Access: SUPV + SUPV/USER
	Module address: 128 byte boundaries
	INT requester on any of the seven levels

ORDERING INFORMATION

SMVME5100 4-Channel Asynchronous RS-232/RS-422
SMVME5101 4-Channel Asynchronous RS-232
SMMAN510x 4-Channel Asynchronous Communication Module User Manual

Signetics

SMVME9100
VMEbus Evaluation Kit

Product Specification (Brief)

Microsystems Products

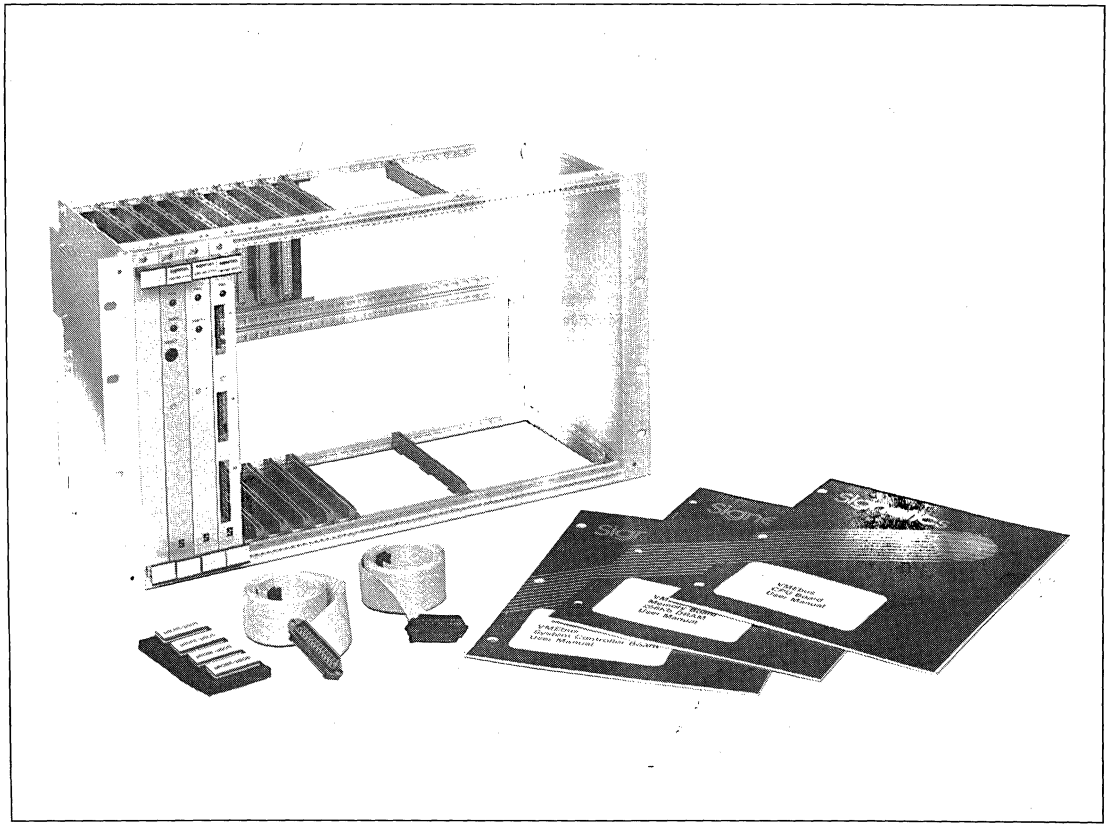
DESCRIPTION

The Signetics SMVME9100 Evaluation Kit is a fully integrated VMEbus computing nucleus. The SMVME9100 comes fully assembled and tested, saving you hours of system integration time trying to bring up your system. All boards are preconfigured to work together as a system that is capable of running the firmware provided. Only a power supply and power cable are needed to complete the system. All cabling and miscellaneous hardware are included when the optional SMVME0400 Power Supply is purchased, which eliminates the need to hunt through catalogs or wait on delivery for that one missing piece. Simply apply power, attach a CRT terminal, and you are up and running.

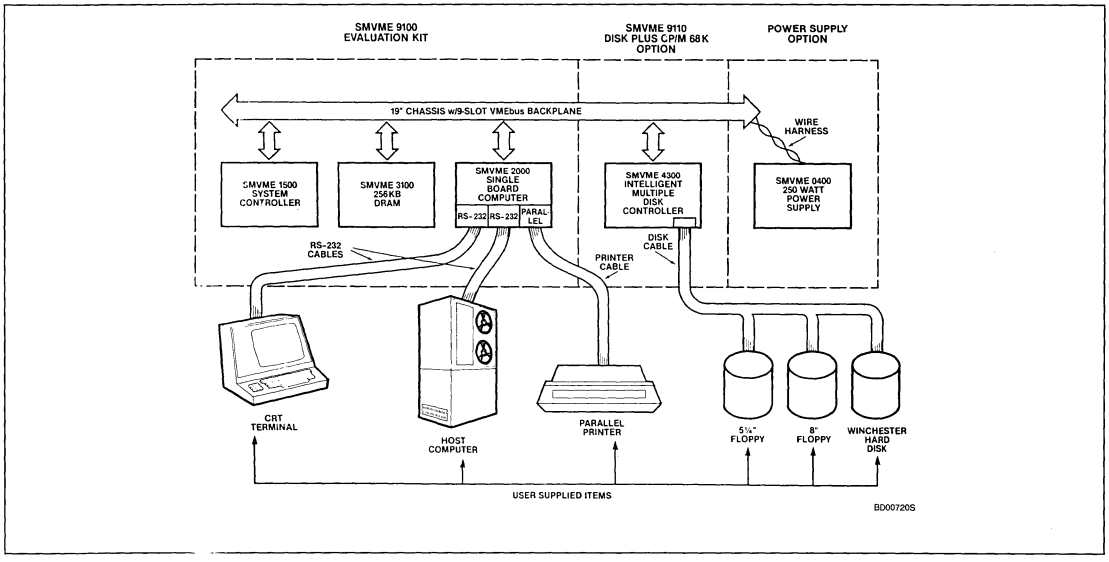
The SMVME9100 is ideal for low-cost evaluation of the VMEbus, Signetics VMEbus boards and Signetics software. If you need 68000 development system capability or wish to evaluate our intelligent multiple disk controller, then add the SMVME9110 option.

FEATURES

- **Low-cost Evaluation of the VMEbus and Signetics VMEbus products**
- **Everything you need to get a fast start on VME**
- **Fully integrated and ready to run hardware and software**
- **Expandable to form a low-cost resident development system with the addition of the SMVME9110**



VMEbus Evaluation Kit SMVME9100



THE HARDWARE

The hardware is shipped in a preconfigured format. There are no jumpers to set, no PROMs to burn, and only an optional power supply and power cable to buy. The following items are preassembled and included in the kit.

CPU BOARD

The powerful and flexible SMVME2000 single-board computer features the 8MHz SCN68000 MPU, SCN2681 DUART configured for two RS-232 channels, SCN68230 configured as a real-time clock, a Centronics parallel printer interface, and six 28-pin universal memory sockets. The board is preconfigured for the pROBE-68K Debug Monitor and pSOS-68K Real-Time Executive.

DYNAMIC RAM BOARD

The board provides 256KB of DRAM plus byte parity. It features dynamic 8-, 16- and 32-bit data and two-way interleaving. It is preconfigured for use with the pROBE-68K Debug Monitor and pSOS-68K; the memory access protection mechanism is preset for both supervisor and user access.

SYSTEM CONTROLLER BOARD

Provides VMEbus arbitration, power and restart circuitry, system reset function, system clocks, and a watchdog timer monitoring the data transfer bus transactions.

CARD CAGE

Full 19 in. rack-mount Eurocard chassis includes a 9-slot P1 backplane, preinstalled card guides for the VMEbus slots, and the optional power supply. Power connectors are included for attachment of user-supplied power supply.

CABLING AND HARDWARE

All necessary cables (except the power cable) and miscellaneous hardware are included. One RS-232 cable is designed to attach to a CRT terminal and the second is designed to attach to a foreign host for downloading of "S" records.

OPTIONAL POWER SUPPLY

The SMVME0400 Power Supply is an optional accessory that plugs into the SMVME9100. Featuring 30A at +5V, 3A at +12V and 1A at −12V, it also includes power fail circuitry, a power cable, and a wire harness to the SMVME9100's backplane.

SOFTWARE

The software provided has already been preconfigured to run on the above hardware and includes the pROBE-68K Debug Monitor and pSOS-68K Real-Time Executive.

pROBE-68K DEBUG MONITOR

An evaluation copy of this debugging tool, system monitor, and performance analysis tool is ROM-resident and features high-level debug commands, single-line assembler/disassembler, virtual terminal capability, breakpoints, memory/register, display/alter, program download and more.

pSOS-68K REAL-TIME EXECUTIVE

An evaluation copy of the Signetics fast and efficient real-time kernal is also included. pSOS features high-speed interrupt response and task switching, ROM-based plug-in silicon operation system requiring no sysgen, hardware independent I/O, process management, memory management, message and event management, real-time clock services and more.

ORDERING INFORMATION

SMVME9100	Evaluation Kit
SMVME9110	Disk Controller plus CP/M-68K on both 8 in. single-sided single density diskettes and 5¼ in. double-sided double density diskettes
SMVME0400	250W Power Supply Option

Signetics

pSOS-68K
Real-Time, Multitasking, Operating System Kernel

Product Specification (Brief)

Microsystems Products

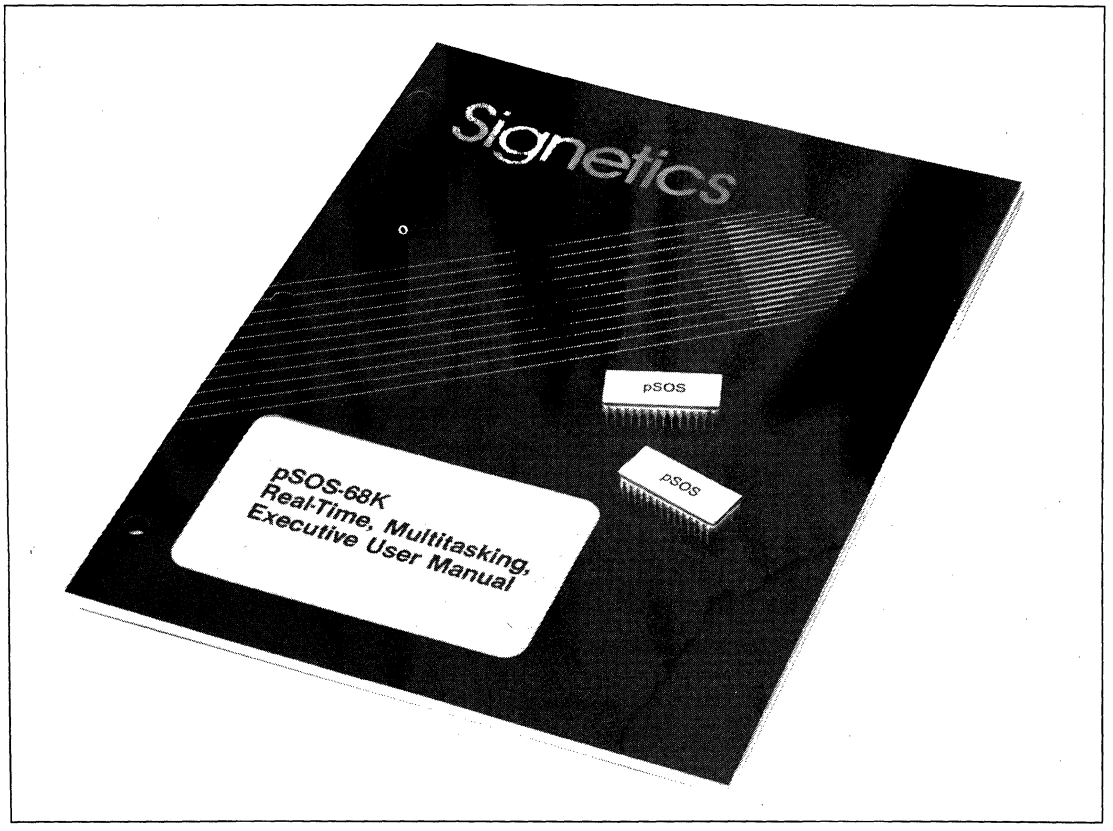
DESCRIPTION

pSOS-68K is a real-time, multitasking operating system nucleus for the Signetics SCN68000 CPU. pSOS, which stands for plug-in Silicon Operating System, is a pioneering product in state-of-the-art silicon software. Its plug-in modularity, simplicity of interface, and ease of use, represent a new direction in software components comparable to that achieved with LSI technology in hardware designs.

pSOS is shippable in silicon, and may be used much like a floating point coprocessor. Alternatively, in order to conserve PROM space and to allow flexibility for diverse system configurations, pSOS can be provided in loadable, object form which can be **plugged in** as the executive module in the user's software. pSOS is self-configuring, position-independent, and **requires no modification, compilation, or linking.** Hence, **it can be used with any development system** and implementation language. This is in contradistinction to competitive products which are offered in source language form, and thus demand language syntactic compatibility on the user's development system, and usually require extensive modifications. With pSOS-68K, the user can concentrate solely on his own software.

FEATURES

- **High-speed context switch and interrupt response**
- **Plug-in, silicon operating system**
- **Usable with any development system/language**
- **Easy-to-use interface through user traps**
- **Compact, efficient architecture – only 5KB**
- **Hardware configuration-independent**
- **Device-independent I/O**
- **No sysgen required**
- **Run-time protection mechanisms**



Real-Time, Multitasking, Operating System Kernel — pSOS-68K

pSOS-68K is extremely compact and modular, requiring only 5KB of code space. Included among its many features are **hierarchical processes with run-time protection mechanisms, named objects, and position-independent code.**

In addition to its novel, silicon-based technology, pSOS is uniquely versatile. It is excellent as a fast, multitasking executive in real-time, ROM-based applications. Moreover, pSOS-68K can serve as the OS kernel and I/O supervisor in memory-mapped, disk-based, real-time systems. pSOS allows easy, **modular customization for any MMU hardware.** Since pSOS was designed to be independent of any specific hardware configuration, it is an extremely useful product for both the component designer and boards designer. pSOS offers an efficient, compact, and manageable building block for industrial control, advanced office and graphics work stations, data-base machines, and other S68000 applications.

FUNCTIONAL OVERVIEW

pSOS-68K constitutes a layer of supervisory software around the SCN68000 CPU. Its rich feature set and simple interface encourage an efficient, structured application design, and improve productivity through reduced design and debugging efforts.

The user application is partitioned into logically concurrent processes. Related processes can be grouped, forming protected subbranches in the process tree. pSOS schedules and allocates resources between processes, and supervises asynchronous processing. Processes are scheduled by priority; equal priority processes are round-robined by time-slice.

User processes and I/O drivers interface with pSOS-68K via the user trap S68000 instructions which comprise the kernel service calls and I/O supervisor calls.

Kernel Service Requests

The rich set of kernel calls serve five groups of functions.

Process Management
The first group of service calls allow dynamic creation and deletion, prioritization, and control of processes.

Memory Management
These calls perform dynamic memory allocation and reclamation. An efficient and enhanced first-fit allocation algorithm manages up to two user-specified regions – on-board and off-board RAM. This service group is designed to be replaceable to work with any MMU, including the Signetics SCN68451. All MMU-related changes are localized to these nine calls; no other changes to pSOS are required or recommended. The Assign_segment service switches memory segments between processes, providing an efficient method of exchanging large data blocks.

Interprocess Communication
This group of primitives allow processes to communicate, synchronize, and effect mutual exclusion, and thus perform crucial services in a multitasking environment. The Message Buffer facilities provided are efficient and multipurpose. Request/send_messages, for example, supplant P/V semaphore primitives and, moreover, allow reclamation of resource-related messages upon asynchronous process deletions. Messages are routed via exchanges which provide effective N-process to M-process communications. Queuing is user selectable and Messages can be placed in either a priority or FIFO queue. A "Jam message" (Jam_x) directive allows an important message to be placed at the top of the queue.

Time-base Management
If Real-Time Clock (RTC) hardware is included, time-base services provide a real-time clock, allow processes to pause for specified periods, and provide timeout options to other kernel services such as Request_message and Allocate_memory calls.

Miscellaneous
The RETI call allows Interrupt Service routines and Privileged processes, respectively, to invoke a pSOS dispatch cycle.

I/O Supervisor Requests

The pSOS I/O Supervisor imposes a logical, device-independent structure on user provided device drivers. I/O requests are made via standard calls to the I/O Supervisor, employing logical device numbers. Each device driver must present six procedures, some of which may be simply null or error returns. I/O operations are designed for efficiency, and require no lengthy request blocks. Moreover, highly time-critical I/O can be performed outside the I/O Supervisor, using Interprocess Communication services between the Interrupt Service routines and the Servicing processes.

BUILDING AN APPLICATION

pSOS-68K is uniquely versatile and easy to use. Figure 1 depicts the components in a typical application, where dashed lines suggest data connectivity, and solid arrows indicate action flow paths.

Table 1. SERVICE AND I/O CALLS

PROCESS MANAGEMENT
Spawn_process
Activate_process
Delete_process
Identify_process
Suspend_process
Resume_process
Get_process_priority
MEMORY MANAGEMENT
Allocate_segment
Free_segment
Assign_segment
Attach_section*
Detach_section*
Lock_section*
Unlock_section*
Load_process_to_MMU*
Log_to_phys_address_xlate*
MESSAGE/EVENT MANAGEMENT
Create_exchange
Delete_exchange
Attach_exchange
Send_message
Jam_x
Request_message
Signal_process event
Wait_event
REAL-TIME CLOCK SERVICES
Announce_tick
Pause
Set_time_of_day
Get_time_of_day
INTERRUPT HANDLING
Ret_from_interrupt
I/O MANAGEMENT
Device_init
Device_open
Device_close
Device_read
Device_write
Device_control

* For MMU implementation only.

User software consists of two types of objects – Processes and I/O drivers. A small Configuration Table supplies all hardware and application-dependent parameters to pSOS, including address and size of free RAM, RTC frequency, number and location of I/O drivers, and the user Root_process descriptor.

Prior to calling the pSOS initializer, the following modules must be memory resident: pSOS, Configuration Table, Root_process, and Device drivers. Some or all of these modules may be ROM'ed or boot-loaded. Similarly, the Vector page may be ROM'ed or initialized by a small Bootprom monitor to point to the kernel entries and user interrupt handlers. The reset procedure then branches to the

4

Real-Time, Multitasking, Operating System Kernel pSOS-68K

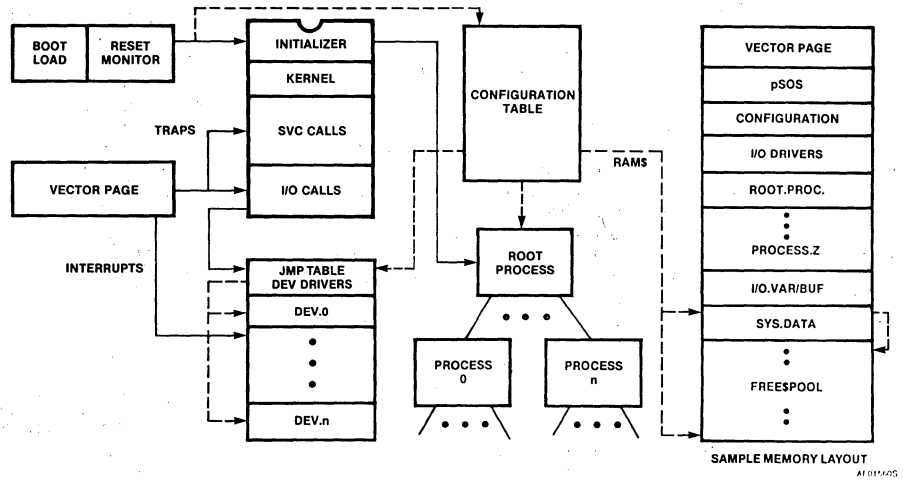


Figure 1. pSOS Application Components

pSOS initializer with a pointer to Configuration Table. Per the Configuration parameters, the initializer carves a kernel data segment from the free RAM area, and sets up its data and list structures. The Init_procedure in each Device driver is then called in turn, which resets the Device and initializes its data area for itself, if necessary. The kernel initializer then calls a routine which sizes up the remaining unused RAM and sets up the Memory Manager.

The last initialization step activates the user Root_process, whose responsibility it is to spawn, initialize, and activate all user processes needed at startup time. Thereafter, processes communicate with each other, spawn offspring processes, share resources, and perform I/O, through the rich set of pSOS services.

SPECIFICATIONS

HARDWARE REQUIREMENTS	– Memory: 5KB code, 512 bytes data minimum – Real-Time Clock: Recommended
STANDARD MODULES	– pSOS Kernel with Memory Manager (unmapped) – I/O Supervisor – Exception logger/reporter (virtual console)
DOCUMENTATION	– pSOS-68K Real-Time Multitasking Executive User's Manual
OTHER	– Maximum interrupt latency, 23μs (8MHz) – Maximum process switch, 180μs (8MHz) – 23 entry jump table replaces sysgen requirement

PART NUMBER*	**PRODUCT DESCRIPTION**
SMSFT8000	Five copies pSOS Object Code
SMSFT8001	100 copies pSOS Object Code
SMSFT8002	Unlimited copies pSOS Object Code
SMSFT8003	Unlimited copies pSOS Object Code and Source Code – 68000 Assembler Syntax

SHIPPING MEDIA OPTIONS**
Two 2764 PROMs, ½ in. magnetic tape, 800 bpi for VAX-11 computers, VMS operating system.

*Requires signed Software License Agreement.
**Please specify part number.

Signetics

SMSFT801X
pROBE System Kernel Debugger

Product Specification (Brief)

Microsystems Products

DESCRIPTION

pROBE-68K is a comprehensive, system-wide, debug and performance analysis tool for any Signetics VMEbus based systems running under the pSOS-68K Real-Time Multitasking operating system. pROBE-68K is an ideal debug tool for system integration and development, as well as a system performance monitor and measuring device for an already operational system.

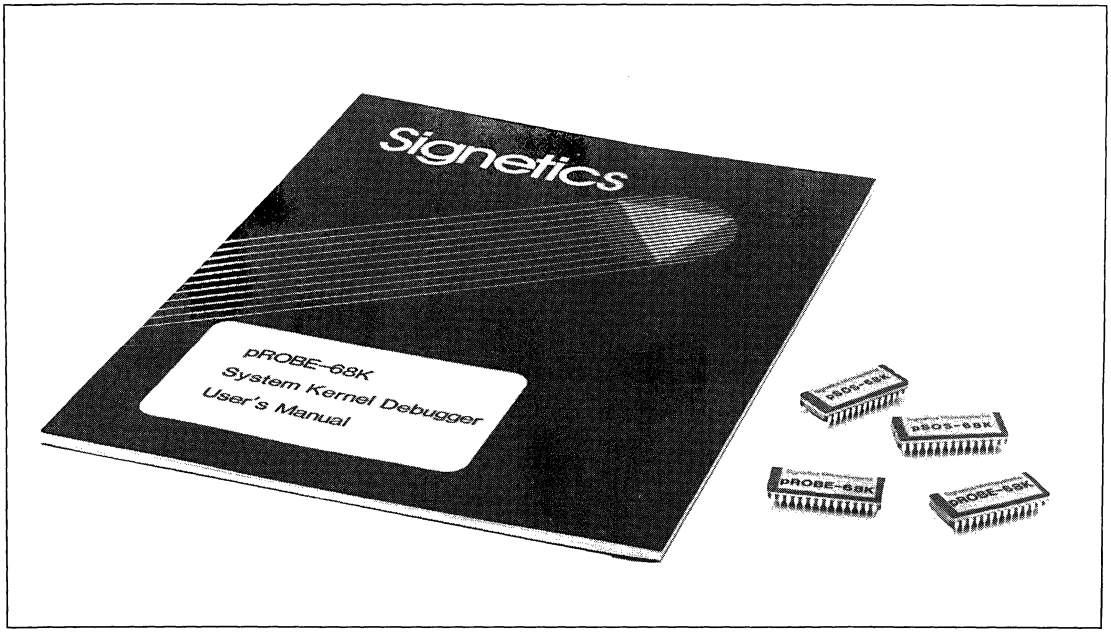
pROBE-68K is offered as an option with the pSOS-68K operating system. As a debug tool pROBE-68K takes full advantage of its knowledge of the pSOS-68K operating environment, principles, and data structures which can be especially helpful in debugging even the most complex user applications. As a performance measurement device pROBE-68K can perform such tasks as compiling actual operational statistics which aid in the location of system bottlenecks and which help to enhance and fine tune system performance.

pROBE-68K provides five categories of command functionality to offer the user complete control of the debug/monitor environment. Monitor-type commands permit memory and registers to be displayed and altered, handle most download routines, and assemble and disassemble code. Execution control commands control single stepping and both instruction and pSOS-68K related event breakpoints, such as process state changes, memory request, etc. pSOS-68K query commands produce formatted displays of status information on key system resources such as available memory, user processes, message exchanges, etc. pSOS-68K system service call commands may be invoked by pROBE-68K to manipulate, isolate, or simulate system execution and activity. Profile commands enable selective compilation of statistics on system parameters, resource utilization, and efficiency and throughput. This combination of commands helps make pROBE-68K a superior and very functional debugger.

FEATURES

- **Enhanced, comprehensive debugger & performance analysis tool**
- **Designed to work in conjunction with pSOS-68K Real-Time Multitasking operating system**
- **Supports breakpoints on instructions, kernel and I/O calls, state changes and pSOS-68K events**
- **User friendly command set**
- **Command set allows access to all VMEbus I/O, control and memory facilities and the full 16MB address range of the SCN68000**
- **Single-line assembler and disassembler for instruction modification**
- **Permits queries of key system resources and status**

4



pROBE System Kernel Debugger SMSFT801X

SUMMARY OF PROBE-68K COMMANDS

EC	Evaluate Constant
HO	Host Communication (Transparent Mode)
DL	Download from Host
DM	Display Memory
PM	Patch Memory
FM	Fill Memory
DI	Disassemble Memory
AS	In-Line Assembly (Mnemonic Patch)
DR	Display Registers
PR	Patch Register(s)
DF	Display Offset Registers
PF	Patch Offset Register(s)
GO	Start Execution
ST	Single Step
DB	Define Break
CB	Clear Breakpoint(s)
LB	List Breakpoint(s)
WB	Why Breakpoint Occurred
SC	Make pSOS Service Call
QS	Show System Status
QP	Show Process Information
QX	Show Message-Exchange Information
QM	Show Memory Manager Status
EP	Enable Profiling
LP	Display Profile Data

MEMORY REQUIREMENTS

pROBE-68K occupies 32KB of code and requires 2KB of static data storage and 512 bytes of stack.

I/O REQUIREMENTS

pROBE-68K requires a minimum of one and maximum of two serial ports. One serial port is connected to an operator (dumb) terminal. The other port may be used to communicate with a host computer and to accept data which is downloaded in S-record format.

PREREQUISITES

pSOS-68K Real-Time Multitasking Operating System.

ORDERING INFORMATION*

SMSFT8010 ** Five copies pROBE-68K and pSOS-68K Object Code. Shipping media is four 27128 PROMs.

SMSFT8011 ** 100 copies pROBE-68K and pSOS-68K Object Code. Shipping media is ½" magnetic tape, 800 bpi for VAX-11 computers, VMS operating system.

SMSFT8012 ** Unlimited copies pROBE-68K and pSOS-68K Object Code. Shipping media is ½" magnetic tape, 800 bpi for VAX-11 computers, VMS operating system.

SMSFT8013 ** Unlimited copies pROBE-68K and pSOS-68K Object Code and pSOS-68K Source Code. Shipping media is ½" magnetic tape, 800 bpi for VAX-11 computers, VMS operating system.

DOCUMENTATION

SMMAN8010 pROBE-68K System Kernel Debugger User's Manual

SMMAN8000 pSOS Real-Time, Multitasking Executive User's Manual

SMMAN8003 pSOS Design Document

*pROBE-68K and pSOS-68K require a signed Software License Agreement. Break-The-Seal licensing used with all single copy Evaluation Kits.

Signetics

SMSFT10000
S68000 Cross Software Macro Assembler

Product Specification (Brief)

Microsystems Products

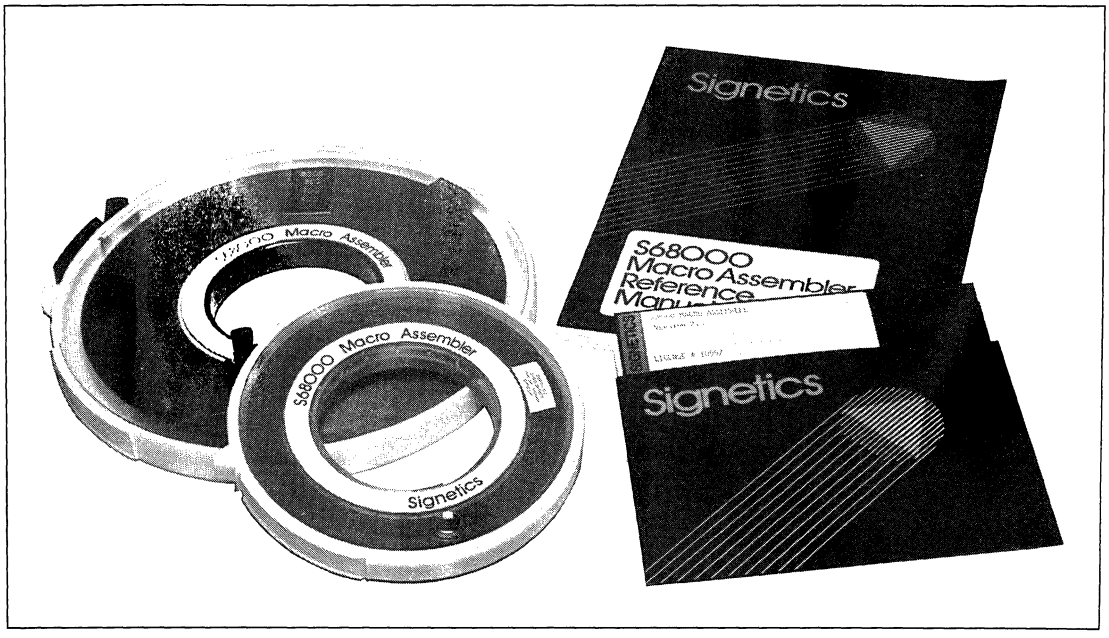
DESCRIPTION

The Signetics 68000 Macro Assembler is a powerful assembler that combines the convenience of Motorola compatible instruction mnemonics with an unambiguous language definition and many useful constructs usually found only in high-level languages. The Signetics 68000 Macro Assembler is a perfect complement to high-level languages and an excellent implementation language with capabilities such as modular programming support, record and field data definition ability, a source library facility that includes source only when referenced, and assembler directives that minimize the task of parameter passing to subroutines. The Signetics macro facility offers normal macro features such as parameter substitution and nesting but is not a character oriented facility and does not support concatenation of symbols. The linkage editor for linking separately assembled modules is included in the package. This linkage editor may also be used with the object modules generated by the Pascal cross compiler, or to link Pascal and assembler generated modules together. The upload and download software that allows host communication is also included in the package. It is designed to interface with the Signetics User Work Station (UWS) and allows transfer of object code between the host computer and the UWS as well as between the UWS and the host computer.

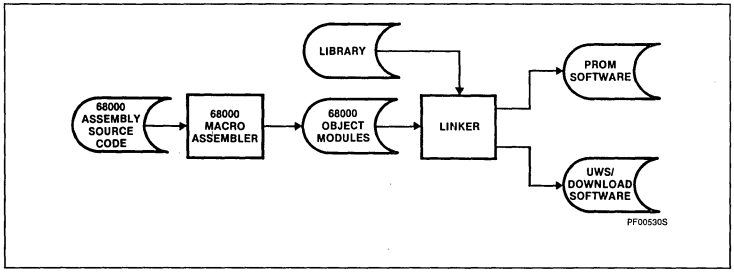
FEATURES

- **Powerful, Motorola compatible assembly language**
- **Interfaces directly to Signetics 68000 Pascal**
- **Unambiguous source language definition**
- **Registers may have symbolic names**
- **Highly flexible constant definition**
- **Both symbolic and mnemonic logical operators are supported**
- **Jump instruction format is optimized automatically**
- **Full complement of assembler directives**
- **Structured conditional assembly directives**
- **Modular programming support**
- **Source library facility**
- **Powerful macro facility**
- **Linkage editor and download software included**

4



S68000 Cross Software Macro Assembler — SMSFT10000



- The modular programming support allows sharing of program modules generated by the assembler or by high-level languages such as Pascal.
- Sections of source code identified as libraries by the "LIB" and "ENDLIB" assembler will process automatically if they are referenced.
- The conditional and duplicated assembly capabilities allow maximum flexibility in source code structure.
- The use of subroutines is encouraged by the availability of assembler directives that support the definition and manipulation of parameter lists.

BENEFITS

- The Signetics 68000 Macro Assembler is Motorola instruction compatible, so no extensive retraining is required for software engineers.

OPERATING ENVIRONMENT

OPERATING SYSTEM	COMPUTER	REQUIREMENTS
VMS	any VAX-11 computer	(a.) VMS V2.4 or later (b.) 64KB of real memory per active assembler user (c.) 10MB of disk storage
RSX-11M	any PDP-11 computer; any LSI-11/23 series computer	(a.) RSX-11M V3.2 or later (b.) minimum 128KB total system memory to support one assembler user; 64KB for each additional assembler user (c.) 10MB of disk storage

Contact Rikki Kirzner for further product information on computers, operating systems, media, or formats not shown.

ORDERING INFORMATION

ORDER NO.	PRODUCT	OPERATING SYSTEM	MEDIA
SMSFT10000	S68000 Macro Assembler*	VMS	Half- inch magnetic tape 800 bpi
SMSFT10300	S68000 Macro Assembler*	RSX- 11M	Half-inch magnetic tape 800 bpi
SMSFT10390	S68000 Macro Assembler*	RSX-11M	8- inch double density diskette

DOCUMENTATION ORDERING INFORMATION

ORDER NUMBER	PRODUCT
SMMAN5205	S68000 Macro Assembler Reference Manual
SMMAN7211	S68000 Macro Assembler Installation Guide, RSX-11M Operating System
SMMAN7210	S68000 Macro Assembler Installation Guide, VMS Operating System
SMMAN3231	S68000 Macro Assembler User Guide

*Requires Software License Agreement.

Signetics

SMSFT12000
S68000 Cross Software C Language Cross Compiler

Product Specification (Brief)

Microsystems Products

DESCRIPTION

The Signetics S68000 C Language Cross Compiler is a highly portable general purpose, low-level structured programming language for the SCN68000 Microprocessor. The C language was originally conceived and developed by Bell Telephone Laboratories as a systems implementation language for the UNIX® operating system. C has rapidly become a popular programming language for many microprocessor applications, since it easily adapts to the capabilities of most processor architectures.

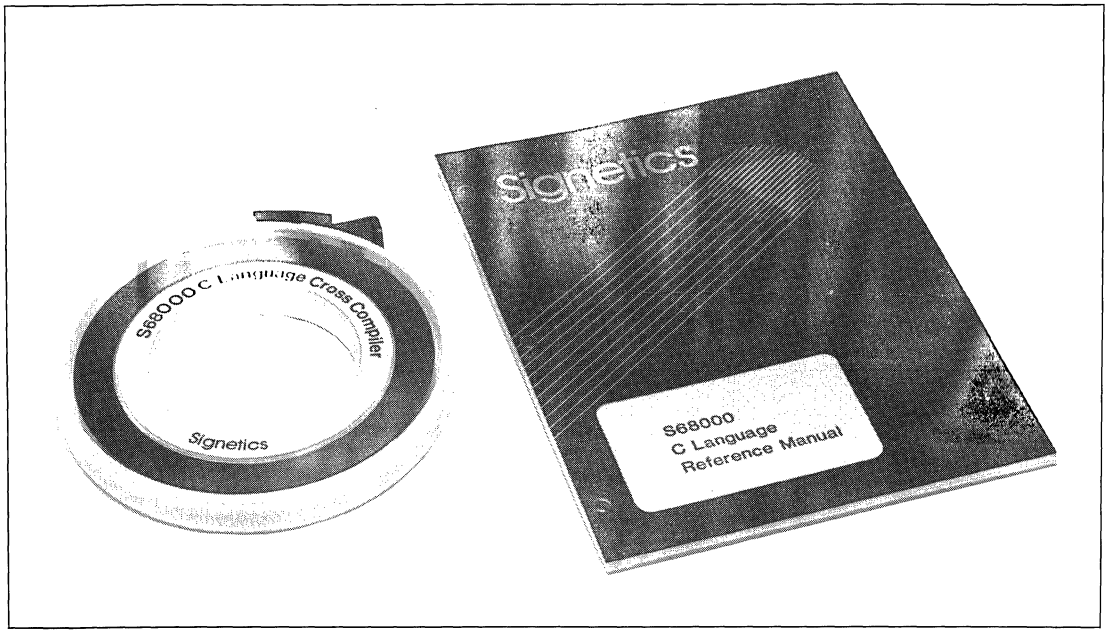
C is considered to be a low-level programming language because users can directly manipulate important aspects of the hardware from within the language. However, C supports many features and constructs found in high-level languages, including a variety of data types and operators and structured control flow statements such as if-else, while, do-while, for, and case statements.

C supports such elementary data types as bit fields, signed and unsigned integers, signed and unsigned characters, etc. These elementary data types can become the building blocks to more complex data structures such as arrays of objects. The C language contains a large number of operators which can be used to manipulate these elementary data types. C's major advantage over most high-level languages is that it permits manipulation of variables containing the addresses of operands (known as pointers). C permits memory to be referenced as directly and efficiently as assembly language does. C also supports such bit manipulation operations as masking, setting, and testing individual bits or bit fields.

FEATURES

- **General purpose, powerful systems implementation language for the SCN68000 Microprocessor**
- **Wide variety of data types, storage classes and constructs**
- **Easy interface to Signetics S68000 Macro Assembler**
- **Produces ROMable code**
- **Comprehensive set of operators**
- **Conforms to Version 7 UNIX C and the C language specification as defined in *The C Programming Language* by Kernighan and Ritchie***

4



UNIX is a registered trademark of Bell Laboratories.
*Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1978.

S68000 Cross Software C Language Cross Compiler SMSFT12000

The basic organization unit of the C program is called a function. Functions may be compiled separately and can be linked together for execution. C functions can be recursive, re-entrant and can make use of both predefined library functions and user-defined functions. All functions in C are called by value (i.e., the actual value of the parameter is passed to the function). In most high-level languages only the address of the parameter is passed to the function. This is known as call by reference, and since a pointer can be a parameter, the C language also supports call by reference.

C is especially suitable for applications traditionally written in assembly language, such as operating systems, instrumentation firmware, and real-time process control software. Programmers have explicit control over the way values are stored when they use the C language. They can define variables to be local or global and can direct that frequently used variables be stored in SCN68000 registers rather than on the stack. This produces faster access time and shorter object code.

The Signetics C Cross Compiler conforms to Version 7 UNIX C and produces assembler code for the SCN68000 Microprocessor.

```
#define true 1
#define false 0
#define size 8190
#define sizepl 8191

        char flags [sizepl];

main()
{
        int i, primes, k, count, iter;

        printf ("10 iterations \ n");
        for (iter = 1; iter< = 10; iter + +)
        {
                count = 0;
                for(i = 0; i< = size; i + +)
                        flags [i] = true;
                for(i = 0; i<= size; i + +)
                {
                        if (flags[i])
                        {
                                primes = i+i+3;
                                k = i + primes;
                                while (k< = size)
                                {
                                        flags[k] =false;
                                        k + = primes;
                                }
                                count = count+1;
                        }
                }
        }
        printf(" \ n%d primes \ n",count);
}
```

DF00330S

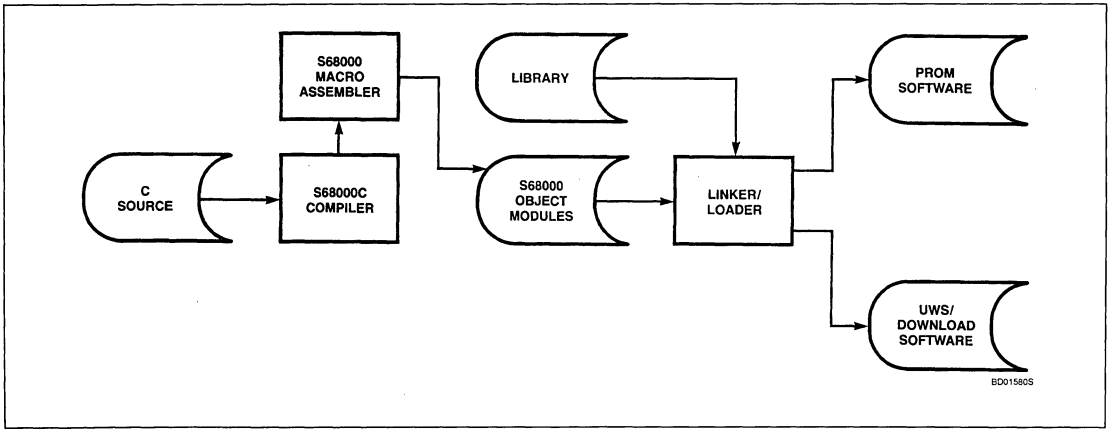
S68000 Cross Software C Language Cross Compiler SMSFT12000

FEATURES

- Multiple data types, storage classes and high-level constructs.
- Allows explicit control over storage of values and provides address and bit manipulation.
- Permits separate compilation of functions.
- Portable general purpose systems implementation language for S68000.
- Conforms to Version 7 UNIX C and the specification defined by Kernighan and Ritchie in *The C Programming Language*, 1978.

BENEFITS

- Programmer can express objects in a way that is more natural to his application.
- Gives user control over code generation and hardware address similar to that of assembler language.
- Supports a modular tool-building approach to software development and allows programmers to interface C and S68000 assembler routines.
- C is suitable for a wide range of applications.
- C is an established, widely used and well defined language.



OPERATING ENVIRONMENT

OPERATING SYSTEM	COMPUTERS	REQUIREMENTS
VMS	any VAX-11 computer	(a.) 64KB of real memory per active compilor usor (b.) VMS V3.4 or later (c.) 10MB of disk storage

ORDERING INFORMATION

ORDER NUMBER	PRODUCT	OPERATING SYSTEM	MEDIA
SMSFT12000	C Language S68000 Cross Compiler S68000 Macro Assembler*	VMS	Half-inch magnetic tape 800 bpi

*Requires Software License Agreement.
For other host computers, operating systems, or alternate media contact factory.

DOCUMENTATION ORDERING INFORMATION

ORDER NUMBER	PRODUCT
SMMAN5210	S68000 C Language Reference Manual
SMMAN5205	S68000 Macro Assembler Reference Manual

Signetics

SMSFT16000
S68000 Cross Software Pascal Cross Compiler

Product Specification (Brief)

Microsystems Products

DESCRIPTION

The Signetics 68000 Pascal Cross Compiler is a powerful, flexible, high-level programming language. It complies with the IEEE 770-81 standard which is based on the Pascal language developed by Niklaus Wirth in 1968. The compiler provides a highly structured environment that simplifies program development, produces more reliable code, and increases the productivity of the software engineer. Programming errors are minimized by complete checking of data types within modules including user defined data types.

The Signetics 68000 Pascal Cross Compiler is designed to be extremely modular with an emphasis on portability. The compiler, itself, is a result of utilizing truly state-of-the-art compiler design techniques to produce a superior language translator.

The Pascal Cross Compiler allows the mass storage and development tools available on minicomputers and mainframes to be used in the creation of microprocessor software. It is not necessary to purchase expensive, special purpose computers that are typically used only for microprocessor software development. The upload/download software provides a convenient path for object code to be sent from the host computer to the Signetics User Work Station (UWS) and for partially or completely debugged object code to be saved on the disk of the host computer. All object code is sent utilizing a protocol that includes error detection and control facilities to ensure the validity of the transmitted data.

FEATURES

- **High-level structured programming language**
- **Structured data types**
  - **Array, string, record, boolean, character**
  - **User-defined data types**
- **Modular programming extension**
- **Assembly language program interface**
- **Complies with IEEE 770-81 standard**
- **Generates 68000 native code**
- **PROMable run-time code**
- **Full listing format control**
- **Flexible compiler option control**
- **Cross linkage editor and upload/download software included**

S68000 Cross Software Pascal Cross Compiler | SMSFT16000

SIGNETICS PASCAL EXTENSIONS

Separate Compilation
- Allows modular programming and facilitates sharing of routines by multiple programs and projects.

Assembly Language Interface
- Permits mixing Pascal and assembly language routines for maximum performance and ease of direct hardware control.

INCLUDE Facility
- Facilitates sharing of source code and declarations of constants and variables by multiple programs and projects.

Compile Time Switches
- Provides control of compiler options such as array index checking.

Assembly Code Generation
- Simplifies performance tuning of critical program modules.

BENEFITS

- Signetics Pascal provides a standardized environment for developing software for the 68000.
- It allows individuals familiar with Pascal to be immediately productive.
- It is easy to learn since the IEEE 770-81 standard it adheres to is based on the Niklaus Wirth software engineering teaching vehicle.
- It has few machine specific extensions to maximize portability.
- It improves productivity by allowing multiple programs and projects to share both source and object code.
- An assembly language interface is provided to allow direct manipulation of hardware and maximize the performance of time critical modules.
- The Pascal compiler can optionally generate assembly language with source code interlisted as comments to minimize the task of performance tuning in real-time systems.
- A full range of listing controls is provided to enhance maintainability of the source code.
- The compiler generated code is optimized in constant and sub-expression usage, array indexing, jumps, and storage allocation for maximum use of target system resources.
- An object module linker with library capability is included with the Pascal, which links together Pascal and assembly language modules and allows specification of starting addresses of all modules.

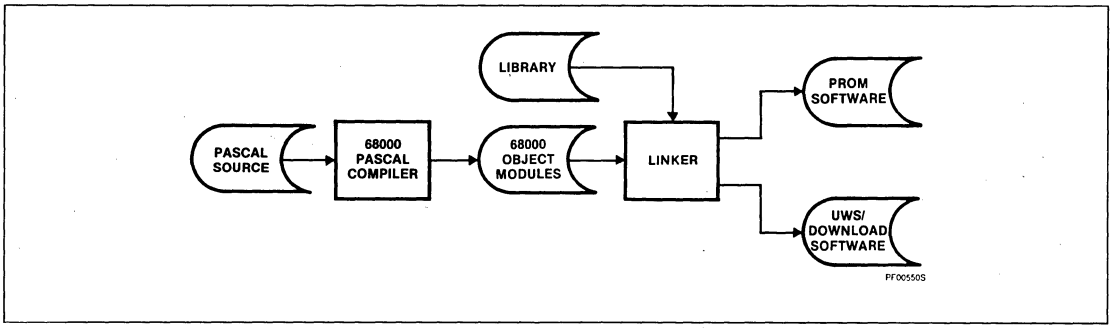
EXAMPLE

The following Pascal program is an example of the structure of a Pascal program. This program is designed to eliminate all trailing blanks from the end of each line within a file such that the line termination character occurs after the last non-blank character in the line. It illustrates the use of subroutines, typing of variables, and loop structures. The actual implementation of input and output routines is system dependent, but the read and write statements used are those of the IEEE 770-81 standard.

```
program compact (source, destination);
        const   BLANK = ' ';
                LINESIZE = 140;
        type    linerange = 1..LINESIZE;
                linetype = packed array[linerange] of char;
        var     source, destination : text;
                line : linetype;
                length, cur, lastnonblank : linerange:
                c : char;
                procedure getsourceline;
                begin   length : = 1;
                        line[length] : = BLANK;
                        while not (eoln (source)) and
                        (length < LINESIZE) do
                begin   read (source, c);
                        line[length] : = c;
                        length : = length + 1
                end;
                readln (source);
                if length > 1 then length : = length - 1
                end;
        begin   reset (source);
                rewrite (destination)
                while not eof(source) do
                begin getsourceline;
                      lastnonblank : = 1;
                      for cur : = 1 to length
                      do if line[cur] < > BLANK
                      then last nonblank : = cur;
                      for cur : = 1 to lastnonblank
                      do write (destination, line[curl]);
                      writeln (destination)
        end     end.
```

4

S68000 Cross Software Pascal Cross Compiler SMSFT16000



OPERATING ENVIRONMENT

OPERATING SYSTEM	COMPUTERS	REQUIREMENTS
VMS	any VAX-11 computer	(a.) 64KB of real memory per active compiler user (b.) VMS V2.4 or later (c.) 10MB of disk storage
RSX-11M	any PDP-11 computer; any LSI-11/23 series computer	(a.) RSX-11M operating system V3.2 or later (b.) minimum 128KB total system memory to support one compiler user; 64KB for each additional compiler user (c.) 10MB of disk storage

ORDERING INFORMATION

ORDER NUMBER	PRODUCT	OPERATING SYSTEM	MEDIA
SMSFT16000	Pascal 68000* S68000 Macro Assembler*	VMS	Half-inch magnetic tape 800 bpi
SMSFT16300	Pascal 68000* S68000 Macro Assembler*	RSX-11M	Half-inch magnetic tape 800 bpi
SMSFT16390	Pascal 68000* S68000 Macro Assembler*	RSX-11M	8-inch double density diskette

*Requires Software License Agreement.
For other host computers, operating systems, or alternate media contact factory.

DOCUMENTATION ORDERING INFORMATION

ORDER NUMBER	PRODUCT
SMMAN5200	S68000 Pascal Language Reference Manual
SMMAN5205	S68000 Macro Assembler Reference Manual
SMMAN7200	S68000 Pascal Installation Guide, VMS Operating System
SMMAN7210	S68000 Macro Assembler Installation Guide, VMS Operating System
SMMAN7201	S68000 Pascal Installation Guide, RSX-11M Operating System
SMMAN7211	S68000 Macro Assembler Installation Guide, RSX-11M Operating System
SMMAN3230	S68000 Pascal Compiler User Guide
SMMAN3231	S68000 Macro Assembler User Guide

Signetics

SMSFT51970
SIGbug Monitor

Product Specification (Brief)

Microsystems Products

DESCRIPTION

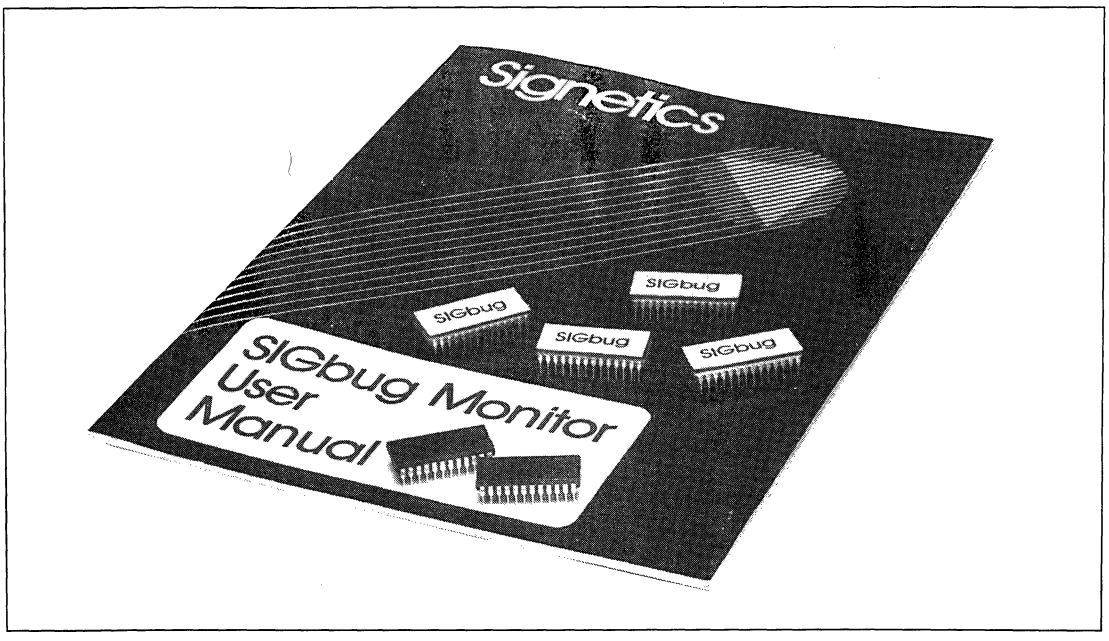
SIGbug is an EPROM-based resident package ready for installation and immediate use with any Signetics VMEbus CPU installed in a VMEbus-based backplane.

SIGbug provides a powerful evaluation and system debugging tool for VMEbus systems. The 27128 EPROM-based package will operate in 64KB of ROM space. SIGbug uses the first 4096 words of system RAM storage for interrupt vectors and temporary storage; half of this may be relocated at the user's discretion. In order to take full advantage of the capability of SIGbug, at least 2048 words of off-board memory must be available. The package permits full speed execution and control of system and user developed programs in a Signetics VMEbus system environment. SIGbug may be utilized with a VMEbus CPU board or an entire VMEbus system in a standalone environment with only a user provided standard RS-232C asynchronous ASCII terminal required as additional equipment. Alternately, it may be used with a second serial I/O port connected to a host computer for downloading of data in Motorola "S" record format. When connected to a host computer in this manner, the SIGbug console terminal appears as a normal asynchronous ASCII terminal to the host operating system. The second serial I/O port, the host computer interface, would be implemented through the second asynchronous serial I/O port provided on the CPU board.

FEATURES

- **27128 EPROM resident VMEbus system debug monitor**
- **18 commands for debug**
- **Single line assembler/ disassembler for ease of program modification**
- **Full speed execution of system and user developed programs operating in the VMEbus system environment**
- **Virtual terminal capability for download from a cross development host computer**
- **Powerful software and system debug command set allows access to all VMEbus I/O, control and memory facilities plus the full 16MB direct address range of the 68000 processor**
- **Includes all required installation and operation documentation**

4



SIGbug Monitor | SMSFT51970

OPERATIONAL DESCRIPTION

In a typical debug session, the user will download his program to a Signetics VMEbus-based system from the host computer used for software development. After loading, SIGbug commands may be used to examine and modify memory, set breakpoints to control program execution, and track program progress. The user may set-up and examine a variety of conditions using any of the powerful commands listed in the table to the right, such as the register display/patch series and the memory manipulation commands. If corrections or program patches are required, these may be performed and checked in the Signetics VMEbus-based board system by using the built-in line assembler/disassembler.

The user may communicate with the host computer as a terminal for purposes other than download by executing the host mode command. By using the Configure command, the baud rate for the host serial port may be changed.

SIGbug may be used for debug in total systems environments which include other VMEbus compatible boards.

ORDERING INFORMATION

SMSFT51970	SIGbug, 27128 EPROM, for Signetics 68000 VMEbus-based boards, monitor, object
SMMAN326A	SIGbug User Manual

SIGBUG COMMANDS

COMMAND	DESCRIPTION
AS [< syntax type >] [< address >]	Enter line assembler
DI < address range >	Disassemble
DM < start address > [.. < end address >]	Display memory
DR	Display registers
PM < address > < data string > [* < multiplier >] | < address range > < data string >	Patch memory
PR < register > < value > { < register > < value > }	Patch registers
GO [< address >]	Execute program
SS [ON | OFF | < step-count >]	Single step program execution
CF [< host port baud rate >]	Configure host port
DL < file > [< address range >] [< load address >]	Download S-records
HO [< exit character >] [< echo >] [< host type >]	Transparent mode
DB < address > | < all >	Delete breakpoints
DT < address > | < all >	Delete traces
LB	List breakpoints
LT	List traces
SB < address > [L < loop count >]	Set breakpoint
ST < address range > [NR]	Set trace
RS < address range >	Relocate stack
Command Line Edit and Control Functions	
(BREAK)	Abort command
(DEL)	Delete character
(CTRL-H)	Delete character
(CTRL-Q)	Continue output
(CTRL-S)	Suspend/continue output
(CTRL-X)	Cancel Command Line
(CTRL-Z)	Exit from subenvironment
(CR)	Command Terminator

Signetics

Microprocessor Products

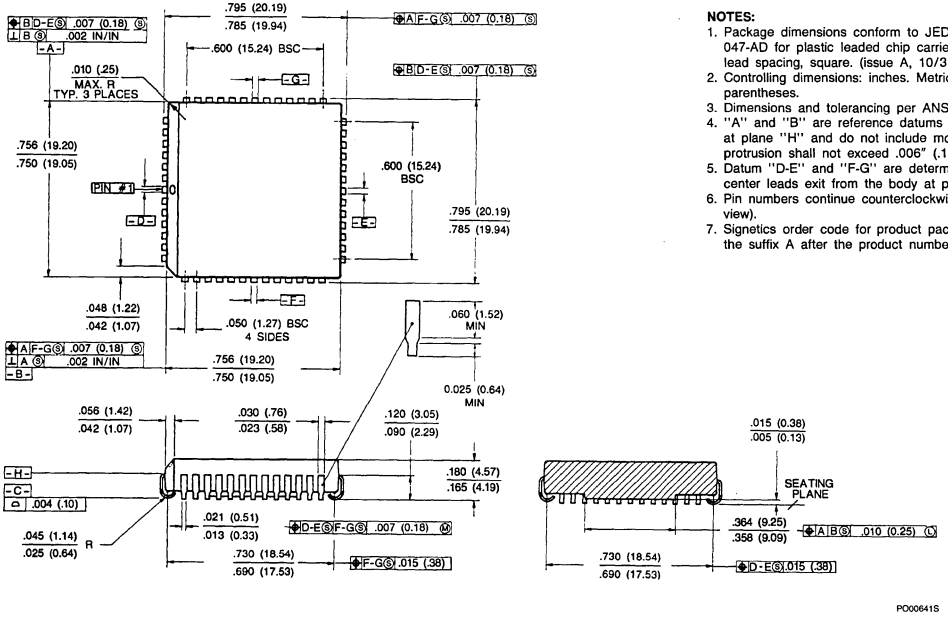
Section 5
Package Outlines

5

January 1986

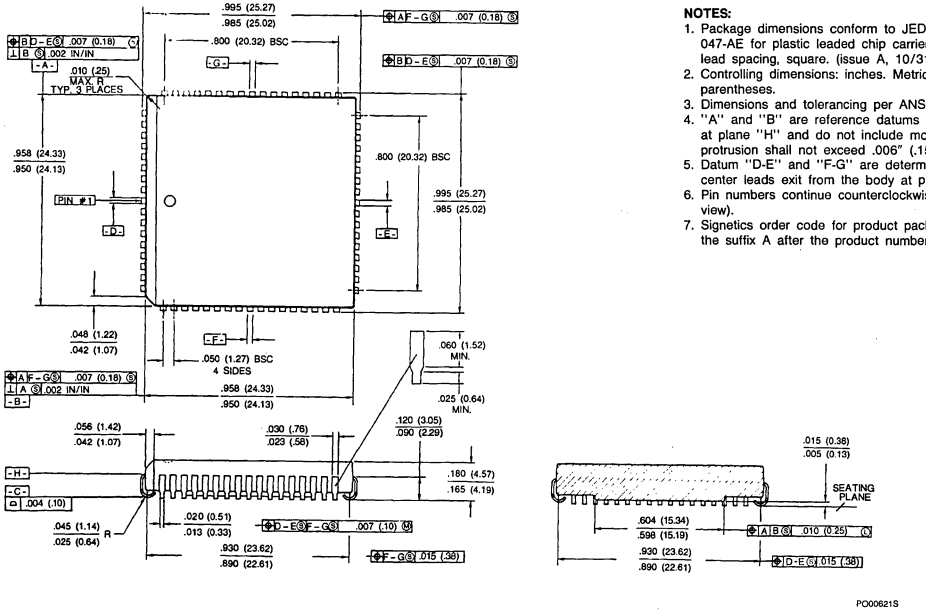
Package Outlines

AA1 52-PIN PLCC



NOTES:
1. Package dimensions conform to JEDEC specification MO-047-AD for plastic leaded chip carrier 52 leads, .050 inch lead spacing, square. (issue A, 10/31/84).
2. Controlling dimensions: inches. Metric are shown in parentheses.
3. Dimensions and tolerancing per ANSI Y14.5M-1982.
4. "A" and "B" are reference datums on the molded body at plane "H" and do not include mold flash. Mold flash protrusion shall not exceed .006" (.15mm) on any side.
5. Datum "D-E" and "F-G" are determined where these center leads exit from the body at plane "H".
6. Pin numbers continue counterclockwise to Pin 52 (top view).
7. Signetics order code for product packaged in a PLCC is the suffix A after the product number.

AB1 68-PIN PLCC

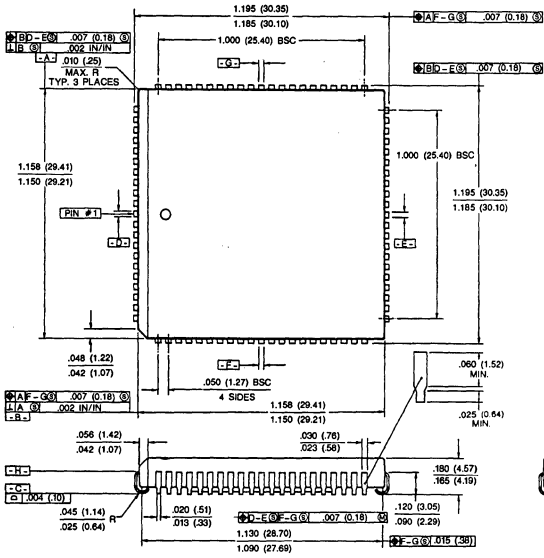


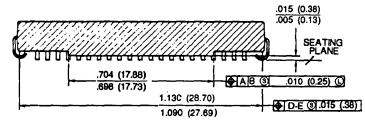
NOTES:
1. Package dimensions conform to JEDEC specification MO-047-AE for plastic leaded chip carrier 68 leads, .050 inch lead spacing, square. (issue A, 10/31/84).
2. Controlling dimensions: inches. Metric are shown in parentheses.
3. Dimensions and tolerancing per ANSI Y14.5M-1982.
4. "A" and "B" are reference datums on the molded body at plane "H" and do not include mold flash. Mold flash protrusion shall not exceed .006" (.15mm) on any side.
5. Datum "D-E" and "F-G" are determined where these center leads exit from the body at plane "H".
6. Pin numbers continue counterclockwise to Pin 68 (top view).
7. Signetics order code for product packaged in a PLCC is the suffix A after the product number.

5

Package Outlines

AC1 84-PIN PLCC

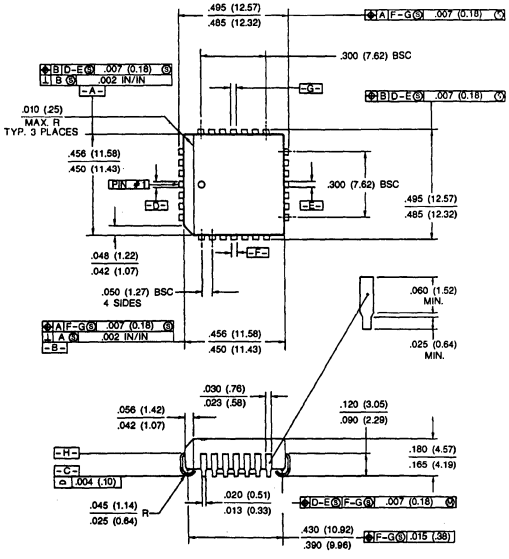


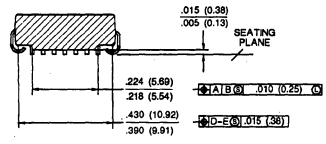


NOTES:
1. Package dimensions conform to JEDEC specification MO-047-AF for plastic leaded chip carrier 84 leads, .050 inch lead spacing, square. (issue A, 10/31/84).
2. Controlling dimensions: inches. Metric are shown in parentheses.
3. Dimensions and tolerancing per ANSI Y14.5M-1982.
4. "A" and "B" are reference datums on the molded body at plane "H" and do not include mold flash. Mold flash protrusion shall not exceed .006" (.15mm) on any side.
5. Datum "D-E" and "F-G" are determined where these center leads exit from the body at plane "H".
6. Pin numbers continue counterclockwise to Pin 84 (top view).
7. Signetics order code for product packaged in a PLCC is the suffix A after the product number.

PO00591S

AQ1 28-PIN PLCC



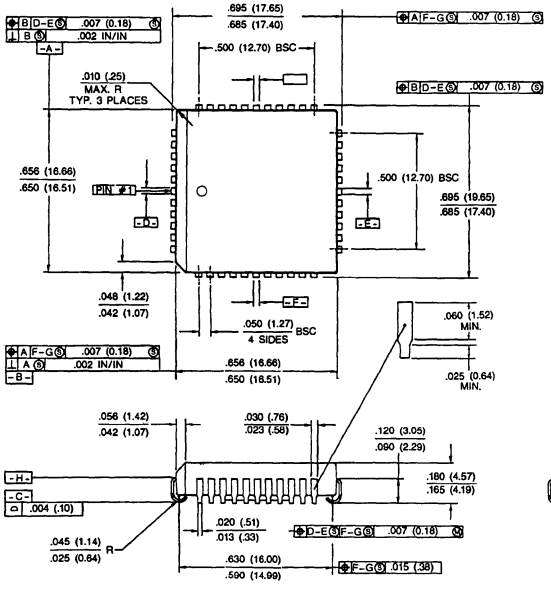


NOTES:
1. Package dimensions conform to JEDEC specification MO-047-AB for plastic leaded chip carrier 28 leads, .050 inch lead spacing, square. (issue A, 10/31/84).
2. Controlling dimensions: inches. Metric are shown in parentheses.
3. Dimensions and tolerancing per ANSI Y14.5M-1982.
4. "A" and "B" are reference datums on the molded body at plane "H" and do not include mold flash. Mold flash protrusion shall not exceed .006" (.15mm) on any side.
5. Datum "D-E" and "F-G" are determined where these center leads exit from the body at plane "H".
6. Pin numbers continue counterclockwise to Pin 28 (top view).
7. Signetics order code for product packaged in a PLCC is the suffix A after the product number.

PO00611S

Package Outlines

AX1 44-PIN PLCC



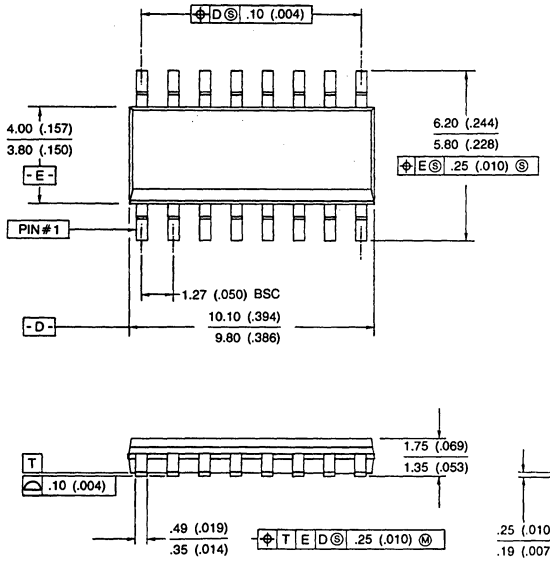
NOTES:

1. Package dimensions conform to JEDEC specification MO-047-AL for plastic leaded chip carrier 44 leads, .050 inch lead spacing, square. (issue A, 10/31/84).
2. Controlling dimensions: inches. Metric are shown in parentheses.
3. Dimensions and tolerancing per ANSI Y14.5M-1982.
4. "A" and "B" are reference datums on the molded body at plane "H" and do not include mold flash. Mold flash protrusion shall not exceed .006" (.15mm) on any side.
5. Datum "D-E" and "F-G" are determined where these center leads exit from the body at plane "H".
6. Pin numbers continue counterclockwise to Pin 44 (top view).
7. Signetics order code for product packaged in a PLCC is the suffix A after the product number.

5

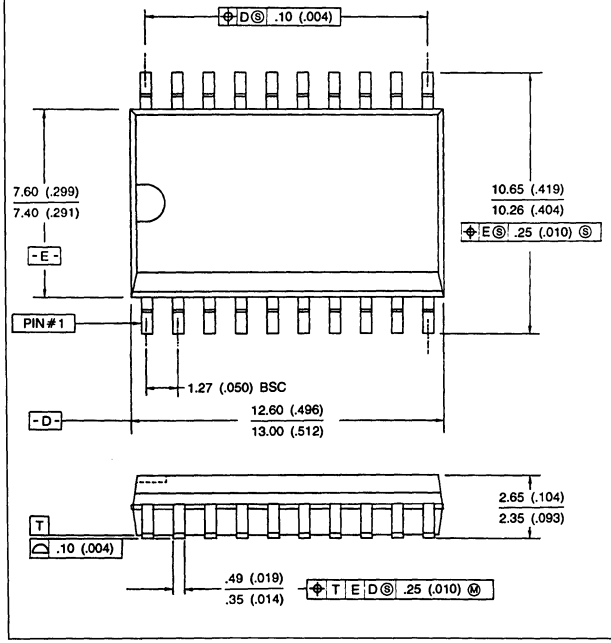
Package Outlines

DJ1 16-PIN SO



NOTES:
1. Package dimensions conform to JEDEC specification MS-012-AC for standard small outline (SO) package, 16 leads, 3.75mm (.150") body width (issue A, June 1985).
2. Controlling dimensions are mm. Inch dimensions in parentheses.
3. Dimensions and tolerancing per ANSI Y14.5M- 1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .15mm (.006") on any side.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #16 when viewed from top.
6. Signetics ordering code for a product packaged in a plastic small outline (SO) package is the suffix D after the product number.

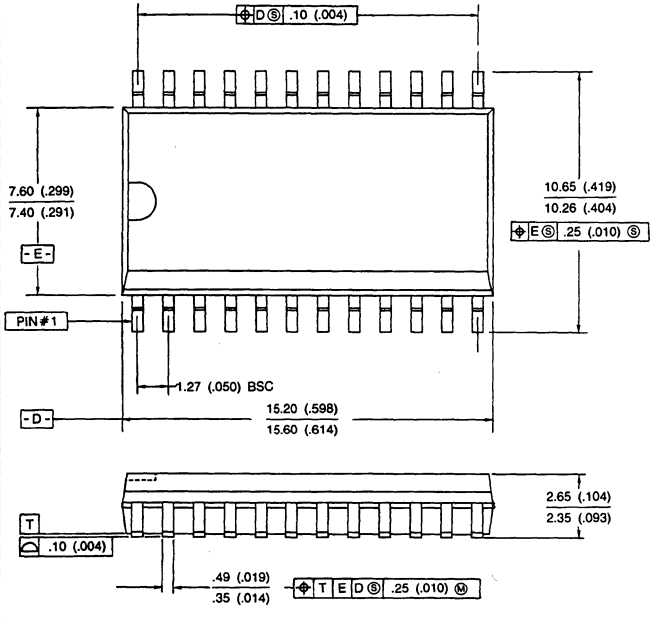
DL2 20-PIN SO

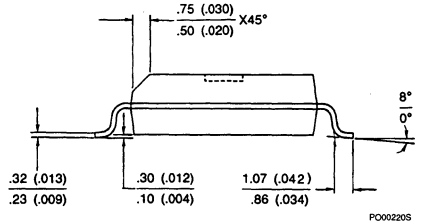


NOTES:
1. Package dimensions conform to JEDEC specification MS-013-AC for standard small outline (SO) package, 20 leads, 7.50mm (.300") body width (issue A, June 1985).
2. Controlling dimensions are mm. Inch dimensions in parentheses.
3. Dimensions and tolerancing per ANSI Y14.5M- 1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .15mm (.006") on any side.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #20 when viewed from top.
6. Signetics ordering code for a product packaged in a plastic small outline (SO) package is the suffix D after the product number.

Package Outlines

DN2 24-PIN SO





PO00220S

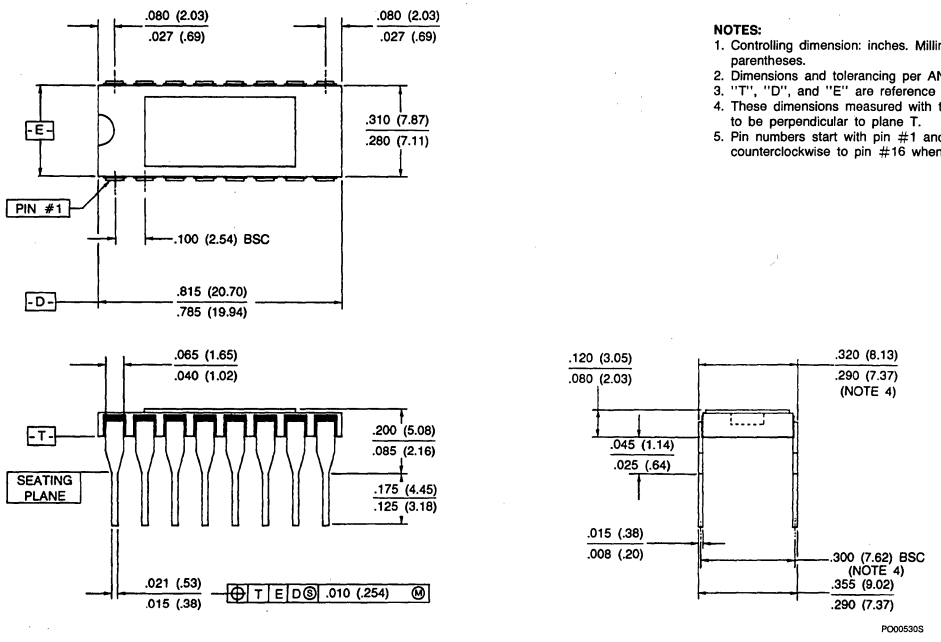
NOTES:

1. Package dimensions conform to JEDEC specification MS-013-AD for standard small outline (SO) package, 24 leads, 7.50mm (.300") body width (issue A, June 1985).
2. Controlling dimensions are mm. Inch dimensions in parentheses.
3. Dimensions and tolerancing per ANSI Y14.5M- 1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .15mm (.006") on any side.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #24 when viewed from top.
6. Signetics ordering code for a product packaged in a plastic small outline (SO) package is the suffix D after the product number.

5

Package Outlines

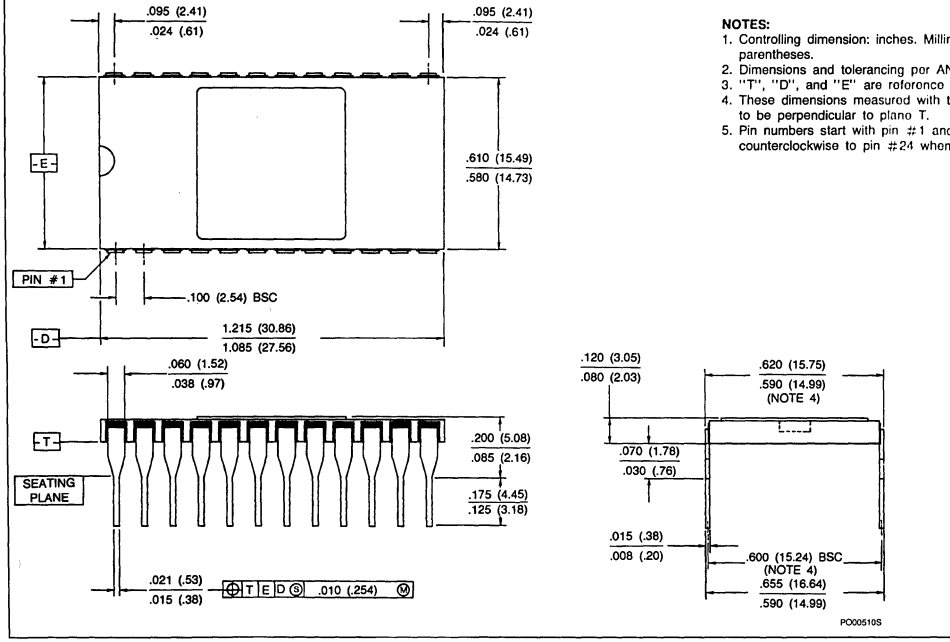
IJ1 16-PIN CERAMIC DIP



NOTES:
1. Controlling dimension: inches. Millimeters are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M – 1982.
3. "T", "D", and "E" are reference datums on the body.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #16 when viewed from the top.

PO00530S

IN2 24-PIN CERAMIC DIP

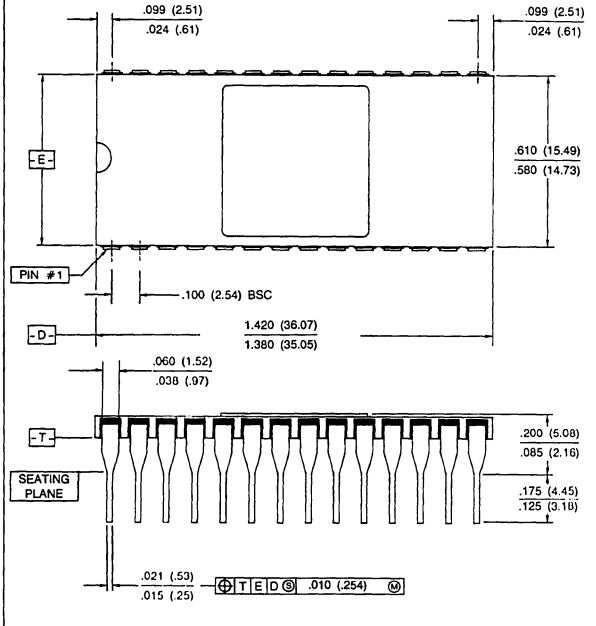


NOTES:
1. Controlling dimension: inches. Millimeters are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M – 1982.
3. "T", "D", and "E" are reference datums on the body.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #24 when viewed from the top.

PO00510S

Package Outlines

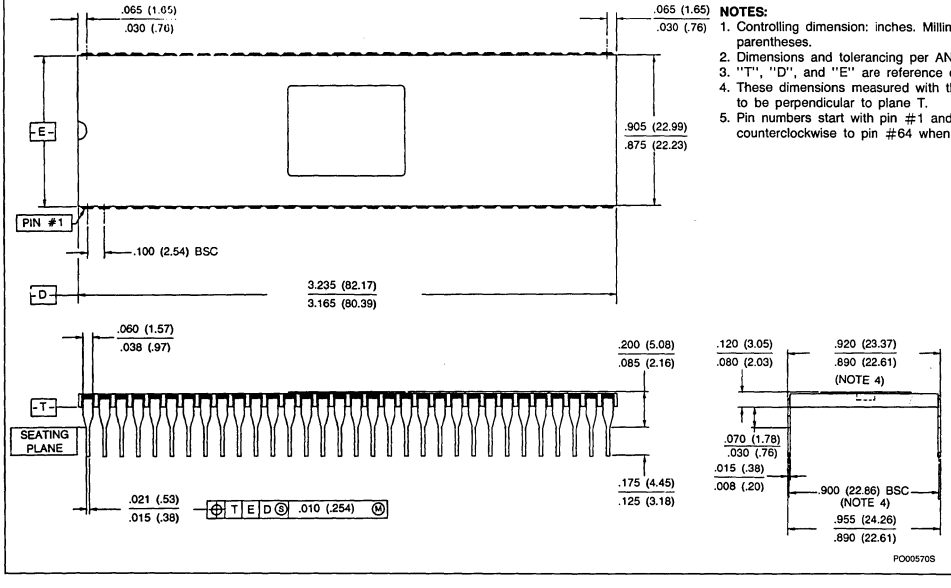
IQ3 28-PIN CERAMIC DIP



PO00500S

NOTES:
1. Controlling dimension: inches. Millimeters are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M – 1982.
3. "T", "D", and "E" are reference datums on the body.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #28 when viewed from the top.

IS4 64-PIN CERAMIC DIP



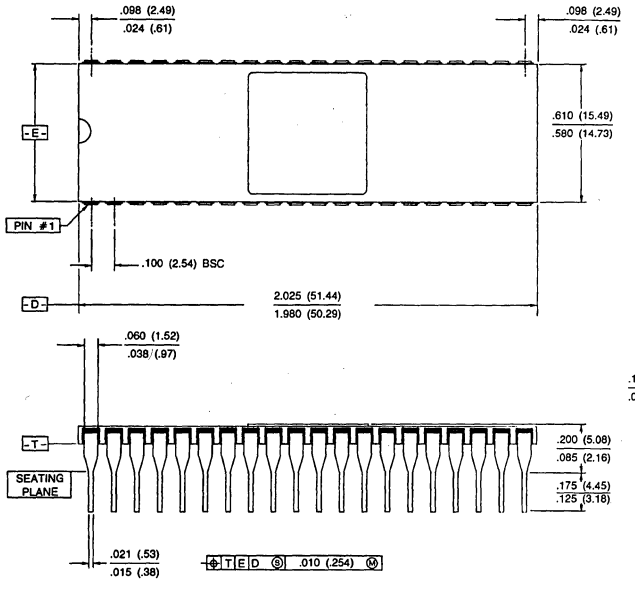
PO00570S

NOTES:
1. Controlling dimension: inches. Millimeters are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M – 1982.
3. "T", "D", and "E" are reference datums on the body.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #64 when viewed from the top.

5

Package Outlines

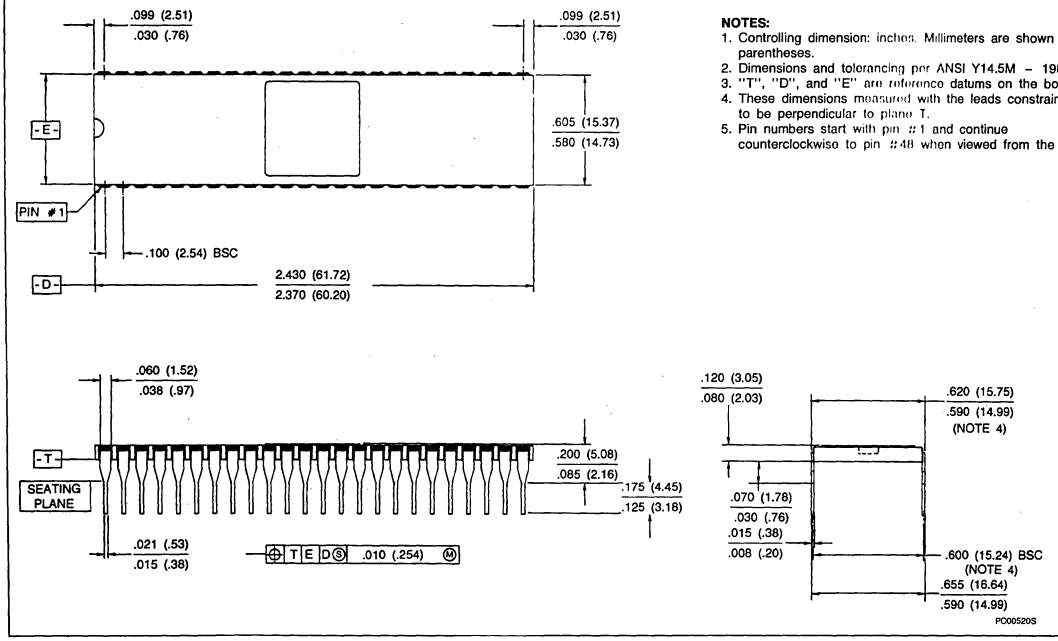
IW3 40-PIN CERAMIC DIP



NOTES:
1. Controlling dimension: inches. Millimeters are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M – 1982.
3. "T", "D", and "E" are reference datums on the body.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #40 when viewed from the top.

POXXXXXS

IY3 48-PIN CERAMIC DIP

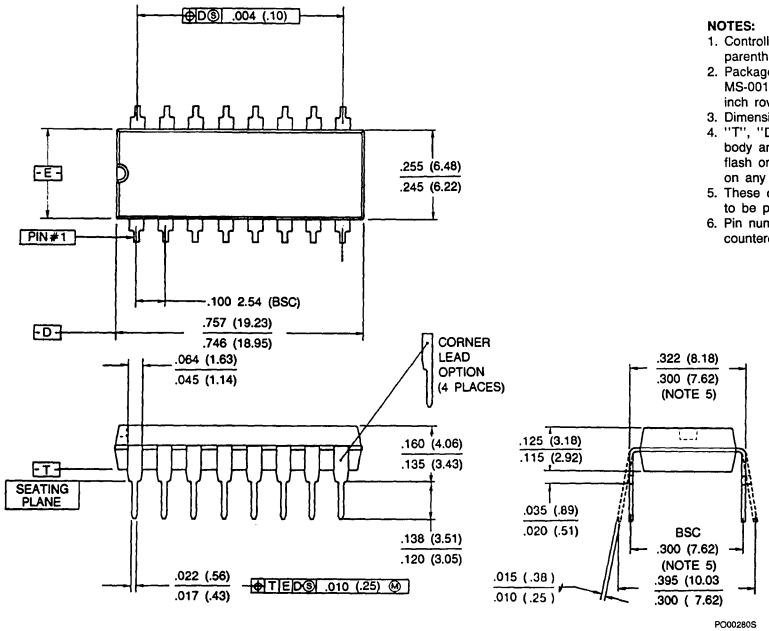


NOTES:
1. Controlling dimension: inches. Millimeters are shown in parentheses.
2. Dimensions and tolerancing per ANSI Y14.5M – 1982.
3. "T", "D", and "E" are reference datums on the body.
4. These dimensions measured with the leads constrained to be perpendicular to plane T.
5. Pin numbers start with pin #1 and continue counterclockwise to pin #48 when viewed from the top.

PO00520S

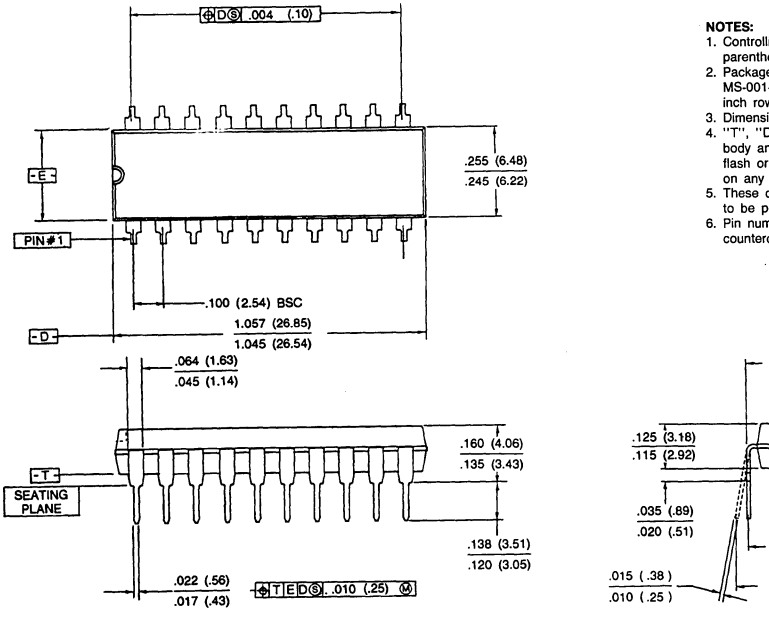
Package Outlines

NJ1 16-PIN PLASTIC DIP



NOTES:
1. Controlling dimension: Inches, metric are shown in parentheses.
2. Package dimensions conform to JEDEC specification MS-001-AA for standard dual in-line (DIP) package .300 inch row spacing (PLASTIC) 16 leads (issue B. 7/85)
3. Dimensions and tolerancing per ANSI Y14. 5M-1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010 inch (.25mm) on any side.
5. These dimensions measured with the leads constrained to be perpendicular to plane T.
6. Pin numbers start with pin #1 and continue counterclockwise to pin #16 when viewed from the top.

NL1 20-PIN PLASTIC DIP

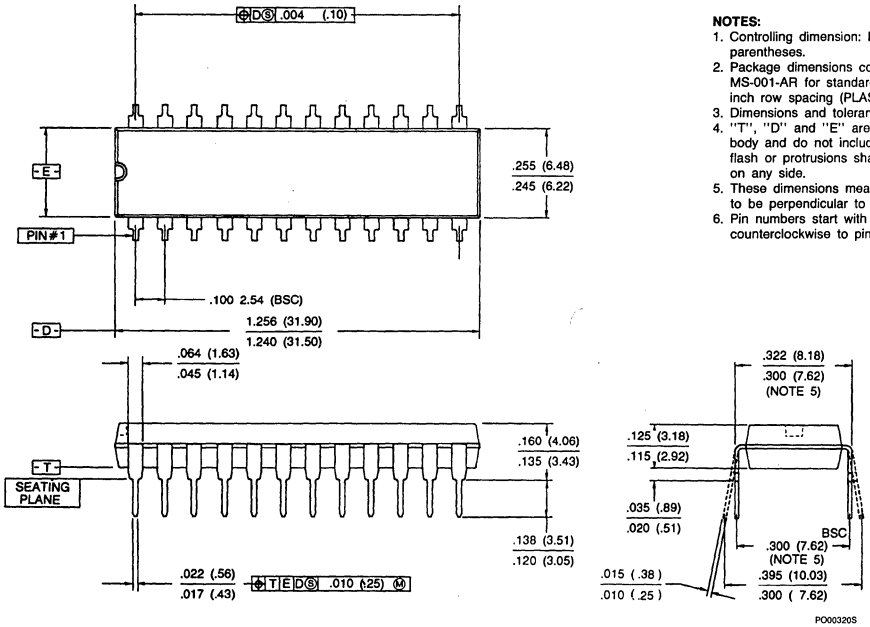


NOTES:
1. Controlling dimension: Inches, metric are shown in parentheses.
2. Package dimensions conform to JEDEC specification MS-001-AE for standard dual in-line (DIP) package .300 inch row spacing (PLASTIC) 20 leads (issue B. 7/85)
3. Dimensions and tolerancing per ANSI Y14. 5M-1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010 inch (.25mm) on any side.
5. These dimensions measured with the leads constrained to be perpendicular to plane T.
6. Pin numbers start with pin #1 and continue counterclockwise to pin #20 when viewed from tho top.

5

Package Outlines

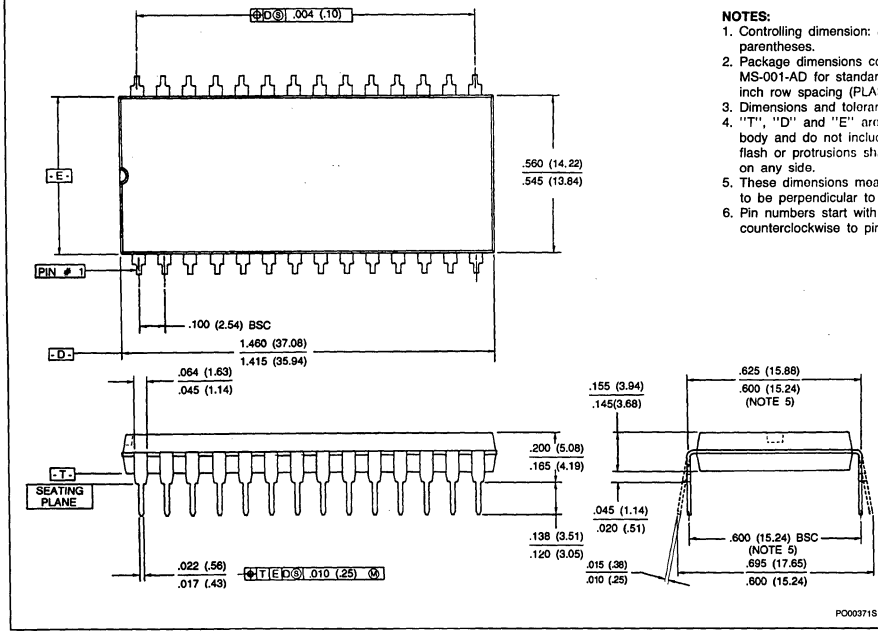
NN1 24-PIN PLASTIC DIP



NOTES:
1. Controlling dimension: Inches, metric are shown in parentheses.
2. Package dimensions conform to JEDEC specification MS-001-AR for standard dual in-line (DIP) package .300 inch row spacing (PLASTIC) 24 leads (issue B. 7/85)
3. Dimensions and tolerancing per ANSI Y14. 5M-1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010 inch (.25mm) on any side.
5. These dimensions measured with the leads constrained to be perpendicular to plane T.
6. Pin numbers start with pin #1 and continue counterclockwise to pin #24 when viewed from the top.

PO00320S

NQ3 28-PIN PLASTIC DIP

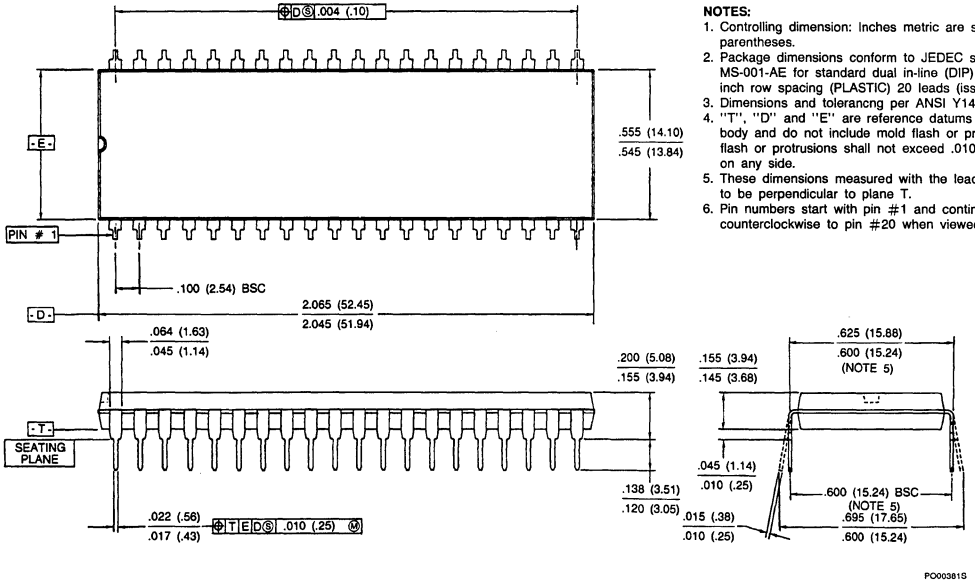


NOTES:
1. Controlling dimension: Inches metric are shown in parentheses.
2. Package dimensions conform to JEDEC specification MS-001-AD for standard dual in-line (DIP) package .300 inch row spacing (PLASTIC) 18 leads (issue B. 7/85)
3. Dimensions and tolerancing per ANSI Y14. 5M-1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010 inch (.25mm) on any side.
5. These dimensions measured with the leads constrained to be perpendicular to plane T.
6. Pin numbers start with pin #1 and continue counterclockwise to pin #18 when viewed from the top.

PO00371S

Package Outlines

NW3 40-PIN PLASTIC DIP



NOTES:
1. Controlling dimension: Inches metric are shown in parentheses.
2. Package dimensions conform to JEDEC specification MS-001-AE for standard dual in-line (DIP) package .300 inch row spacing (PLASTIC) 20 leads (issue B. 7/85)
3. Dimensions and tolerancng per ANSI Y14. 5M-1982.
4. "T", "D" and "E" are reference datums on the molded body and do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010 inch (.25mm) on any side.
5. These dimensions measured with the leads constrained to be perpendicular to plane T.
6. Pin numbers start with pin #1 and continue counterclockwise to pin #20 when viewed from the top.

NY3 48-PIN PLASTIC DIP — Please consult factory for packaging information

NS4 64-PIN PLASTIC DIP — Please consult factory for packaging information

PB1 68-PIN GRID ARRAY — Please consult factory for packaging information

120-PIN GRID ARRAY — Please consult factory for packaging information

5

Signetics

Microprocessor Products

Section 6
Sales Offices

6

January 1986

Sales Offices

SIGNETICS HEADQUARTERS

811 East Arques Avenue
P.O. Box 3409
Sunnyvale,
California 94088-3409
Phone: (408) 991-2000

ALABAMA
Huntsville
Phone: (205) 830-4001

ARIZONA
Phoenix
Phone: (602) 265-4444

CALIFORNIA
Canoga Park
Phone: (818) 340-1431

Irvine
Phone: (714) 833-8980
(213) 588-3281

Los Angeles
Phone: (213) 670-1101

San Diego
Phone: (619) 560-0242

Sunnyvale
Phone: (408) 991-3737

COLORADO
Aurora
Phone: (303) 751-5011

CONNECTICUT
Brookfield
Phone: (203) 775-6888

FLORIDA
Clearwater
Phone: (813) 796-7086

Ft. Lauderdale
Phone: (305) 486-6300

GEORGIA
Atlanta
Phone: (404) 953-0067

ILLINOIS
Itasca
Phone: (312) 250-0050

INDIANA
Kokomo
Phone: (317) 453-6462

KANSAS
Overland Park
Phone: (913) 469-4005

MARYLAND
Glen Burnie
Phone: (301) 787-0220

MASSACHUSETTS
Littleton
Phone: (617) 486-8411

MICHIGAN
Farmington Hills
Phone: (313) 476-1610

MINNESOTA
Edina
Phone: (612) 835-7455

NEW JERSEY
Parsippany
Phone: (201) 334-4405

NEW YORK
Hauppauge
Phone: (516) 348-7877

Liverpool
Phone: (315) 451-5470

Wappingers Falls
Phone: (914) 297-4074

NORTH CAROLINA
Cary
Phone: (919) 481-0400

OHIO
Worthington
Phone: (614) 888-7143

OREGON
Portland
Phone: (503) 297-5592

PENNSYLVANIA
Horsham
Phone: (215) 443-5500

TENNESSEE
Greeneville
Phone: (615) 639-0251

TEXAS
Austin
Phone: (512) 339-9944

Richardson
Phone: (214) 644-3500

CANADA
SIGNETICS CANADA, LTD.
Etobicoke, Ontario
Phone: (416) 626-6676

Nepean, Ontario
Signetics, Canada, Ltd.
Phone: (613) 726-9576

REPRESENTATIVES

ARIZONA
Scottsdale
Thom Luke Sales, Inc.
Phone: (602) 941-1901

CALIFORNIA
Los Gatos
Magna Sales, Inc.
Phone: (408) 354-1626

Santa Clara
Magna Sales
Phone: (408) 727-8753

San Diego
Mesa Engineering
Phone: (619) 278-8021

CONNECTICUT
Brookfield
Kanan Associates
Phone: (203) 775-0494

ILLINOIS
Hoffman Estates
Micro-Tex, Inc.
Phone: (312) 382-3001

KANSAS
Kansas City
B.C. Electronic Sales
Phone: (913) 342-1211

Wichita
B.C. Electronic Sales
Phone: (316) 722-0104

MARYLAND
Columbia
Delta III
Phone: (301) 730-4700

MASSACHUSETTS
Needham Heights
Kanan Associates
Phone: (617) 449-7400

MICHIGAN
Bloomfield Hills
Enco Marketing
Phone: (313) 642-0203

MINNESOTA
Eden Prairie
High Technology Sales
Phone: (612) 944-7274

MISSOURI
Bridgeton
B.C. Electronic Sales
Phone: (314) 291-1101

NEW JERSEY
East Hanover
Emtec Sales, Inc.
Phone: (201) 428-0600

NEW MEXICO
Albuquerque
F.P. Sales
Phone: (505) 345-5553

NEW YORK
Ithaca
Bob Dean, Inc.
Phone: (607) 257-1111

Melville
Emtec Sales, Inc.
Phone: (516) 752-1630

OKLAHOMA
Tulsa
Jerry Robinson and
Associates
Phone: (918) 665-3562

OREGON
Hillsboro
Western Technical Sales
Phone: (503) 640-4621

PENNSYLVANIA
Horsham
Delta Technical Sales Inc.
Phone: (215) 443-5503

Pittsburgh
Covert & Newman
Phone: (412) 531-2002

TEXAS
Houston
OM Sales
Phone: (713) 789-4426

UTAH
Salt Lake City
Electrodyne
Phone: (801) 486-3801

WASHINGTON
Bellevue
Western Technical Sales
Phone: (206) 641-3900

Spokane
Western Technical Sales
Phone: (509) 922-7600

WISCONSIN
Waukesha
Micro-Tex, Inc.
Phone: (414) 542-5352

Sales Offices

CANADA

Nepean, Ontario
Tech-Trek, Ltd.
Phone: (613) 564-0049

Pointe Claire, Quebec
Tech-Trek, Ltd.
Phone: (514) 697-3385

Rexdale, Ontario
Tech-Trek, Ltd.
Phone: (416) 674-1717

Richmond, B.C.
Tech-Trek, Ltd.
Phone: (604) 271-3149

Winnipeg, Manitoba
Tech-Trek, Ltd.
Phone: (204) 222-1321

DISTRIBUTORS

ALABAMA

Huntsville
Hamilton/Avnet Electronics
Phone: (205) 837-7210
Schweber Electronics
Phone: (205) 882-2200

ARIZONA

Phoenix
Schweber Electronics
Phone: (602) 997-4874
Wyle LEMG
Phone: (602) 249-2232

Tempe
Arrow Electronics
Phone: (602) 968-4800
Hamilton/Avnet Electronics
Phone: (602) 231-5100

CALIFORNIA

Calabasas
Wyle LEMG
Phone: (818) 880-9000

Canoga Park
Schweber Electronics
Phone: (818) 999-4702

Chatsworth
Anthem Electronics
Phone: (818) 700-1000
Arrow Electronics
Phone: (818) 701-7500
Avnet Electronics
Phone: (818) 700-2600
Hamilton/Avnet Electronics
Phone: (818) 700-6500

Costa Mesa
Avnet Electronics
Phone: (714) 754-6111
Hamilton Electro Sales
Phone: (714) 641-4100

Culver City
Hamilton Electro Sales
Phone: (213) 558-2121

Garden Grove
Wyle LEMG Computer Products
Phone: (714) 891-1717

Gardena
Schweber Electronics
Phone: (213) 327-8409

Hayward
Arrow Electronics
Phone: (415) 487-4600

Irvine
Anthem Electronics
Phone: (714) 768-4444
Schweber Electronics
Phone: (714) 863-0200
Wyle LEMG
Phone: (714) 863-1611

Ontario
Hamilton/Avnet Electronics
Phone: (714) 989-4602

Rancho Cordova
Wyle LEMG
Phone: (916) 638-5282

Sacramento
Hamilton/Avnet Electronics
Phone: (916) 925-2216
Schweber Electronics
Phone: (916) 929-9732

San Diego
Anthem Electronics
Phone: (619) 453-4871
Arrow Electronics
Phone: (619) 565-4800
Hamilton/Avnet Electronics
Phone: (619) 571-7510
Schweber Electronics
Phone: (619) 450-0454
Wyle LEMG
Phone: (619) 565-9171

San Jose
Anthem Electronics Inc.
Phone: (408) 946-8000
Schweber Electronics
Phone: (408) 946-7171

Santa Clara
Wyle LEMG
Phone: (408) 727-2500

Sunnyvale
Arrow Electronics
Phone: (408) 745-6600
Hamilton/Avnet Electronics
Phone: (408) 743-3355

Tustin
Arrow Electronics
Phone: (714) 838-5422

Woodland Hills
Avnet Electronics
Phone: (818) 700-2600

COLORADO

Aurora
Arrow Electronics
Phone: (303) 696-1111

Denver
Wyle Distribution Group
Phone: (303) 457-9953

Englewood
Anthem Electronics
Phone: (303) 790-4500
Hamilton/Avnet Electronics
Phone: (303) 779-9998
Schweber Electronics
Phone: (303) 799-0258

Thornton
Wyle LEMG
Phone: (303) 457-9953

CONNECTICUT

Danbury
Hamilton/Avnet Electronics
Phone: (203) 797-2800
Schweber Electronics
Phone: (203) 748-7080

Meriden
Lionex Corp.
Phone: (203) 237-2282

Wallingford
Arrow Electronics
Phone: (203) 265-7741

FLORIDA

Altomont Springs
Schweber Electronics
Phone: (305) 331-7555

Deerfield Beach
Arrow Electronics
Phone: (305) 429-8200

Ft. Lauderdale
Hamilton/Avnet Electronics
Phone: (305) 971-2900

Hollywood
Schweber Electronics
Phone: (305) 927-0511

Palm Bay
Arrow Electronics
Phone: (305) 725-1480

St. Petersburg
Hamilton/Avnet Electronics
Phone: (813) 576-3930

Winter Park
Hamilton/Avnet Electronics
Phone: (305) 628-3888

GEORGIA

Norcross
Arrow Electronics
Phone: (404) 449-8252
Hamilton/Avnet Electronics
Phone: (404) 447-7507
Schweber Electronics
Phone: (404) 449-9170

ILLINOIS

Elk Grove
Schweber Electronics
Phone: (312) 364-3750

Schaumburg
Arrow Electronics
Phone: (312) 397-3440

Bensenville
Hamilton/Avnet Electronics
Phone: (312) 860-7700

INDIANA

Carmel
Hamilton/Avnet Electronics
Phone: (317) 844-9333

Indianapolis
Arrow Electronics
Phone: (317) 243-9353

IOWA

Cedar Rapids
Hamilton/Avnet Electronics
Phone: (319) 362-4757
Schweber Electronics
Phone: (319) 373-1417

KANSAS

Overland Park
Hamilton/Avnet Electronics
Phone: (913) 888-8900
Schweber Electronics
Phone: (913) 492-2921

MARYLAND

Columbia
Arrow Electronics
Phone: (301) 995-0003
Hamilton/Avnet Electronics
Phone: (301) 995-3500
Lionex Corp.
Phone: (301) 964-0040

Sales Offices

Gaithersburg
Schweber Electronics
Phone: (301) 840-5900

MASSACHUSETTS
Wilmington
Lionex Corp.
Phone: (617) 657-5170
Schweber Electronics
Phone: (617) 275-5100

Woburn
Arrow Electronics
Phone: (617) 933-8130
Hamilton/Avnet Electronics
Phone: (617) 273-7500

MICHIGAN
Ann Arbor
Arrow Electronics
Phone: (313) 971-8220

Grand Rapids
Hamilton/Avnet Electronics
Phone: (616) 243-8805

Livonia
Hamilton/Avnet Electronics
Phone: (313) 522-4700
Schweber Electronics
Phone: (313) 525-8100

MINNESOTA
Edina
Arrow Electronics
Phone: (612) 830-1800
Schweber Electronics
Phone: (612) 941-5280

Minnetonka
Hamilton/Avnet Electronics
Phone: (612) 932-0600

MISSOURI
Earth City
Hamilton/Avnet Electronics
Phone: (314) 344-1200
Schweber Electronics
Phone: (314) 739-0526

St. Louis
Arrow Electronics
Phone: (314) 567-6888

NEW HAMPSHIRE
Manchester
Arrow Electronics
Phone: (603) 668-6968
Hamilton/Avnet Electronics
Phone: (603) 624-9400
Schweber Electronics
Phone: (603) 625-2250

NEW JERSEY
Cherry Hill
Hamilton/Avnet Electronics
Phone: (609) 424-0100

Fairfield
Arrow Electronics
Phone: (201) 575-5300
Hamilton/Avnet Electronics
Phone: (201) 575-3390
Lionex Corporation
Phone: (201) 227-7960
Schweber Electronics
Phone: (201) 227-7880

Marlton
Arrow Electronics
Phone: (609) 596-8000

NEW MEXICO
Albuquerque
Hamilton/Avnet Electronics
Phone: (505) 765-1500
Arrow Electronics
Phone: (505) 243-4566

NEW YORK
Buffalo
Summit Distributors
Phone: (716) 887-2800

East Syracuse
Hamilton/Avnet Electronics
Phone: (315) 437-2641

Hauppauge, L.I.
Arrow Electronics
Phone: (516) 231-1000
Hamilton/Avnet Electronics
Phone: (516) 231-9800
Lionex Corp.
Phone: (516) 273-1660

Liverpool
Arrow Electronics
Phone: (315) 652-1000

Rochester
Arrow Electronics
Phone: (716) 427-0300
Hamilton/Avnet Electronics
Phone: (716) 475-9130
Schweber Electronics
Phone: (716) 424-2222

Westbury, L.I.
Schweber Electronics
Phone: (516) 334-7474

NORTH CAROLINA
Raleigh
Arrow Electronics
Phone: (919) 876-3132
Hamilton/Avnet Electronics
Phone: (919) 878-0810
Schweber Electronics
Phone: (919) 876-0000

OHIO
Beechwood
Schweber Electronics
Phone: (216) 464-2970

Centerville
Arrow Electronics
Phone: (513) 435-5563

Cleveland
Hamilton/Avnet Electronics
Phone: (216) 831-3500

Dayton
Hamilton/Avnet Electronics
Phone: (513) 439-6700
Schweber Electronics
Phone: (513) 439-1800

Solon
Arrow Electronics
Phone: (216) 248-3990

Westerville
Hamilton/Avnet Electronics
Phone: (614) 882-7004

OKLAHOMA
Tulsa
Quality Components
Phone: (918) 664-8812
Schweber Electronics
Phone: (918) 622-8000

OREGON
Hillsboro
Wyle LEMG
Phone: (503) 640-6000

Lake Oswego
Anthem Electronics
Phone: (503) 684-2661
Hamilton/Avnet Electronics
Phone: (503) 635-8831

Tigard
Arrow Electronics
Phone: (503) 684-1690

PENNSYLVANIA
Horsham
Lionex Corp.
Phone: (215) 443-5150
Schweber Electronics
Phone: (215) 441-0600

Monroeville
Arrow Electronics
Phone: (412) 856-7000

Pittsburgh
Hamilton/Avnet Electronics
Phone: (412) 281-4150
Schweber Electronics
Phone: (412) 782-1600

TEXAS
Addison
Quality Components
Phone: (214) 733-4300

Austin
Arrow Electronics
Phone: (512) 835-4180
Hamilton/Avnet Electronics
Phone: (512) 837-8911
Quality Components
Phone: (512) 835-0220
Wyle LEMG
Phone: (512) 834-9957
Schweber Electronics
Phone: (512) 458-8253

Carrolton
Arrow Electronics
Phone: (214) 380-6464

Dallas
Schweber Electronics
Phone: (214) 661-5010

Houston
Arrow Electronics
Phone: (713) 530-4700
Hamilton/Avnet Electronics
Phone: (713) 780-1771
Schweber Electronics
Phone: (713) 784-3600
Wyle LEMG
Phone: (713) 879-9953

Irving
Hamilton/Avnet Electronics
Phone: (214) 659-4111

Richardson
Wyle LEMG
Phone: (214) 235-9953

Sugar Land
Quality Components
Phone: (713) 240-2255

UTAH
Salt Lake City
Anthem Electronics
Phone: (801) 973-8555
Arrow Electronics
Phone: (801) 972-0404
Hamilton/Avnet Electronics
Phone: (801) 972-2800
Wyle LEMG
Phone: (801) 974-9953

WASHINGTON
Bellevue
Arrow Electronics
Phone: (206) 643-4800
Hamilton/Avnet Electronics
Phone: (206) 453-5844
Wyle LEMG
Phone: (206) 453-8300

Redmond
Anthem Electronics
Phone: (206) 881-0850

WISCONSIN

Sales Offices

Brookfield
Schweber Electronics
Phone: (414) 784-9020

New Berlin
Hamilton/Avnet Electronics
Phone: (414) 784-4510

Oak Creek
Arrow Electronics
Phone: (414) 764-6600

CANADA
Brampton, Ontario
Zentronics, Ltd.
Phone: (416) 451-9600

Burnaby, Vancouver
Hamilton/Avnet Electronics
Phone: (604) 272-4242

Calgary, Alberta
Hamilton/Avnet Electronics
Phone: (403) 230-3586
Zentronics, Ltd.
Phone: (403) 272-1021

Downsview, Ontario
Arrow/Cesco
Phone: (416) 661-0220

Mississauga, Ontario
Hamilton/Avnet Electronics
Phone: (416) 677-7432

Montreal, Quebec
Arrow/Cesco
Phone: (514) 735-5511
Zentronics, Ltd.
Phone: (514) 735-5361

Nepean, Ontario
Arrow/Cesco
Phone: (613) 226-6903
Hamilton/Avnet Electronics
Phone: (613) 226-1700
Zentronics, Ltd.
Phone: (613) 226-8840

Quebec, Quebec
Arrow/Cesco
Phone: (418) 687-4231

Richmond, B.C.
Zentronics, Ltd.
Phone: (604) 273-5575

Ville St. Laurent, Quebec
Hamilton/Avnet Electronics
Phone: (514) 335-1000
Zentronics, Ltd.
Phone: (514) 735-5361

Waterloo
Zentronics, Ltd.
Phone: (519) 884-5700

Winnipeg
Zentronics, Ltd.
Phone: (204) 775-8661

FOR SIGNETICS PRODUCTS WORLDWIDE:

ARGENTINA
Philips Argentina S.A.
Buenos Aires
Phone: (1) 652-3983

AUSTRALIA
Philips Industries Holdings Ltd.
Artarmon, N.S.W.
Phone: (2) 439-3322

AUSTRIA
Osterrichische Philips Bauelemente
Wien
Phone: 43-222-93-26-2

BELGIUM
N.V. Philips & MBLE
Bruxelles
Phone: 32-02-242-7400

BRAZIL
Ibrape
Sao Paulo
Phone: (11) 211-2600

CHILE
Philips Chilena S.A.
Santiago
Phone: (2) 39-4001

DENMARK
Miniwatt A/S
Kobenhavn N.V.
Phone: 45-01-69-1622

FINLAND
Oy Philips Ab
Helsinki
Phone: 358-1-7271

FRANCE
R.T.C. La Radiotechnique-Compelec
Paris
Phone: 33.1.338.8000

GERMANY
Valvo
Hamburg
Phone: 49-40-3296-19

GREECE
Philips S.A. Hellenique
Athens
Phone: 9215111

HONG KONG
Philips Hong Kong, Ltd.
Kwai Chung
Phone: (0) 245121

INDIA
Philips India & Elect. Ltd.
Bombay
Phone: (22) 212628

INDONESIA
P.T. Philips-Ralin Electronics
Jakarta
Phone: (21) 512572

IRELAND
Philips Electrical Ltd.
Dublin
Phone: 353-1-69-3355

ISRAEL
Rapac Electronics, Ltd.
Tel Aviv
Phone: (3) 477115

ITALY
Philips S.p.A.
Milano
Phone: 39-2-6994

JAPAN
Signetics Japan, Ltd.
Tokyo
Phone: 813-230-1521
Osaka
Phone: 816-304-6171

KOREA
Philips Industries (Korea) Ltd.
Seoul
Phone: (2) 794-5011

MEXICO
Electronica S.A. de C.V.
Mexico D.F.
Phone: (721) 61300

NETHERLANDS
Philips Nederland B.V.
Eindhoven
Phone: 31-40-79-3333

NEW ZEALAND
Philips Electrical Ind. Ltd.
Auckland
Phone: (9) 605914

NORWAY
Norsk A/S Philips
Oslo
Phone: 47-2-680200

PERU
Cadesa
Lima
Phone: (14) 319253

PHILIPPINES
Philips Industrial Dev., Inc.
Makati-Rizal
Phone: (2) 868951

PORTUGAL
Philips Portuguesa SARL
Lisboa
Phone: 351-19-68-3121

SINGAPORE
Philips Project Dev. Pte., Ltd.
Singapore
Phone: 350-2538

SOUTH AFRICA
E.D.A.C. (PTY), Ltd.
Joubert Park
Phone: (11) 401-4600

SPAIN
Miniwatt S.A.
Barcelona
Phone: 301 63 12

SWEDEN
A.B. Elcoma
Stockholm
Phone: 46-08-67-9780

SWITZERLAND
Philips A.G.
Zurich
Phone: 41-01-988-2211

TAIWAN
Philips Taiwan, Ltd.
Taipei
Phone: (2) 712-0500

THAILAND
Philips Electrical Co. of Thailand Ltd.
Bangkok
Phone: 233-6330-9

TURKEY
Turk Philips Ticaret A.S.
Istanbul
Phone: 43 59 10

UNITED KINGDOM
Mullard, Ltd.
London
Phone: 44-01-580-6633

UNITED STATES
Signetics International Corp.
Sunnyvale, California
Phone: (408) 739-7700

Signetics

Microprocessor Products

Section 7
Numeric Index

Signetics

Numeric Index

Microprocessor Products

signetics

a subsidiary of U.S. Philips Corporation

Signetics Corporation
811 E. Arques Avenue
P.O. Box 3409
Sunnyvale, California 94088-3409
Telephone 408/991-2000

98-8000-000

Printed in U.S.A. AIS/BA56MCR11285
800 pages