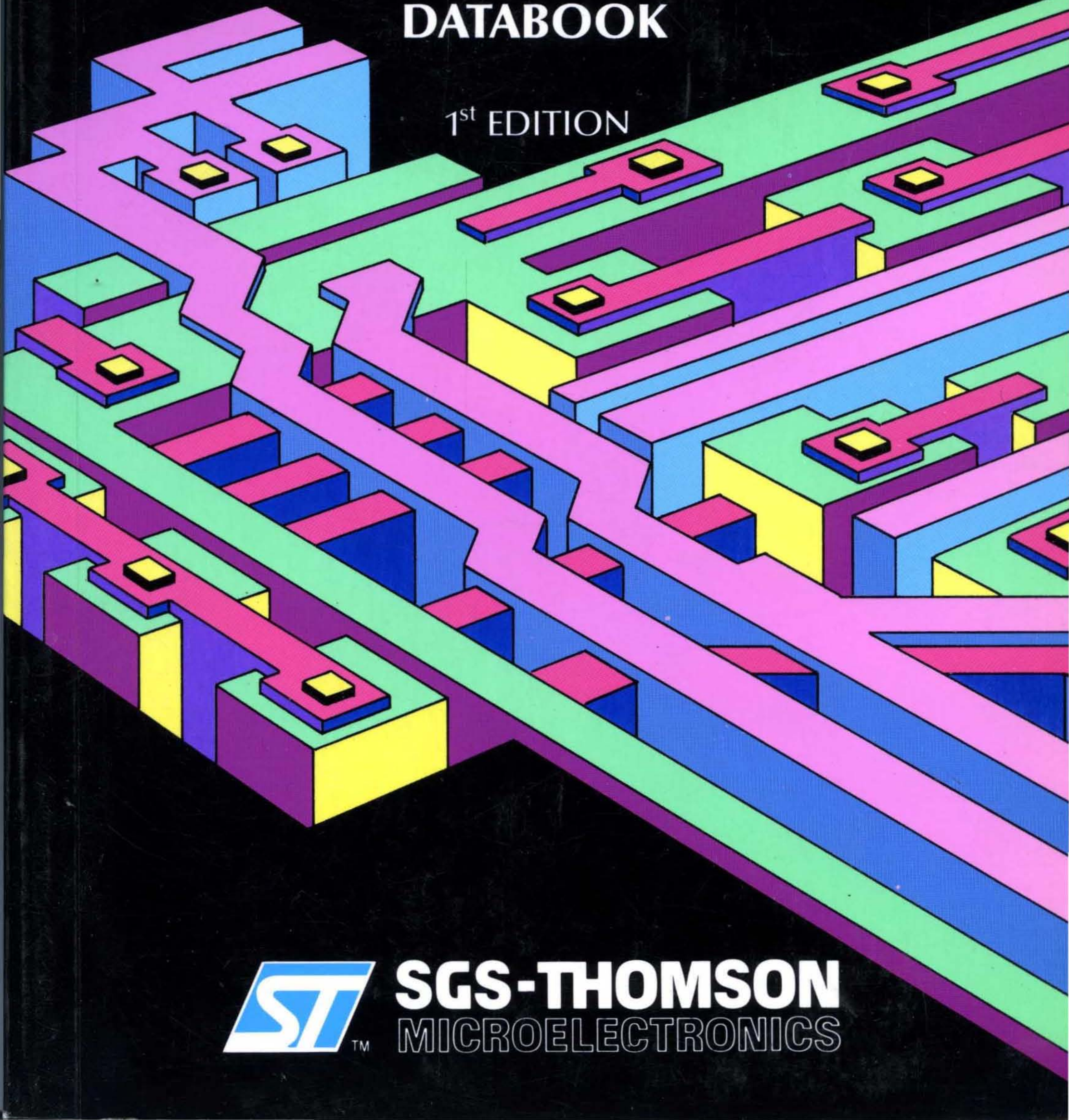
16 BIT MPUs & ASSOCIATED PERIPHERALS

DATABOOK

1st EDITION



SGS-THOMSON
MICROELECTRONICS

16 BIT MPUs & ASSOCIATED PERIPHERALS

DATABOOK

1st EDITION

APRIL 1989

TABLE OF CONTENTS

GENERAL INDEX

PRODUCT GUIDE

Part. Number	Description	Page
TS68000	16/32-Bit Microprocessor, 16M Bytes, Direct Addressing, 56 Powerful Instructions, 14 Addressing Modes	13
TS68008	8-Bit Version of the TS68000, 1M Bytes Direct Addressing, TS68000 Software Compatible	89
TS68230	Parallel Interface Timer, up to 24 I/O Lines, 24-Bit Programmable Timer	173
MK68564	Dual Serial Input Output Controller, Asynchronous Synchronous byte Oriented, Synchronous Bit Oriented Protocol, up to 1.1M Bits per Second	235
MK68901	Multi Function Peripheral, 8 I/O Lines, 16 Interrupt Sources, Single Channel USART, Four 8-Bit Timers	281
TS68HC901	CMOS Version and Fully Compatible with the MK68901	315
Z8038	FIFO Input Output Interface Unit, 128-Byte RAM Buffer, Data Transactions Managing, 12 Operating Modes	363
Z8530	Serial Communication Controller, 2 Independent Channels, Multiprotocol, up to 1M-Bit per Second	449
Z8531	Asynchronous Serial Communication Controller, 2 Independent Channels	543
Z8536	Counter/Timer and Parallel I/O Unit, Two 8-Bit Ports, 4-Bit Special Purpose Port, 16-Vector Interrupt Controller, Three 16-Bit Timers	565

68000 MICROPROCESSORS

SGS-THOMSON MICROELECTRONICS

TS68000

HMOS 16/32–BIT MICROPROCESSOR

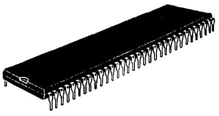
PRELIMINARY INFORMATION

The TS68000 is the first implementation of the 68000 16/32 microprocessor architecture. The TS68000 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 TS68008 8-bit data bus implementation of the 68000 and is downward code-compatible with the TS68020 32-bit implementation of the architecture. Any user-mode programs written using the TS68000 instruction set will run unchanged on the TS68008 and TS68020. This is possible because the user programming model is identical for all three processors and the instruction sets are proper subsets of the complete architecture.

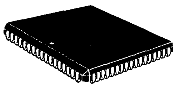
The resources available to the TS68000 user consist of the following :

- 16 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
- 4 available versions : 8MHz, 10MHz, 12.5MHz and 16MHz

As shown in the user programming model, the TS68000 offers 16 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.

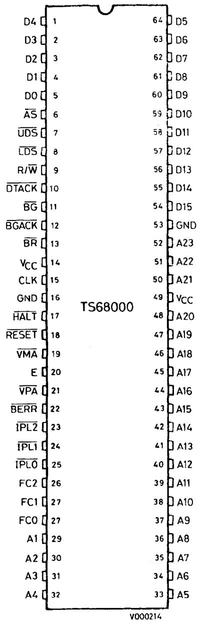


P
(Plastic Package)



FN
(PLCC 68)

PIN CONNECTIONS



SECTION 1

INTRODUCTION

The TS68000 is the first implementation of the 68000 16/32 microprocessor architecture. The TS68000 has a 16-bit data bus and 24-bit address bus while the full architecture provides for 32-bit address an data buses. It is completely code-compatible with the TS68008 8-bit data bus implementation of the 68000 and is downward code-compatible with the TS68020 32-bit implementation of the architecture. Any user-mode programs written using the TS68000 instruction set will run unchanged on the TS68008 and TS68020. This is possible because the user programming model is identical for all four processors and the instruction sets are proper subsets of the complete architecture.

The resources available to the TS68000 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 user programming model (figure 1-1), the TS68000 offers 16 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.

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 1-2.

The status register (figure 1-3) 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.

1.1. 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.

Figure 1.1 : User Programming Model.

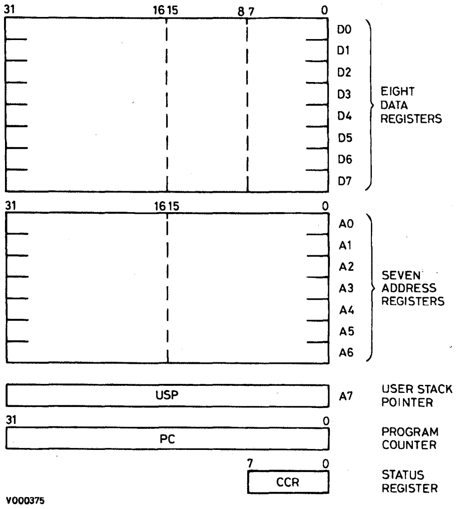


Figure 1.2 : Supervisor Programming Model Supplement.

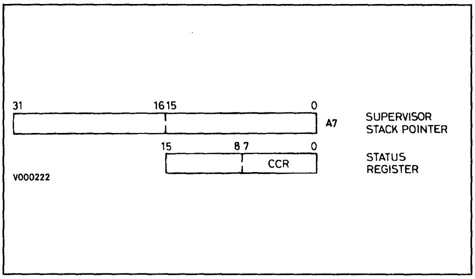


Figure 1.3 : Status Register.

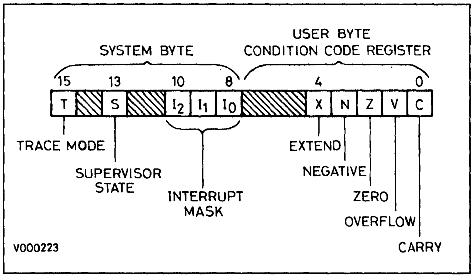


Table 1.1 : Addressing Modes.

Addressing Modes	Syntax
Register Direct Addressing Data Register Direct Address Register Direct	 Dn An
Absolute Data Addressing Absolute Short Absolute Long	 xxx W xxx L
Program Counter Relative Addressing Relative with Offset Relative with Index Offset	 d_{16}(PC) d_8(PC, Xn)
Register Indirect Addressing Register Indirect Postincrement Register Indirect Predecrement Register Indirect Register Indirect with Offset Indexed Register Indirect with Offset	 (An) (An) + – (An) d_{16}(An) d_8(An, Xn)
Immediate Data Addressing Immediate Quick Immediate	 #xxx #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

The 14 address modes, shown in table 1.1, 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.

1.2. INSTRUCTION SET OVERVIEW

The TS68000 instruction set is shown in table 1-2. Some additional instructions are variations, or subsets, of these and they appear in table 1-3. 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 1.2 : Instruction Set Summary.

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
Bcc	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
DBcc	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 Left
MOVE	Move
MULS	Signed Multiply
MULU	Unsigned Mulitply
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
Scc	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

Table 1.3 : 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
	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

SECTION 2

DATA ORGANIZATION AND ADDRESSING CAPABILITIES

This section contains a description of the registers and the data organization of the TS68000.

2.1. 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.

2.2. 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.

2.2.1. 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.

2.2.2. 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 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.

2.3. 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 2.1. 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 TS68000 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 2.2. The numbers indicate the order in which the data would be accessed from the processor.

2.4. ADDRESSING

Instructions for the TS68000 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 2.1 : Word Organization in Memory.

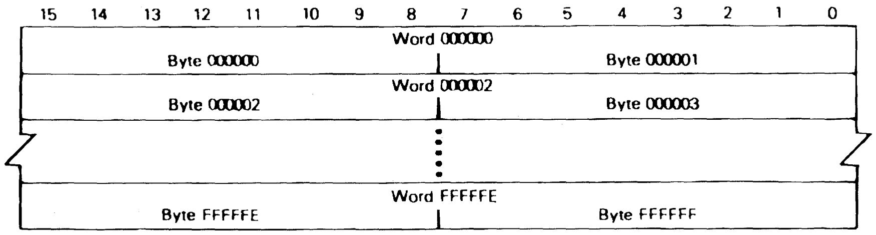
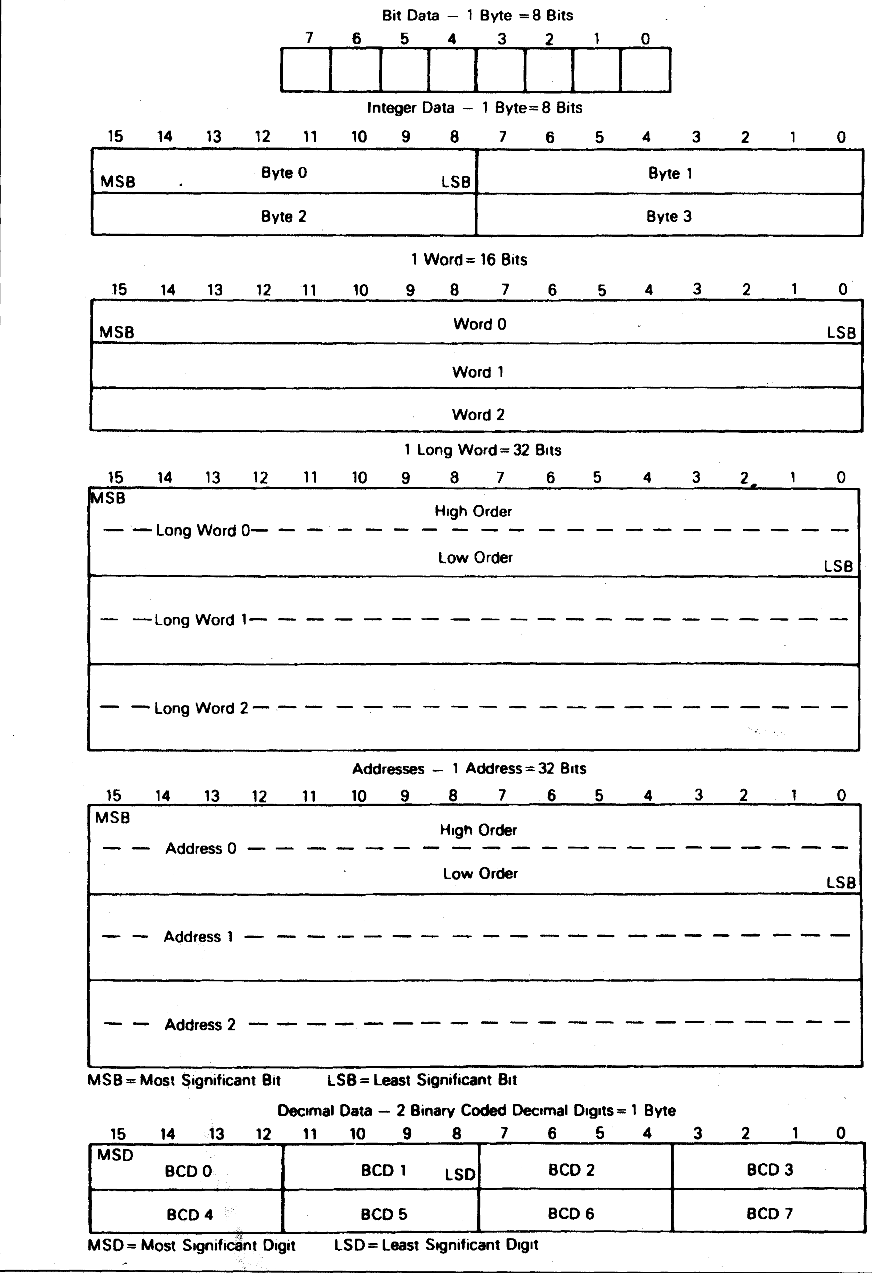


Figure 2.2 : Memory Data Organization.



2.5. INSTRUCTION FORMAT

Instructions are from one to five words in length as shown in figure 2.3. 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.

2.6. PROGRAM/DATA REFERENCES

The TS68000 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. Operands 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.

2.7. 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.

2.8. EFFECTIVE ADDRESS

Most instructions specify the location of an operand by using the effective address field in the operation word. For example, figure 2.4 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 2.3. The effective address modes are grouped into three categories : register direct, memory addressing, and special.

2.8.1. REGISTER DIRECT MODES. These effective addressing modes specify that the operand is in one of 16 multifunction registers.

2.8.1.1. Data Register Direct.

The operand is in the data register specified by the effective address register field.

2.8.1.2. Address Register Direct.

The operand is in the address register specified by the effective address register field.

2.8.2. MEMORY ADDRESS MODES. These effective addressing modes specify that the operand is in memory and provide the specific address of the operand.

2.8.2.1. 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.

Figure 2.3 : Instruction Operation Word General Format.

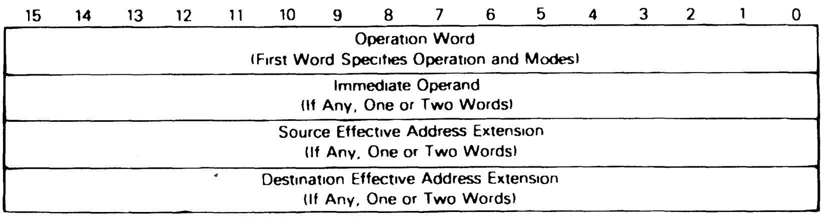
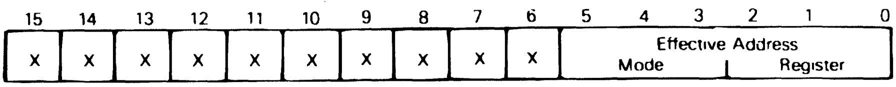

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Operation Word
(First Word Specifies Operation and Modes)
Immediate Operand
(If Any, One or Two Words)
Source Effective Address Extension
(If Any, One or Two Words)
Destination Effective Address Extension
(If Any, One or Two Words)

Figure 2.4 : Single–Effective–Address Instruction Operation Word.



2.8. EFFECTIVE ADDRESS (continued)

2.8.2.2. 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 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.

2.8.2.3. 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.

2.8.2.4. 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.

2.8.2.5. 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.

2.8.3. SPECIAL ADDRESS MODES. The special address modes use the effective address register field to specify the special addressing mode instead of a register number.

2.8.3.1. 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.

2.8.3.2. 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.

2.8.3.3. 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.

2.8.3.4. 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.

2.8.3.5. 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.

2.8.3.6. 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	EORI to SR	MOVE to CCR
ANDI to SR	ORI to CCR	MOVE to SR
EORI to CCR	ORI to SR	MOVE from SR

2.9. EFFECTIVE ADDRESS ENCODING SUMMARY

Table 2.1 is a summary of the effective addressing modes discussed in the previous paragraphs.

2.10. 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 2.1 : 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

SECTION 3

INSTRUCTION SET SUMMARY

This section contains an overview of the form and structure of the TS68000 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 Coded 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.

3.1. 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 3.1 is a summary of the data movement operations.

Table 3.1 : Data Movement Operations.

Instruction	Operand Size	Operation
EXG	32	$R_X \leftrightarrow R_Y$
LEA	32	EA ← An
LINK	–	An → (SP) SP → An SP + Displacement → SP
MOVE	8, 16, 32	s → 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

3.2. 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 3.2 is a summary of the integer arithmetic operations.

Table 3.2 : 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, 26, 32 16, 32	Dx + Dy + X → Dx – (Ax) + – (Ay)+ X →, (Ax)
CLR	8, 16, 32	O → 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	O – (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
– ( ) = indirect with predecrement
( ) + = indirect with postdecrement
# = immediate data

3.3. 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 3.3 is a summary of the logical operations.

Table 3.3 : Logical Operations.

Instruction	Operand Size	Operation
AND	8, 16, 32	DnΛ(EA) → Dn (EA)ΛDn → (EA) (EA)Λ#xxx → (EA)
OR	8, 16, 32	Dn v (EA) → Dn (EA) v Dn → (EA) (EA) v #xxx → (EA)
EOR	8, 16, 32	(EA) ⊕ Dy → (EA) (EA) ⊕ #xxx → (EA)
NOT	8, 16, 32	~ (EA) → (EA)

Notes : ~ = invert
# = immediate data
Λ = logical AND
V = logical OR
⊕ = logical exclusive OR

3.4. 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 3.4 is a summary of the shift and rotate operations.

Table 3.4 : Shift and Rotate Operations.

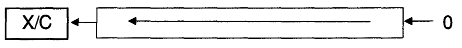
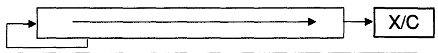
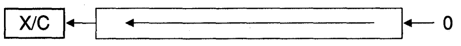
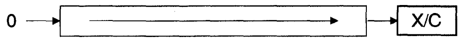
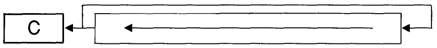
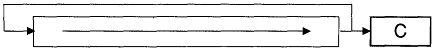
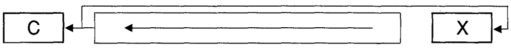
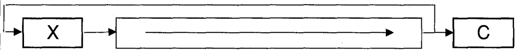
Instruction	Operand Size	Operation
ASL	8, 16, 32	
ASR	8, 16, 32	
LSL	8, 16, 32	
LSR	8, 16, 32	

Table 3.4 : Shift and Rotate Operations (continued).

Instruction	Operand Size	Operation
ROL	8, 16, 32	
ROR	8, 16, 32	
ROXL	8, 16, 32	
ROXR	8, 16, 32	

3.5. 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 3-5 is a summary of the bit manipulation operations (Z is bit 2 of the status register).

Table 3.5 : 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

3.6. 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 3.6 is a summary of the binary coded decimal operations.

Table 3.6 : 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

3.7. 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 3.7.

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 3.7 : 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 condtions)
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

3.8. 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 3.8.

Table 3.8 : 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	Chek 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 Condtion Codes
MOVE EA to CCR	Load New Condition Codes
MOVE SR to EA	Store Status Register
ORI to CCR	Logical OR to Condition Codes

SECTION 4

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.

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.*

4.1. SIGNAL DESCRIPTION

The input and output signals can be functionally organized into the groups shown in figure 4-1. 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.

4.1.1. 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.

4.1.2. 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.

4.1.3. 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.

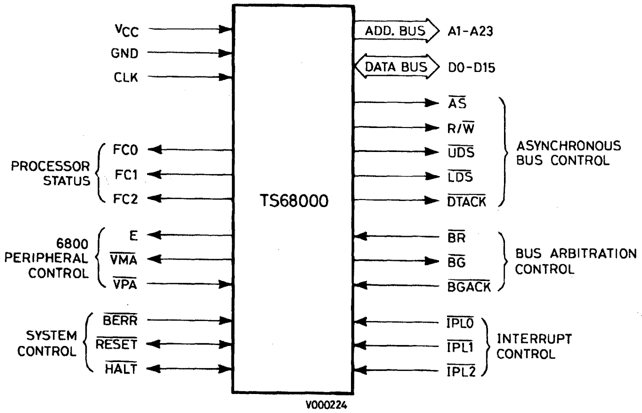
4.1.3.1. Address Strobe ($\overline{AS}$).

This signal indicates that there is a valid address on the address bus.

4.1.3.2. 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.

Figure 4.1 : Input and Output Signals.



4.1.3.3. Upper and Lower Data Strobe (UDS, LDS).

These signals control the flow of data on the data bus, as shown in table 4-1. When the R/W line is high, the processor will read from the data bus as indicated. When the R/W line is low, the processor will write to the data bus as shown.

4.1.3.4. Data Transfer Acknowledge ($\overline{\text{DTACK}}$).

This input indicates that the data transfer is completed. When the processor recognizes $\overline{\text{DTACK}}$ during a read cycle, data is latched and the bus cycle terminated. When $\overline{\text{DTACK}}$ is recognized during a write cycle, the bus cycle is terminated. (Refer to 4.4 Asynchronous Versus Synchronous Operation).

4.1.4. 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.

4.1.4.1. Bus Request ($\overline{\text{BR}}$).

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.

4.1.4.2. Bus Grant ($\overline{\text{BG}}$).

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.

4.1.4.3. Bus Grant Acknowledge ($\overline{\text{BGACK}}$).

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.

4.1.5. INTERRUPT CONTROL ($\overline{\text{IPL0}}$, $\overline{\text{IPL1}}$, $\overline{\text{IPL2}}$). 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 $\overline{\text{IPL0}}$ and the most significant bit is contained in $\overline{\text{IPL2}}$. These lines must remain stable until the processor signals interrupt acknowledge (FC0-FC2 are all high) to insure that the interrupt is recognized.

4.1.6. 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.

Table 4.1 : Data Strobe Control of Data Bus.

$\overline{\text{UDS}}$	$\overline{\text{LDS}}$	R/$\overline{\text{W}}$	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.

4.1.6.1. Bus Error ($\overline{\text{BERR}}$)

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 **4.2.4. Bus Error and Halt Operation** for additional information about the interaction of the bus error and halt signals.

4.1.6.2. Reset ($\overline{\text{RESET}}$)

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 $\overline{\text{RESET}}$ 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 $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$ signals applied at the same time. Refer to **4.2.5. Reset Operation** for further information.

4.1.6.3. Halt ($\overline{\text{HALT}}$)

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 4.3). Refer to **4.2.4. Bus Error and Halt Operation** for additional information about the interaction between the $\overline{\text{HALT}}$ and bus error signals.

When the processor has stopped executing instructions, such as in a double bus fault condition (refer to **4.2.4.4. Double Bus Faults**), the $\overline{\text{HALT}}$ line is driven by the processor to indicate to external devices that the processor has stopped.

4.1.7. EF6800 PERIPHERAL CONTROL. These control signals are used to allow the interfacing of synchronous EF6800 peripheral devices with the asynchronous TS68000. These signals are explained in the following paragraphs.

4.1.7.1. Enable (E)

This signal is the standard enable signal common to all EF6800 type peripheral devices. The period for this output is ten TS68000 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.

4.1.7.2. Valid Peripheral Address ($\overline{\text{VPA}}$)

This input indicates that the device or region addressed is an EF6800 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 Section 6 Interface With Ef6800 Peripherals.

4.1.7.3. Valid Memory Address ($\overline{\text{VMA}}$)

This output is used to indicate to EF6800 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 ($\overline{\text{VPA}}$) input which indicates that the peripheral is an EF6800 Family device.

4.1.8. 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 4.2. The information indicated by the function code outputs is valid whenever address strobe ($\overline{\text{AS}}$) is active.

Table 4.2 : 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

4.1.9. 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.

4.1.10. SIGNAL SUMMARY. Table 4.3 is a summary of all the signals discussed in the previous paragraphs.

4.2. BUS OPERATION

The following paragraphs explain control signal and bus operation during data transfers operations, bus arbitration, bus error and hait conditions, and reset operation.

4.2.1. 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 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 TS68000 for interlocked multiprocessor communications.

Table 4.3 : Signal Summary.

Signal Name	Nmemonic	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	Yes
Clock	CLK	Input	High	No	No
Power Input	V_{CC}	Input	–	–	–
Ground	GND	Input	–	–	–

Note : 1. Open drain

4.2.1.1. 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 4.2. A byte read cycle flowchart is given in figure 4.3. Read cycle timing is given in figure 4.4. Figure 4.5 details word and byte read cycle operations.

Figure 4.2 : Word Read Cycle Flowchart.

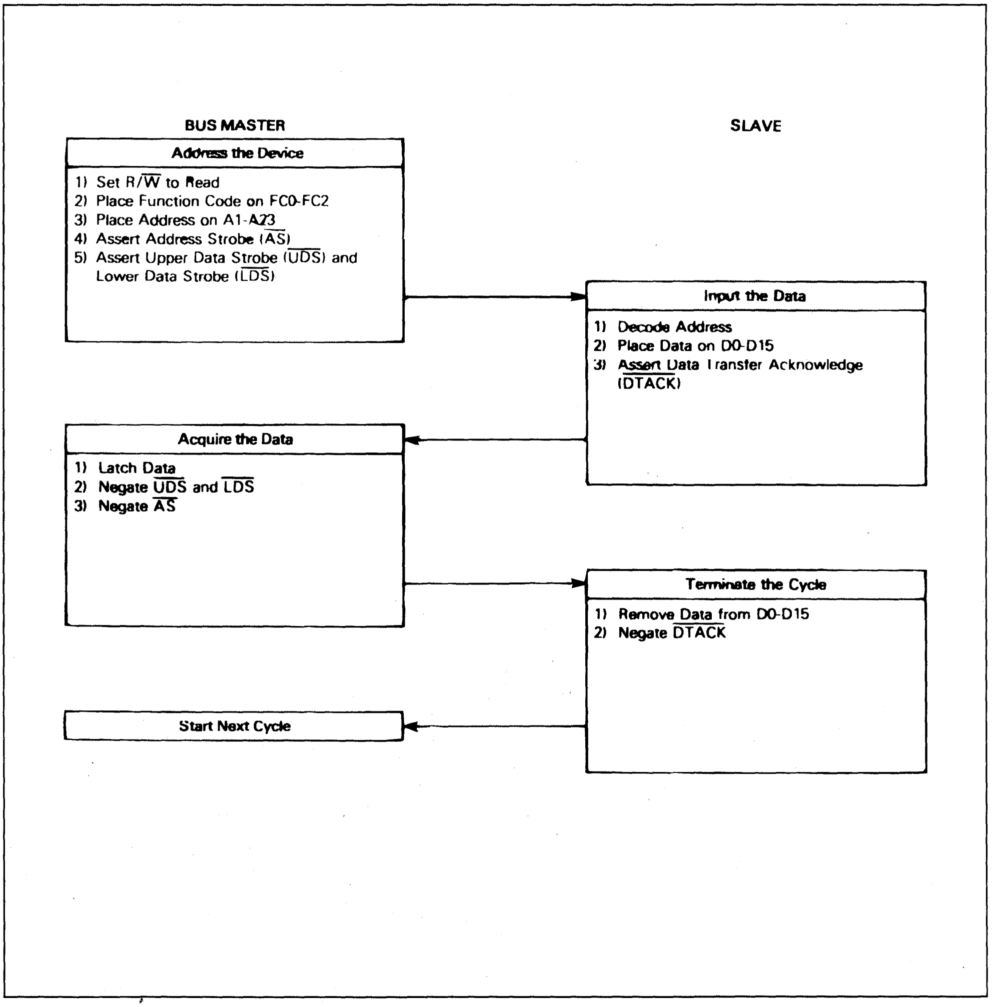

BUS MASTER
SLAVE
Address the Device
1) Set R/$\overline{W}$ to Read
2) Place Function Code on FC0-FC2
3) Place Address on A1-A23
4) Assert Address Strobe ($\overline{AS}$)
5) Assert Upper Data Strobe ($\overline{UDS}$) and Lower Data Strobe ($\overline{LDS}$)
Input the Data
1) Decode Address
2) Place Data on D0-D15
3) Assert Data Transfer Acknowledge ($\overline{DTACK}$)
Acquire the Data
1) Latch Data
2) Negate $\overline{UDS}$ and $\overline{LDS}$
3) Negate $\overline{AS}$
Terminate the Cycle
1) Remove Data from D0-D15
2) Negate $\overline{DTACK}$
Start Next Cycle

Figure 4.3 : Byte Read Cycle Flowchart.

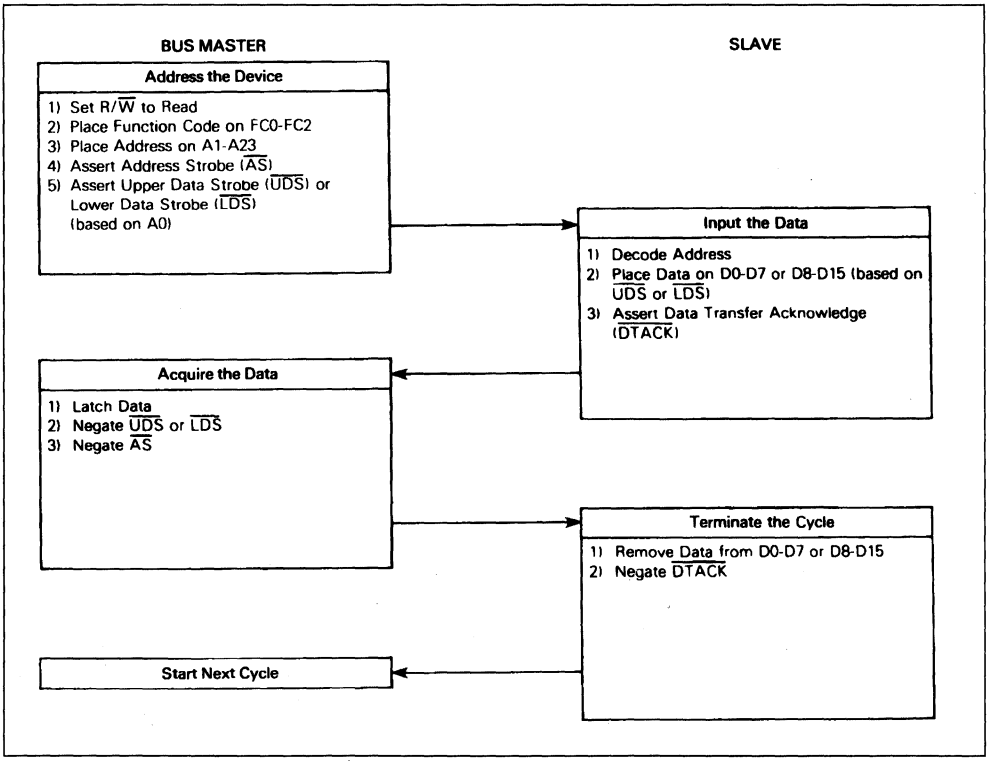

BUS MASTER
SLAVE
Address the Device
1) Set $R/\overline{W}$ to Read
2) Place Function Code on FC0-FC2
3) Place Address on A1-A23
4) Assert Address Strobe ($\overline{AS}$)
5) Assert Upper Data Strobe ($\overline{UDS}$) or Lower Data Strobe ($\overline{LDS}$) (based on A0)
Input the Data
1) Decode Address
2) Place Data on D0-D7 or D8-D15 (based on $\overline{UDS}$ or $\overline{LDS}$)
3) Assert Data Transfer Acknowledge ($\overline{DTACK}$)
Acquire the Data
1) Latch Data
2) Negate $\overline{UDS}$ or $\overline{LDS}$
3) Negate $\overline{AS}$
Terminate the Cycle
1) Remove Data from D0-D7 or D8-D15
2) Negate $\overline{DTACK}$
Start Next Cycle

Figure 4.4 : Read and Write Cycle Timing Diagram.

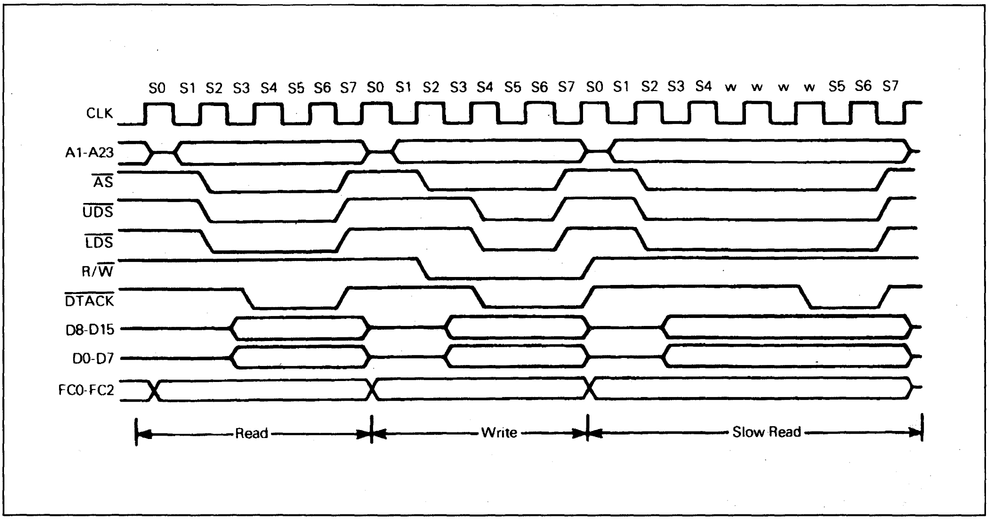
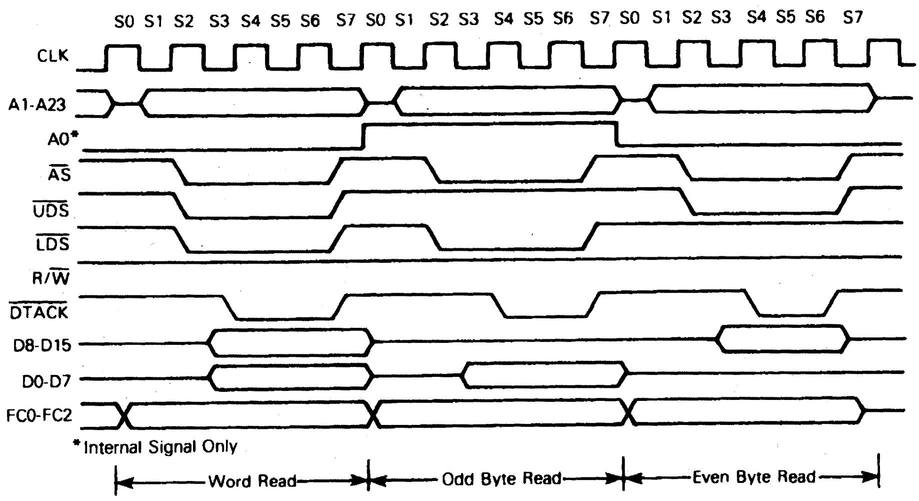


Figure 4.5 : Word and Byte Read Cycle Timing Diagram.



4.2.1.2 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 4.6. A byte write cycle flowchart is given in figure 4.7. Write cycle timing is given in figure 4.4. Figure 4.8 details word and byte write cycle operation.

Figure 4.6 : Word Write Cycle Flowchart

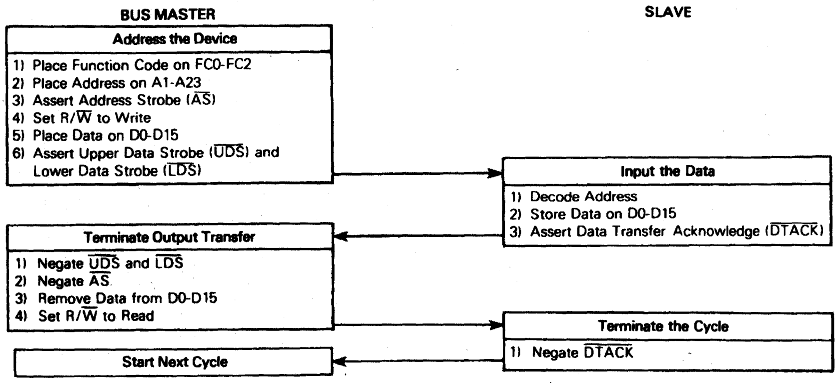

BUS MASTER
SLAVE
Address the Device
1) Place Function Code on FC0-FC2
2) Place Address on A1-A23
3) Assert Address Strobe (AS)
4) Set R/W to Write
5) Place Data on D0-D15
6) Assert Upper Data Strobe (UDS) and Lower Data Strobe (LDS)
Input the Data
1) Decode Address
2) Store Data on D0-D15
3) Assert Data Transfer Acknowledge (DTACK)
Terminate Output Transfer
1) Negate UDS and LDS
2) Negate AS
3) Remove Data from D0-D15
4) Set R/W to Read
Terminate the Cycle
1) Negate DTACK
Start Next Cycle

Figure 4.7 : Byte Write Cycle Flowchart.

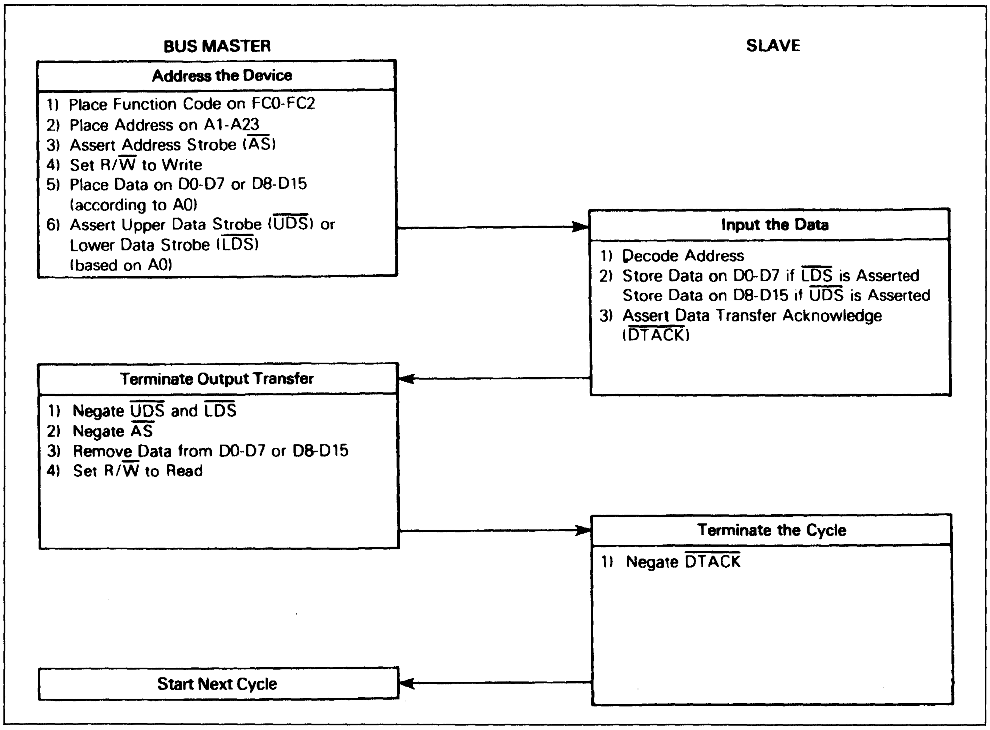
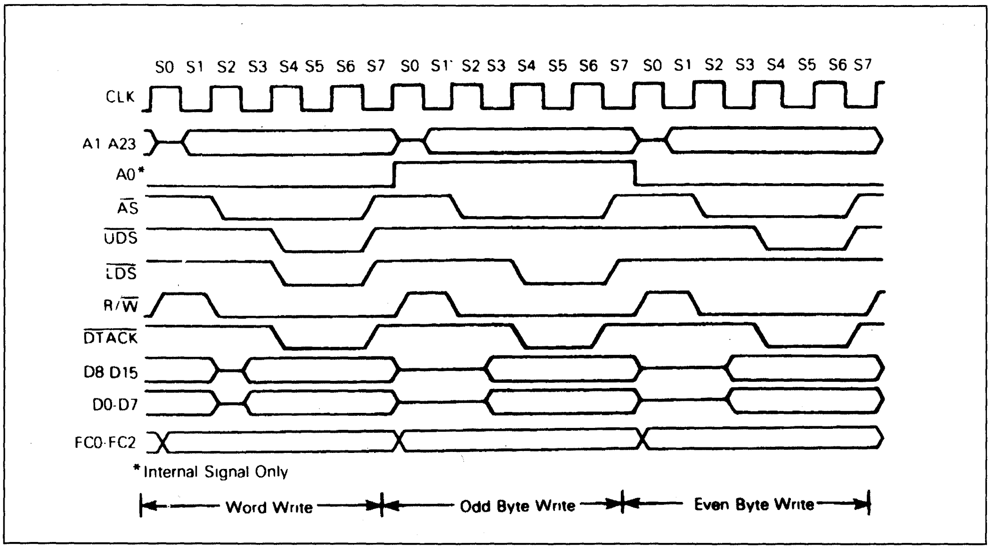

BUS MASTER
SLAVE
Address the Device
1) Place Function Code on FC0-FC2
2) Place Address on A1-A23
3) Assert Address Strobe ($\overline{AS}$)
4) Set R/$\overline{W}$ to Write
5) Place Data on D0-D7 or D8-D15 (according to A0)
6) Assert Upper Data Strobe ($\overline{UDS}$) or Lower Data Strobe ($\overline{LDS}$) (based on A0)
Input the Data
1) Decode Address
2) Store Data on D0-D7 if $\overline{LDS}$ is Asserted
Store Data on D8-D15 if $\overline{UDS}$ is Asserted
3) Assert Data Transfer Acknowledge ($\overline{DTACK}$)
Terminate Output Transfer
1) Negate $\overline{UDS}$ and $\overline{LDS}$
2) Negate $\overline{AS}$
3) Remove Data from D0-D7 or D8-D15
4) Set R/$\overline{W}$ to Read
Terminate the Cycle
1) Negate $\overline{DTACK}$
Start Next Cycle

Figure 4.8 : Word and Byte Write Cycle Timing Diagram.



4.2.1.3. 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 TS68000, 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 flowchart is given in figure 4.9 and a timing diagram is given in figure 4.10.

Figure 4.9 : Read–Modify–Write Cycle Flowchart.

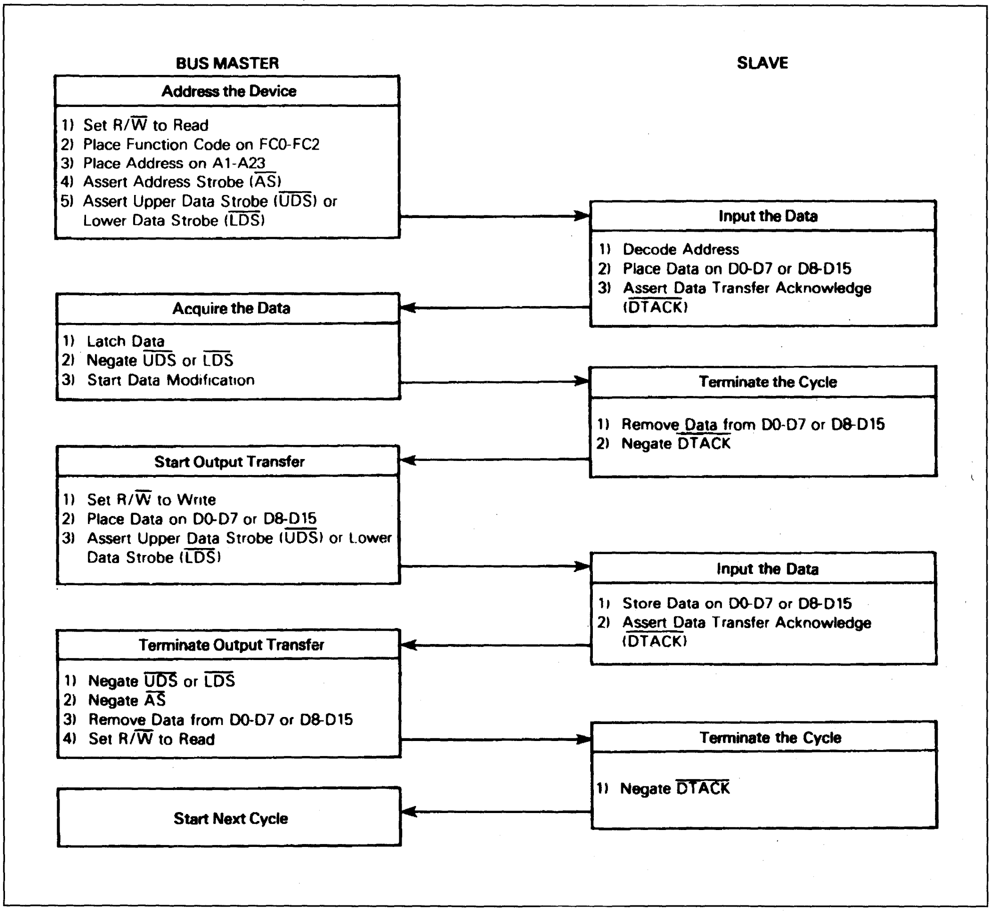
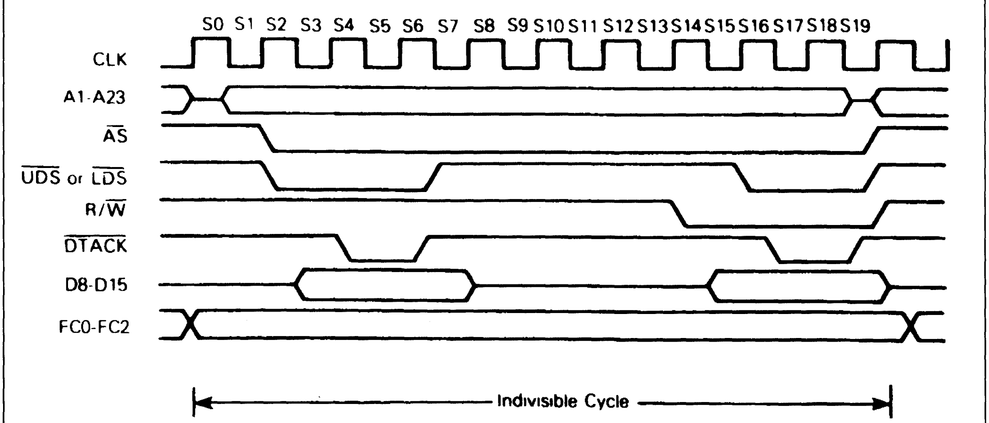

BUS MASTER
SLAVE
Address the Device
1) Set R/$\overline{W}$ to Read
2) Place Function Code on FC0-FC2
3) Place Address on A1-A23
4) Assert Address Strobe ($\overline{AS}$)
5) Assert Upper Data Strobe ($\overline{UDS}$) or Lower Data Strobe ($\overline{LDS}$)
Input the Data
1) Decode Address
2) Place Data on D0-D7 or D8-D15
3) Assert Data Transfer Acknowledge ($\overline{DTACK}$)
Acquire the Data
1) Latch Data
2) Negate $\overline{UDS}$ or $\overline{LDS}$
3) Start Data Modification
Terminate the Cycle
1) Remove Data from D0-D7 or D8-D15
2) Negate $\overline{DTACK}$
Start Output Transfer
1) Set R/$\overline{W}$ to Write
2) Place Data on D0-D7 or D8-D15
3) Assert Upper Data Strobe ($\overline{UDS}$) or Lower Data Strobe ($\overline{LDS}$)
Input the Data
1) Store Data on D0-D7 or D8-D15
2) Assert Data Transfer Acknowledge ($\overline{DTACK}$)
Terminate Output Transfer
1) Negate $\overline{UDS}$ or $\overline{LDS}$
2) Negate $\overline{AS}$
3) Remove Data from D0-D7 or D8-D15
4) Set R/$\overline{W}$ to Read
Terminate the Cycle
1) Negate $\overline{DTACK}$
Start Next Cycle

Figure 4.10 : Read–Modify–Write Cycle Timing Diagram.


S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19
CLK
A1-A23
$\overline{AS}$
$\overline{UDS}$ or $\overline{LDS}$
R/$\overline{W}$
$\overline{DTACK}$
D8-D15
FC0-FC2
Indivisible Cycle

4.2.2. 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 4.11 is a flowchart showing the detail involved in a request from a single device. Figure 4.12 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 4.11 : Bus Arbitration Cycle Flowchart.

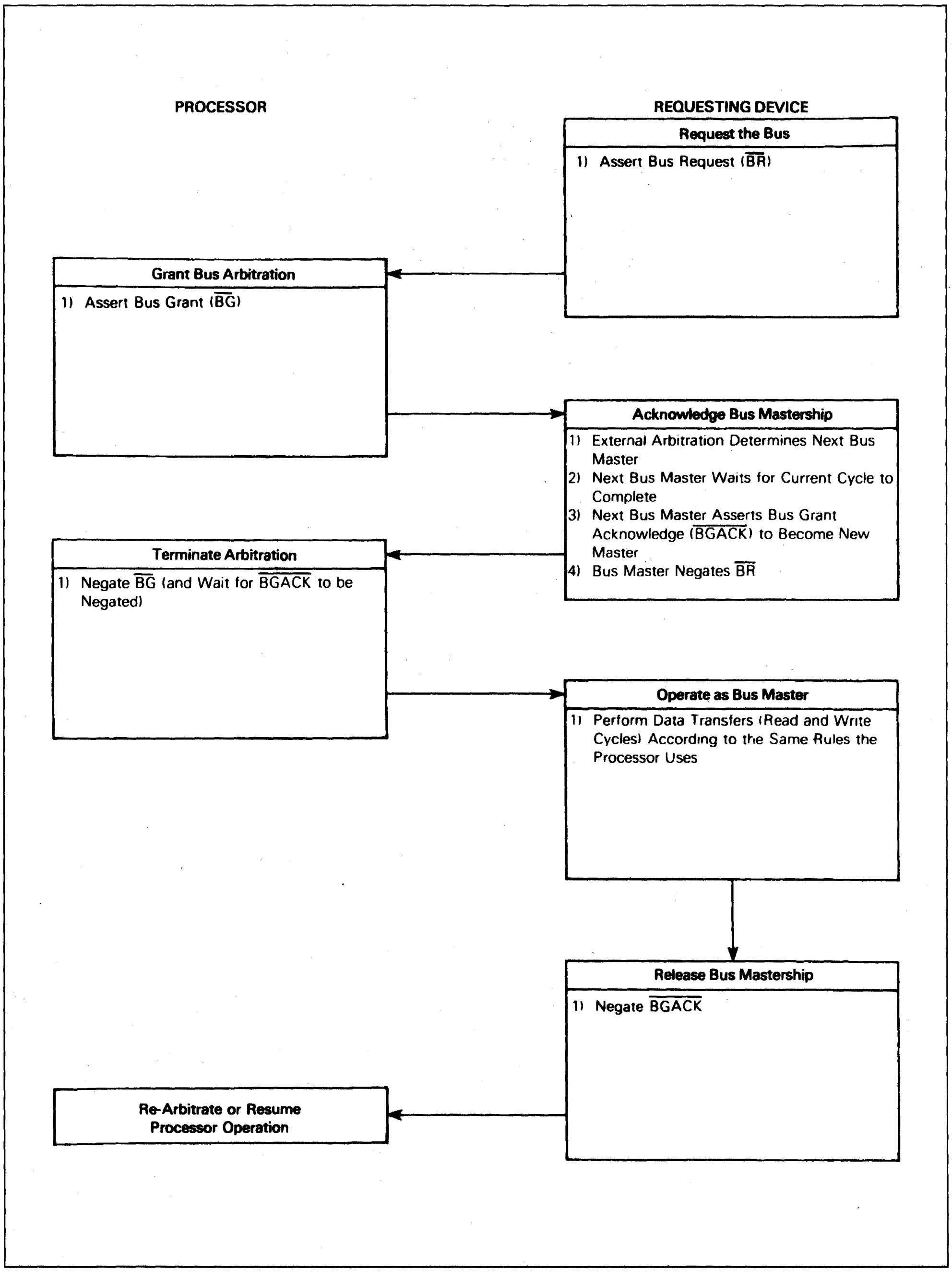
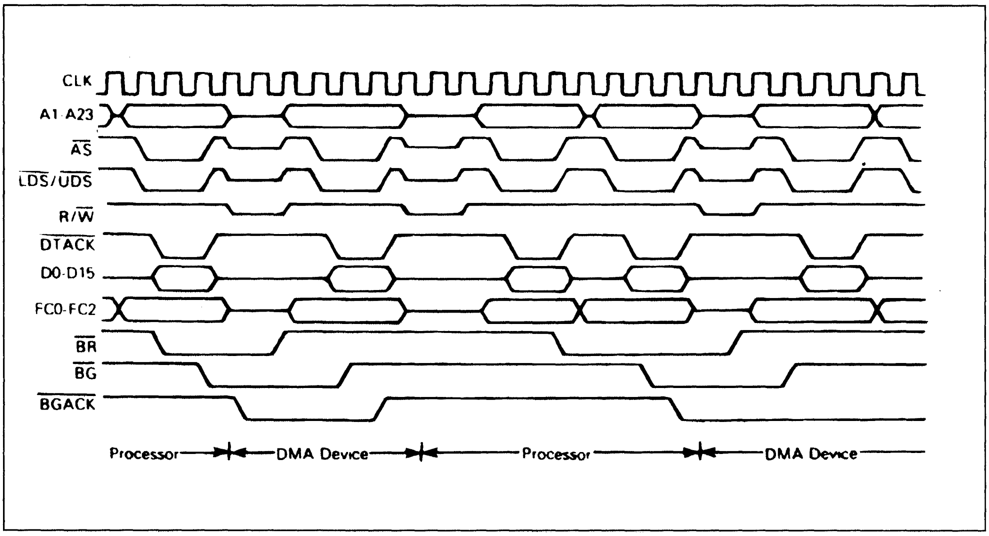

PROCESSOR
REQUESTING DEVICE
Request the Bus
1) Assert Bus Request ($\overline{BR}$)
Grant Bus Arbitration
1) Assert Bus Grant ($\overline{BG}$)
Acknowledge Bus Mastership
1) External Arbitration Determines Next Bus Master
2) Next Bus Master Waits for Current Cycle to Complete
3) Next Bus Master Asserts Bus Grant Acknowledge ($\overline{BGACK}$) to Become New Master
4) Bus Master Negates $\overline{BR}$
Terminate Arbitration
1) Negate $\overline{BG}$ (and Wait for $\overline{BGACK}$ to be Negated)
Operate as Bus Master
1) Perform Data Transfers (Read and Write Cycles) According to the Same Rules the Processor Uses
Release Bus Mastership
1) Negate $\overline{BGACK}$
Re-Arbitrate or Resume Processor Operation

Figure 4.12 : Bus Arbitration Cycle Timing Diagram.



4.2.2.1. Requesting the Bus

External devices capable of becoming bus masters request the bus by asserting the bus request (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.

4.2.2.2 Receiving the Bus Grant

The processor asserts bus grant (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 (AS) signal. In this case, bus grant will be delayed until 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.

4.2.2.3. 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 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 **4.2.3. Bus Arbitration Control**. Note that the processor does not perform any external bus cycles before it re-asserts bus grant.

4.2.3. BUS ARBITRATION CONTROL. The bus arbitration control unit in the TS68000 is implemented with a finite state machine. A state diagram of this machine is shown in figure 4.13. All asynchronous signals to the TS68000 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 setup time (#47) has been met (see figure 4.14). The input signal is sampled on the falling edge of the clock and is valid internally after the next falling edge.

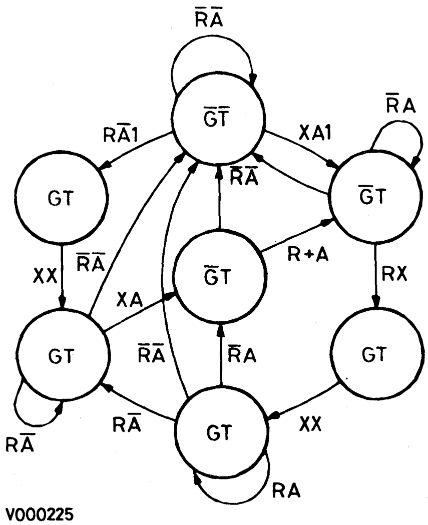
As shown in figure 4.13, input signals labeled R and A are internally synchronized on the bus request and bus grant acknowldege pins respectively. The bus grant output is labeled G and the internal three-state control signal T. If T is true, the address, data, and control buses are placed in a high-impedance 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 4.15. The bus arbitration sequence while the bus is inactive (i.e., executing internal operations such as a multiply instruction) is show in figure 4.16.

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 4.17.

4.2.4. 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 4.13 : TS68000 Bus Arbitration Unit State Diagram.


$\overline{R}\overline{A}$
$\overline{G}\overline{T}$
$R\overline{A}1$
XA1
$\overline{R}A$
GT
$\overline{R}\overline{A}$
$\overline{G}T$
XX
$\overline{R}\overline{A}$
$\overline{G}T$
R+A
RX
XA
GT
$\overline{R}\overline{A}$
$\overline{R}A$
GT
$R\overline{A}$
$R\overline{A}$
GT
XX
RA

V000225

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

Notes : 1. State machine will not change if the bus is S0 or S1. Refer to 4.2.3. Bus Arbitration Control.
2. The address bus will be placed in the high-impedance state if T is asserted and AS is negated.

Figure 4.14 : Timing Relationship of External Asynchronous Inputs to Internal Signals.

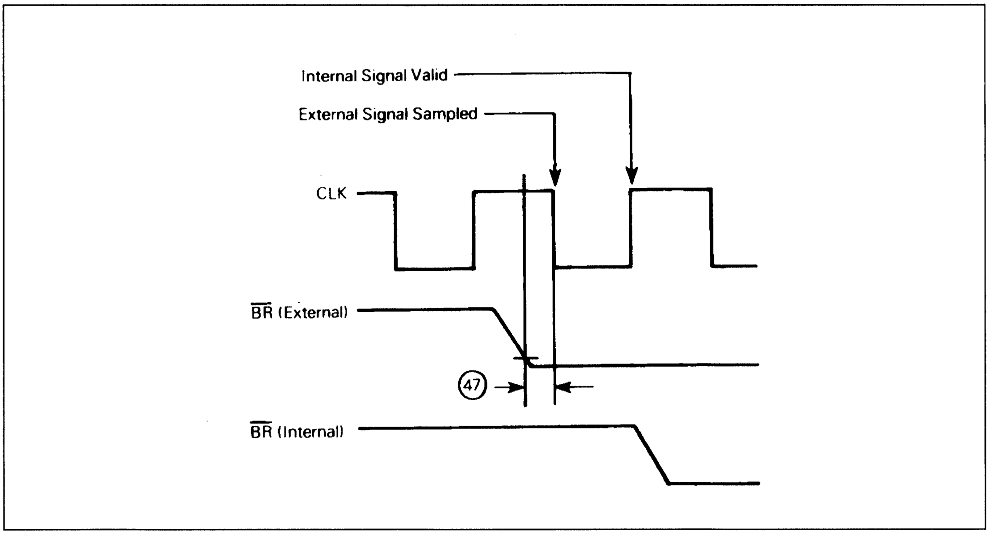


Figure 4.15 : Bus Arbitration Timing Diagram – Processor Active.

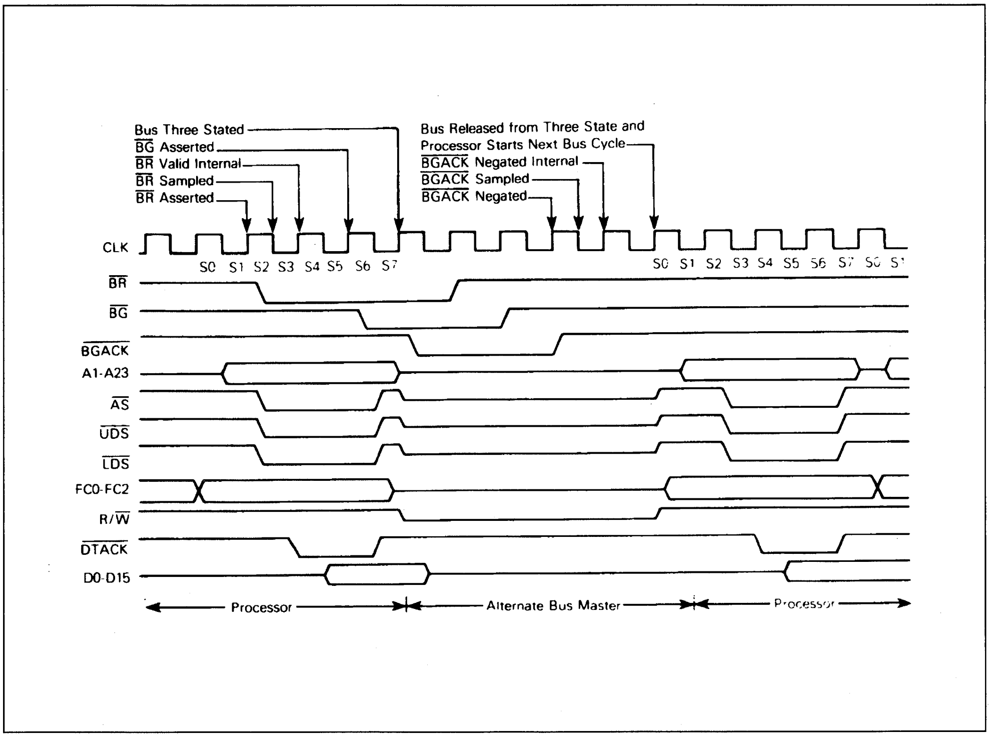


Figure 4.16 : Bus Arbitration Timing Diagram – Bus Inactive.

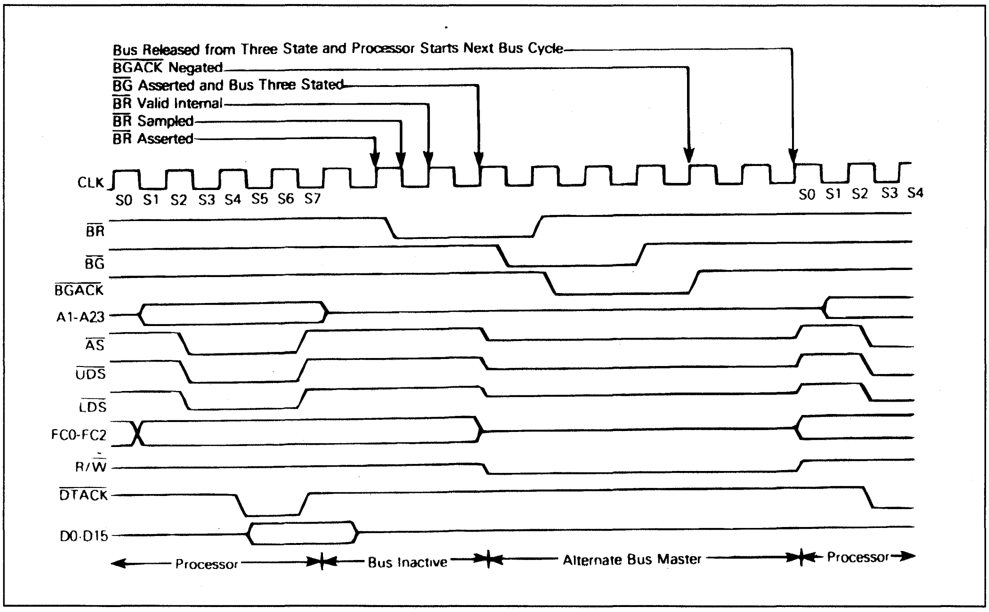
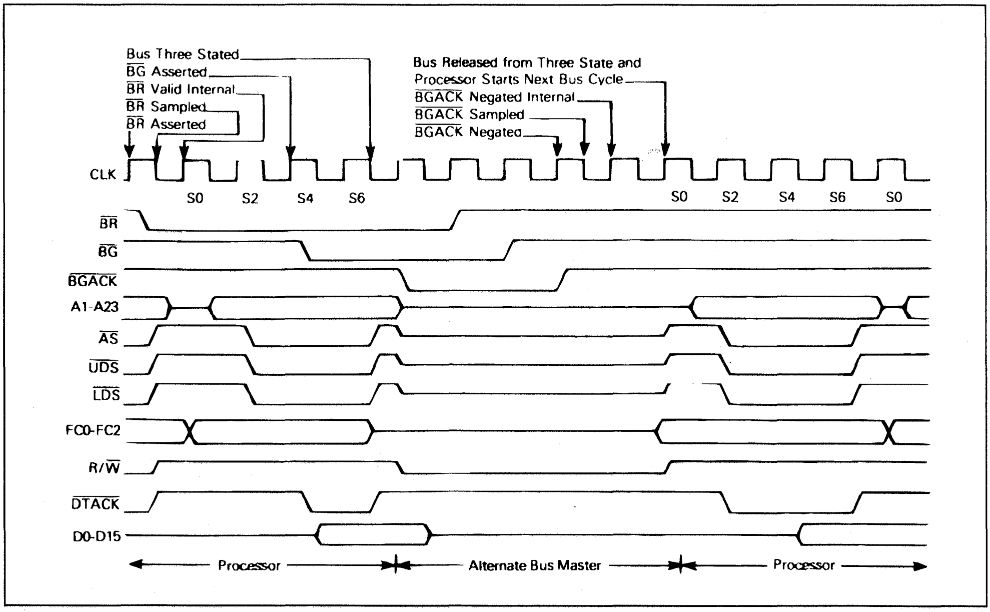


Figure 4.17 : Bus Arbitration Timing Diagram – Special Case.



4.2.4.1. 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 4.18 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 **5.2. Exception Processing** for additional information.

4.2.4.2. 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 4.19 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.

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.

Figure 4.18 : Bus Error Timing Diagram.

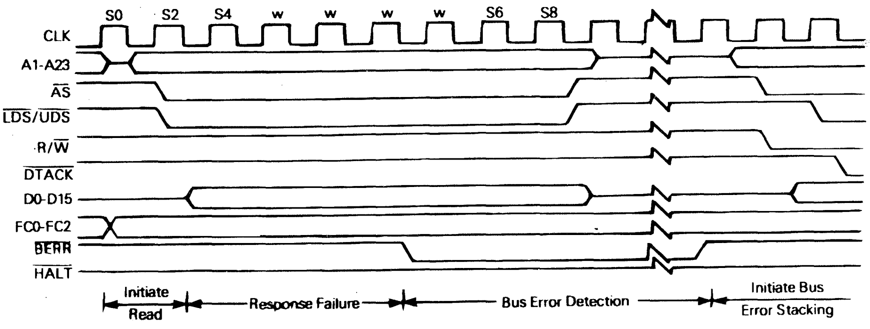
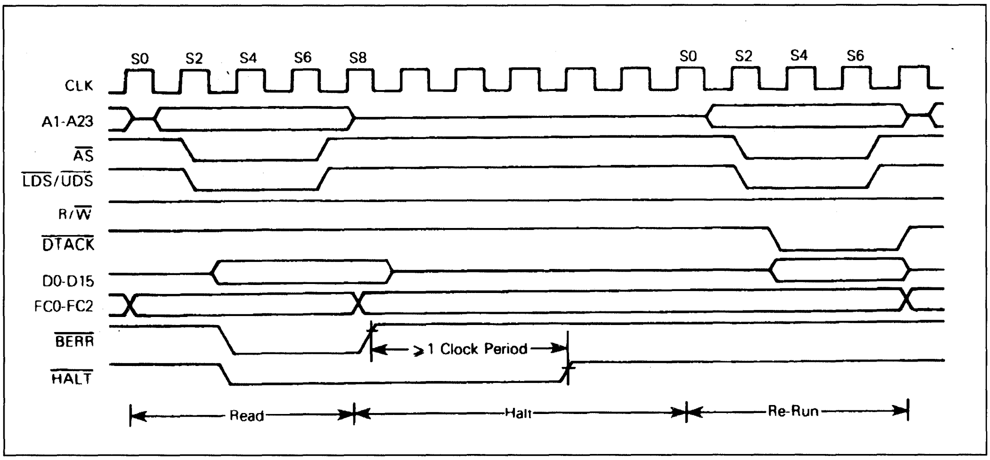


Figure 4.19 : Re–Run Bus Cycle Timing Diagram.



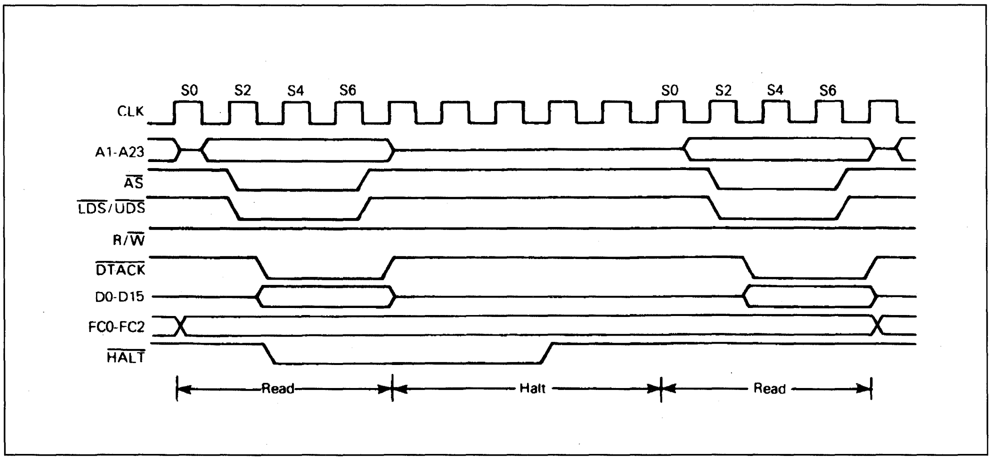
4.2.4.3. Halt Operation

The halt input signal to the TS68000 performs a halt/run/single-step function in a similar fashion to the EF6800 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. If 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 4.20 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 4.20 : Halt Processor Timing Diagram.



SGS-THOMSON MICROELECTRONICS

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.

4.2.4.4. 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.

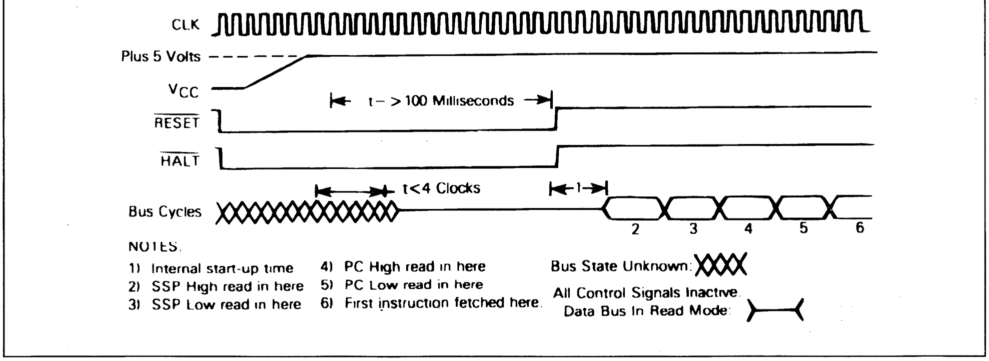
4.2.5. RESET OPERATION. The reset signal is a bidirectional signal that allows either the processor or an external signal to reset the system. Figure 4.21 is a timing diagram for the reset operation. Both the halt and reset lines must be asserted to ensure total reset of the processor.

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 4.21 : Reset Operation Timing Diagram.



4.3. THE RELATIONSHIP OF DTACK, BERR, AND HALT

In order to properly control termination of a bus cycle for a re-run or a bus error condition, DTACK, BERR, and HALT should be asserted and negated on the rising edge of the TS68000 clock. This will assure that when two signals are asserted simultaneously, the required setup 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 TS68000. Parameter #48 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 4.4) :

Normal Termination :	DTACK occurs first (case 1).
Halt Termination :	HALT is asserted at the same time or before DTACK and BERR remains negated (cases 2 and 3).
Bus Error Termination	BERR is asserted in lieu of, at the same time, or before DTACK (case 4) ; BERR is negated at the same time or after DTACK.
Re-Run Termination :	HALT and BERR are asserted in lieu of, at the same time, or before DTACK (cases 6 and 7) ; HALT must be held at least one cycle after BERR. Case 5 indicates BERR may precede HALT on all mask sets which allows fully asynchronous assertion.

Table 4.4 : DTACK, BERR and HALT Assertion Results.

Case N°	Control Signal	Asserted. on Rising Edge of State		Result
		N	N + 2	
1	DTACK	A	S	Normal cycle terminate and continue
	BERR	NA	X	
	HALT	NA	X	
2	DTACK	A	S	Normal cycle terminate and halt. Continue when HALT removed.
	BERR	NA	X	
	HALT	A	S	
3	DTACK	NA	A	Normal cycle terminate and halt. Continue when HALT removed.
	BERR	NA	NA	
	HALT	A	S	
4	DTACK	X	X	Terminate and take bus error trap.
	BERR	A	S	
	HALT	NA	NA	
5	DTACK	NA	X	Terminate and re-run.
	BERR	A	S	
	HALT	NA	A	
6	DTACK	X	X	Terminate and re-run when HALT removed.
	BERR	X	S	
	HALT	A	S	
7	DTACK	NA	X	Terminate and re-run when HALT removed.
	BERR	NA	A	
	HALT	A	S	

Legend 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 4.4 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 4.5 (DTACK is assumed to be negated normally in all cases ; for best results, both DTACK and BERR should be negated when address strobe is negated).

Table 4.5 : $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ Negation Results.

Conditions of Termination in Table 4–4	Control Signal	Negated an Rising Edge of State			Results - Next Cycle
		n		n + 2	
Bus Error	$\overline{\text{BERR}}$	•	or	•	Takes bus error trap
	$\overline{\text{HALT}}$	•	or	•	
Re-run	$\overline{\text{BERR}}$	•	or	•	Illegal sequence, usually traps to vector number 0.
	$\overline{\text{HALT}}$	•			
Re-run	$\overline{\text{BERR}}$	•			Re-runs the bus cycle
	$\overline{\text{HALT}}$			•	
Normal	$\overline{\text{BERR}}$	•			May Lengthen Next Cycle
	$\overline{\text{HALT}}$	•	or	•	
Normal	$\overline{\text{BERR}}$			•	If next cycle is started it will be terminated as a bus error.
	$\overline{\text{HALT}}$	•	or	None	

• = Signal is negated in this bus state.

EXAMPLE A :

A system uses a watch-dog timer to terminate accesses to unpopulated address space. The timer asserts DTACK and BERR simultaneously after time out (case 4).

EXAMPLE B :

A system uses error detection on RAM contents. Designer may (a) delay DTACK until data verified and return BERR and HALT simultaneously to re-run error cycle (case 6), or if valid, return DTACK (case 1) ; (b) delay DTACK until data verified and return BERR at same time as DTACK if data in error (case 4).

4.4. ASYNCHRONOUS VERSUS SYNCRHONOUS OPERATION

4.4.1. ASYNCHRONOUS OPERATION. To achieve clock frequency independence at a system level, the TS68000 can be used in an asynchronous manner. This entails using only the bus handshake lines (AS, UDS, LDS, DTACK, BERR, HALT, and VPA) to control the data transfer. Using this method, 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 (DTACK) to terminate the bus cycle. If no slave responds or the access is invalid, external control logic asserts the BERR, or BERR and HALT, signal to abort or re-run the bus cycle.

The 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 DTACK may precede data is given as parameter #31u 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 AS to the assertion of DTACK. This is because the MPU will insert wait cycles of one clock period each until DTACK is recognized.

4.4.2. SYNCHRONOUS OPERATION. To allow for those systems which use the system clock as a signal to generate DTACK and other asynchronous inputs, the asynchronous input setup time is given as parameter #47. If this setup is met on an input, such as 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 setup time it is not guaranteed not to be recognized. In addition, if 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 setup time given as parameter #27. Given this, parameter #31 may be ignored. Note that if DTACK is asserted, with the required setup 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.

During an active bus cycle, BERR is sampled on every falling edge of the clock starting with S2. 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 BERR is asserted in the absence of DTACK, in which case it will terminate one clock cycle later in S9. VPA is sampled only on the third falling edge of the system clock before the rising edge of the E clock.

SECTION 5

PROCESSING STATES

This section describes the actions of the TS68000 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 TS68000 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 design vide 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.

5.1. 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 by 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.

5.1.1. 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.

5.1.2. 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.

5.1.3. 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.

5.1.4. 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 5.1 lists the classification of references.

Table 5.1 : 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	(unasigned)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	Interrupt Acknowledge

5.2. 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.

5.2.1. 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 5.1), 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 5.2) 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 5.3. The memory layout of for exception vectors is given in table 5.2.

Figure 5.1 : Format of Vector Table Entries.

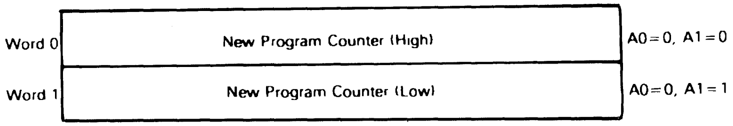


Figure 5.2 : Vector Number Format.

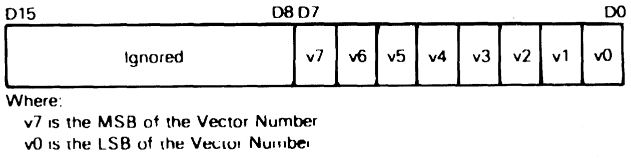

D15
D8 D7
D0
Ignored
v7
v6
v5
v4
v3
v2
v1
v0
Where:
v7 is the MSB of the Vector Number
v0 is the LSB of the Vector Number

Figure 5.3 : Exception Vector Address Calculation.

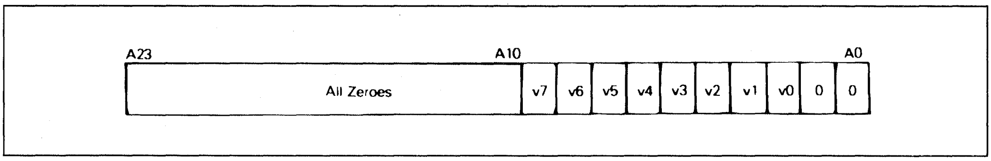


Table 5.2 : Exception Vector Table.

Vector Number(s)	Address			Assigment
	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		–
64–225	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 SGS-THOMSON Microelectronics. No user peripheral devices should be assigned these numbers.

As shown in table 5.2, 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.

5.2.2. 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.

5.2.3. 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.

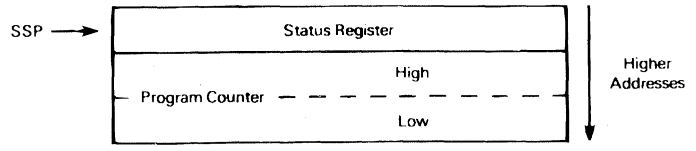
The third step is to save the current processor status, except for the reset exception. The current program counter value and the saved copy of the status register arer stacked using the supervisor stack pointer as shown in figure 5.4. The program counter value stacked usually points to the next unexecuted 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.

5.2.4. 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.

Figure 5.4 : Exception Stack Order (groups 1 and 2).



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 5.3.

Table 5.3 : 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 Divivde	Exception processing is started by normal instruction execution.

5.3. 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 exceptionsn, how each arises, and how each is processed.

5.3.1. 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.

5.3.2. INTERRUPTS. Seven levels of interrrupt 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 instruction execution commences in the interrupt handling routine. A flowchart for the interrupt acknowledge sequence is given in figure 5.5, a timing diagram is given in figure 5.6, and the interrupt processing sequence is shown in figure 5.7.

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.

5.3.3. UNINITIALIZED INTERRUPT. An interrupting device asserts $\overline{\text{VPA}}$ or provides an interrupt during an interrupt acknowledge cycle to the TS68000. If the vector register has not been initialized, the res-ponding TS68000 Family peripheral will provide vector 15, the uninitialized interrupt vector. This provides a uniform way to recover from a programming error.

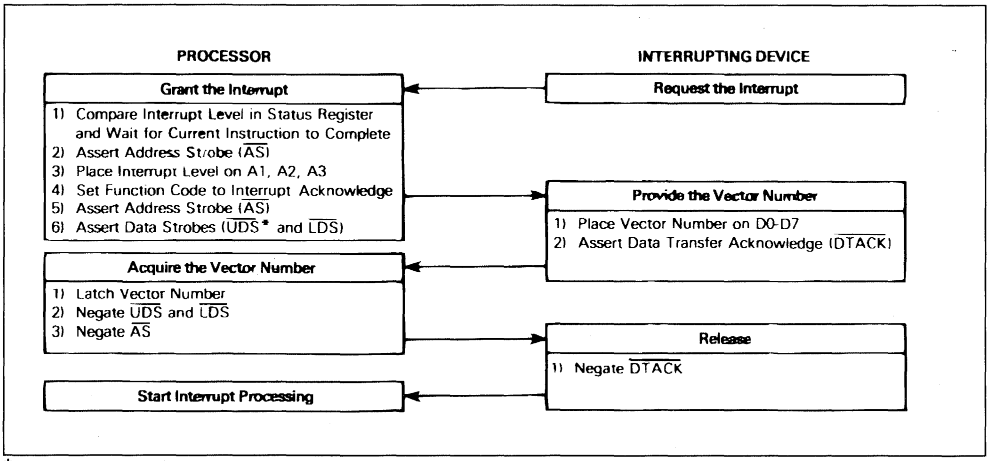
5.3.4. SPURIOUS INTERRUPT. If during the interrupt acknowledge cycle no device responds by asserting $\overline{\text{DTACK}}$ or $\overline{\text{VPA}}$, the bus error line should be asserted to terminate the vector acquisition. The processor separates the processing of this error from bus error by fetching the spurious interrupt vector instead of the bus error vector. The processor then proceeds with the usual exception processing.

5.3.5. 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.

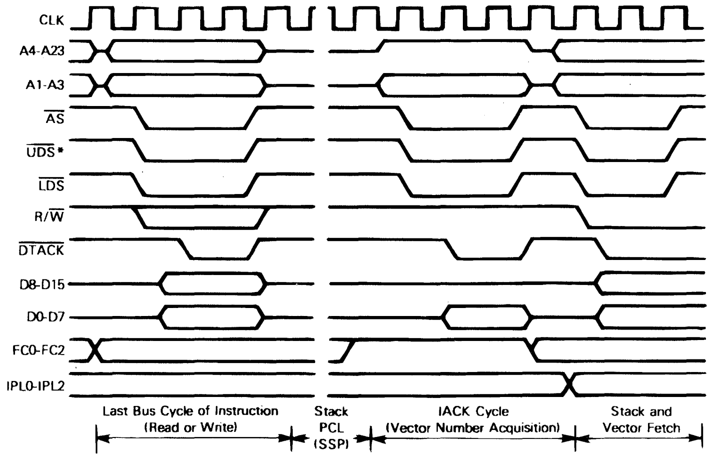
The signed divide (DIVS) and unsigned (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

Figure 5.5 : Vector Acquisition Flowchart.


PROCESSOR
INTERRUPTING DEVICE
Grant the Interrupt
1) Compare Interrupt Level in Status Register and Wait for Current Instruction to Complete
2) Assert Address Strobe ($\overline{\text{AS}}$)
3) Place Interrupt Level on A1, A2, A3
4) Set Function Code to Interrupt Acknowledge
5) Assert Address Strobe ($\overline{\text{AS}}$)
6) Assert Data Strobes ($\overline{\text{UDS}}$* and $\overline{\text{LDS}}$)
Request the Interrupt
Provide the Vector Number
1) Place Vector Number on D0-D7
2) Assert Data Transfer Acknowledge ($\overline{\text{DTACK}}$)
Acquire the Vector Number
1) Latch Vector Number
2) Negate $\overline{\text{UDS}}$ and $\overline{\text{LDS}}$
3) Negate $\overline{\text{AS}}$
Release
1) Negate $\overline{\text{DTACK}}$
Start Interrupt Processing

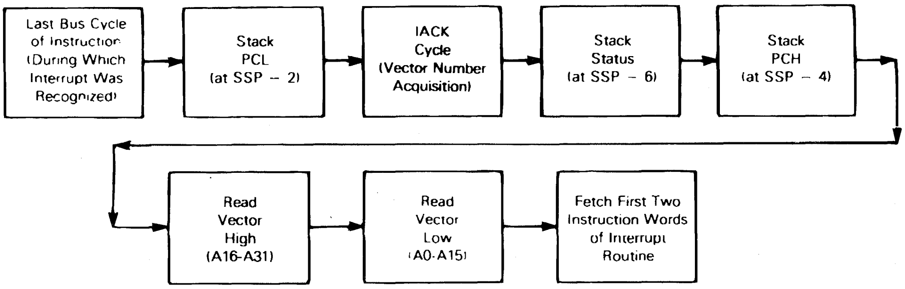
* Although a vector number is one byte, both data stobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D8 through D15 at this time.

Figure 5.6 : Interrupt Acknowledge Cycle Timing Diagram.



* Although a vector number is one byte, both data stobes 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 5.7 : Interrupt Processing Sequence.


Last Bus Cycle of Instruction (During Which Interrupt Was Recognized)
Stack PCL (at SSP – 2)
IACK Cycle (Vector Number Acquisition)
Stack Status (at SSP – 6)
Stack PCH (at SSP – 4)
Read Vector High (A16-A31)
Read Vector Low (A0-A15)
Fetch First Two Instruction Words of Interrupt Routine

Note : SSP refers to the value of the supervisor stack pointer before the interrupt occurs.

5.3.6. 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. SGS-THOMSON 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 TS68000 Family compatible microprocessors. They are : $4AFA, $4AFB, and $4AFC. Two of the patterns, $4AFA and $4AFB, are reserved for SGS-THOMSON Microelectronics system products. The third pattern, S4AFC, 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.

5.3.7. 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 USP

5.3.8. TRACING. To aid in program development, the TS68000 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.

5.3.9. 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 5.8 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 destroy any memory contents. Only the RESET pin can restart a halted processor.

5.3.10. 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 5.9, an address error will execute a short bys cycle followed by exception processing.

Figure 5.8 : Exception Stack Order (group 0).

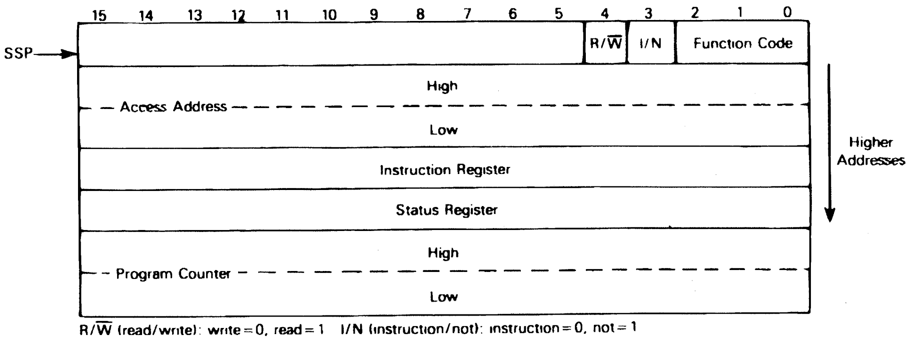
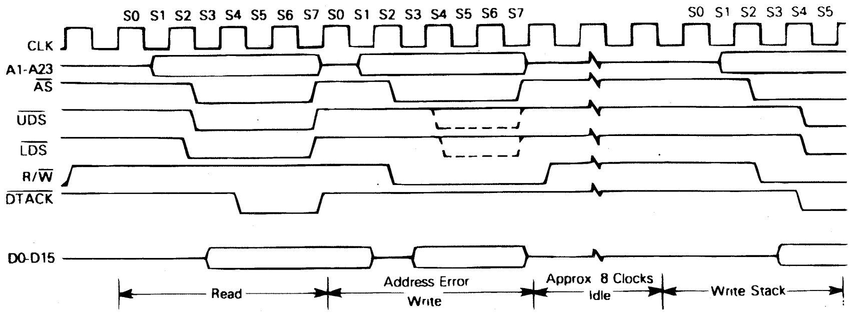


Figure 5.9 : Address Error Timing Diagram.



SECTION 6

INTERFACE WITH EF6800 PERIPHERALS

SGS-THOMSON Microelectronics extensive line of EF6800 peripherals are directly compatible with the TS68000. Some of these devices that are particularly useful are :

EF6821 Peripheral Interface Adapter

EF6840 Programmable Timer Module

EF9345, EF9367 CRT Controllers

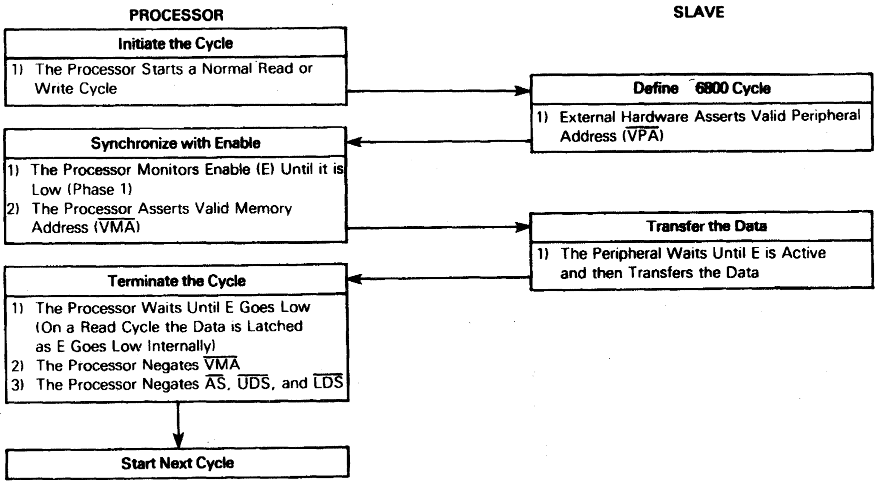
EF6850 Asynchronous Communications Interface Adapter

EF6852 Synchronous Serial Data Adapter

EF6854 Advanced Data Link Controller

To interface the synchronous EF6800 peripherals with the asynchronous TS68000, the processor modifies its bus cycle to meet the EF6800 cycle requirements whenever an EF6800 device address is detected. This is possible since both processors use memory mapped I/O. Figure 6.1 is a flowchart of the interface operation between the processor and EF6800 devices.

Figure 6.1 : EF6800 Interfacing Flowchart.


PROCESSOR
SLAVE
Initiate the Cycle
1) The Processor Starts a Normal Read or Write Cycle
Define 6800 Cycle
1) External Hardware Asserts Valid Peripheral Address ($\overline{VPA}$)
Synchronize with Enable
1) The Processor Monitors Enable (E) Until it is Low (Phase 1)
2) The Processor Asserts Valid Memory Address ($\overline{VMA}$)
Transfer the Data
1) The Peripheral Waits Until E is Active and then Transfers the Data
Terminate the Cycle
1) The Processor Waits Until E Goes Low (On a Read Cycle the Data is Latched as E Goes Low Internally)
2) The Processor Negates $\overline{VMA}$
3) The Processor Negates $\overline{AS}$, $\overline{UDS}$, and $\overline{LDS}$
Start Next Cycle

6.1. DATA TRANSFER OPERATION

Three signals on the processor provide the EF6800 interface. They are : enable (E), valid memory address (VMA), and valid peripheral address (VPA). Enable corresponds to the E or phase 2 signal in existing 6800 systems. The bus frequency in one tenth of the incoming TS68000 clock frequency. The timing of E allows 1 megahertz peripherals to be used with 8 megahertz TS68000s. 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 VPA accesses on successive E pulses.

EF6800 cycle timing is given in figures 6.2, 6.3, 8.7, and 8.8. 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.

Figure 6.2 : TS68000 to EF6800 Peripheral Timing – Best Case.

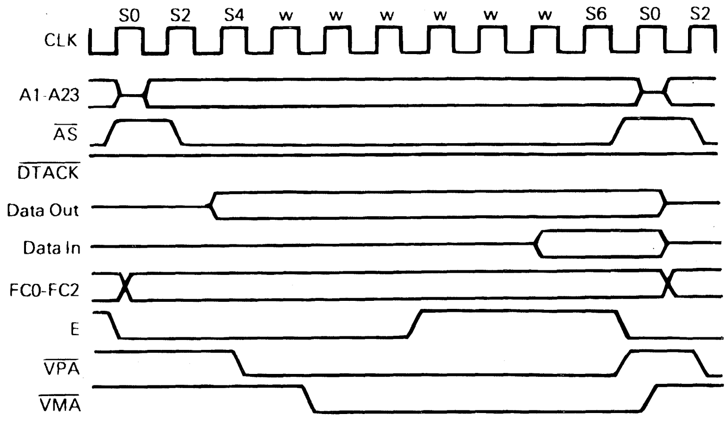
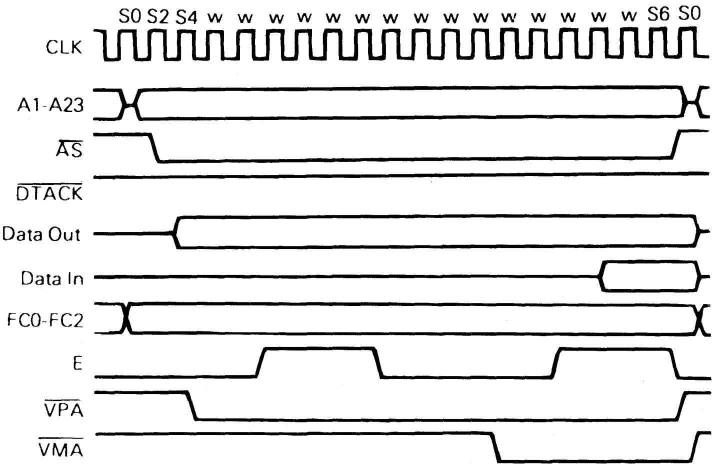


Figure 6.3 : TS68000 to EF6800 Peripheral Timing – Worst Case.



During state 2, the address strobe ($\overline{\text{AS}}$) 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/$\overline{\text{W}}$) 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 $\overline{\text{VPA}}$.

The $\overline{\text{VPA}}$ input signals the processor that the address on the bus is the address of an EF6800 device (or an area reserved for EF6800 devices) and that the bus should conform to the phase 2 transfer characteristics of the EF6800 bus. Valid peripheral address is derived by decoding the address bus, conditioned by the address strobe. Chip select for the EF6800 peripherals should be derived by decoding the address bus conditioned by $\overline{\text{VMA}}$.

After recognition of $\overline{\text{VPA}}$, the processor assures that the enable (E) is low, by waiting if necessary, and subsequently asserts $\overline{\text{VMA}}$. Valid memory address is then used as part of the chip select equation of the peripheral. This ensures that the EF6800 peripherals are selected and deselected at the correct time. The peripheral now runs its cycle during the high portion of the E signal. Figures 6.2 and 6.3 depict the best and worst case EF6800 cycle timing. This cycle length is dependent strictly upon when VPA is asserted in relationship to the E clock.

If we assume that external circuitry asserts $\overline{\text{VPA}}$ as soon as possible after the assertion of $\overline{\text{AS}}$, then 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 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 - $\overline{\text{VPA}}$ 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 - $\overline{\text{VPA}}$ 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-impedence state.

During a write cycle, the data bus is put in the high-impedence state and the read/write signal is switched high. The peripheral logic must remove VPA within one clock after the address strobe is negated.

$\overline{\text{DTACK}}$ should not be asserted while $\overline{\text{VPA}}$ is asserted. Notice that the TS68000 $\overline{\text{VMA}}$ is active low, contrasted with the active high EF6800 VMA. This allows the processor to put its buses in the high-impedance state on DMA requests without inadvertently selecting the peripherals.

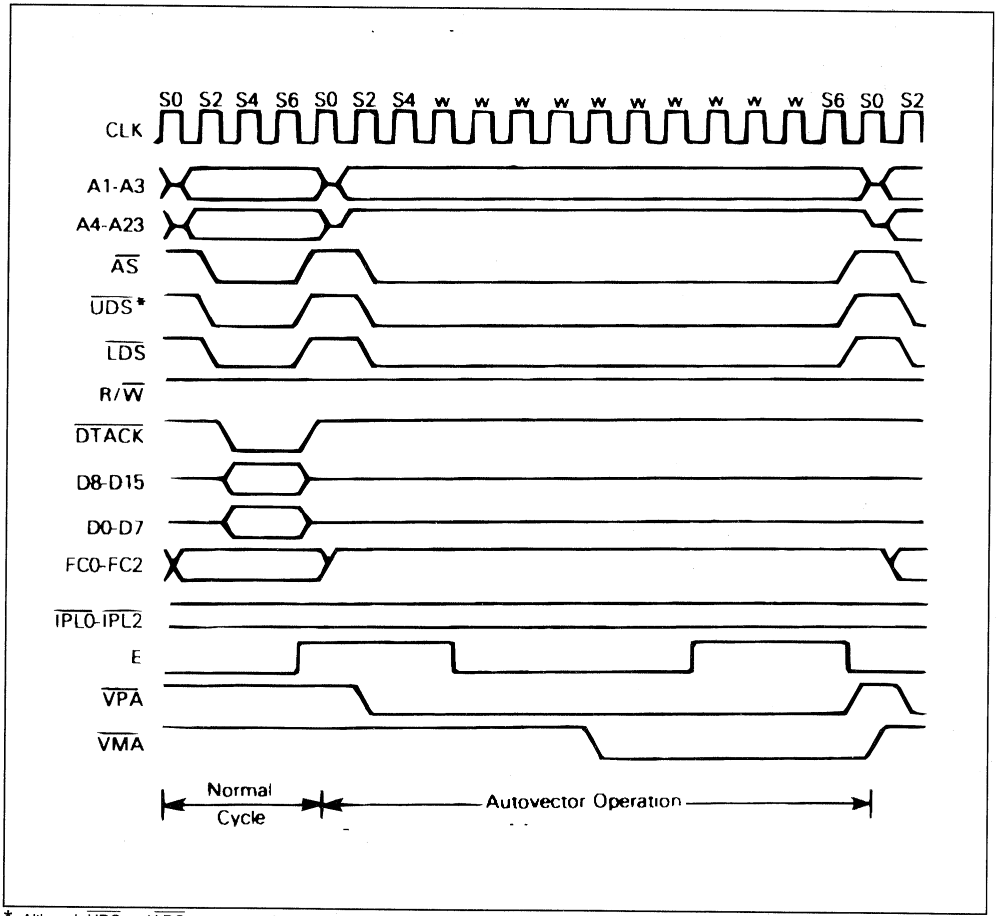
6.2. INTERRUPT INTERFACE OPERATION

During an interrupt acknowledge cycle while the processor is fetching the vector, the VPA is asserted, the TS68000 will assert $\overline{\text{VMA}}$ and complete a normal EF6800 read cycle as shown in figure 6.4. 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 EF6800 interrupt sequence. The basic difference is that there are six normalinterrupt vectors and one NMI type vector. As with both the EF6800 and the TS68000'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 $\overline{\text{VMA}}$ is asserted during autovectoring, care should be taken to insure that the 6800 peripheral address decoding prevents unintended accesses.

Figure 6.4 : Autovector Operation Timing Diagram.



* 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.

SECTION 7

INSTRUCTION SET AND EXECUTION TIMES

7.1. INSTRUCTION SET

The following paragraphs provide information about the addressing categories and instruction set of the TS68000.

7.1.1. 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.

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 7.1 shows the various categories to which each of the effective address modes belong. Table 7.2 is the instruction set summary.

Table 7.1 : 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	X	X	X	–	–

Table 7.2 : Instruction Set.

Mnemonic	Description	Operation	Conditions 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) Λ (source) → Destination	–	*	*	0	0
ANDI	AND Immediate	(destination) Λ Immediate Data → Destination	–	*	*	0	0
ANDI to CCR	AND Immediate to Condition Codes	(source) Λ CCR → CCR	*	*	*	*	*
ANDI to SR	AND Immediate to Status Register	(source) Λ SR → SR	*	*	*	*	*
ASL, ASR	Arithmetic Shift	(destination) shifted by <count> → Destination	*	*	*	*	*
BCC	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 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)	–	*	*	*	*
DBCC	Test Condition, Decrement and Branch	If ~CC then Dn – 1 → Dn ; Dn ≠ – 1 then PC + d → PC	–	–	–	–	–
DIVS	Signed Divide	(destination)/(source) → Destiantion	–	*	*	*	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

Λ logical AND	* affected
ς logical OR	– unaffected
⊕ logical exclusive OR	0 cleared
~ logical complement	1 set
	U undefined

Table 7.2 : Instruction Set (continued).

Mnemonic	Description	Operation	Conditions Codes				
			X	N	Z	V	C
JMP	Jump	Destination → PC	–	–	–	–	–
JSR	Jump to Subroutine	PC → – (SP) ; Destination → PC	–	–	–	–	–
LEA	Load Effective Address	<ea> → 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 the Status Register	(source) → SR	*	*	*	*	*
MOVE from SR	Move from the Status Register	SR → Destination	–	–	–	–	–
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 – (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	<ea> → – (SP)	–	–	–	–	–
RESET	Reset External Device	–	–	–	–	–	–

[ ] = bit number	* affected
Λ logical AND	– unaffected
ς logical OR	0 cleared
⊕ logical exclusive OR	1 set
~ logical complement	U undefined

Table 7.2 : Instruction Set (continued).

Mnemonic	Description	Operation	Conditions Codes				
			X	N	Z	V	C
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	(destination)$_{10}$ – (source)$_{10}$ – X → Destination	*	U	*	U	*
SCC	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 Resgister Halves	Register [13:16] ÷ 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	–	–	–	–	–

[ ] = bit number	* affected
Λ logical AND	– unaffected
ς logical OR	0 cleared
⊕ logical exclusive OR	1 set
~ logical complement	U undefined

7.1.2. INSTRUCTION PREFETCH. The TS68000 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.

7.2. 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 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.

The number of periods includes instruction fetch and all applicable operand fetches and stores.

7.2.1. EFFECTIVE ADDRESS OPERAND CALCULATION TIMING. Table 7.3 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 parenthesis as (r/w). Note there are no write cycles involved in processing the effective address.

7.2.2. MOVE INSTRUCTION EXECUTION TIMES. Tables 7.4 and 7.5 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 7.3 : 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.

Table 7.4 : 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**(3/1)	**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)
xxxW	**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)
xxxL	**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)	**20**(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)	**22**(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 7.5 : 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)
xxxW	**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)
xxxL	**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 7.6 : Standard Instruction Execution Times.

Instruction	Size	op < ea >, Ant	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 only
* indicates maximum value
** The base time of six clock periods is increased to eight 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 TS68000 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 >
MULU : n = concatanate 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.

7.2.3. STANDARD INSTRUCTION EXECUTION TIMES. The number of clock periods shown in table 7.6 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 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 7.6 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

7.2.4. IMMEDIATE INSTRUCTION EXECUTION TIMES. The number of clock periods shown in table 7.7 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 7.7, the headings have the following meanings : # = immediate operand, Dn = data register operand, An = address register operand, M = memory operand, and SR = status register.

7.2.5. SINGLE OPERAND INSTRUCTION EXECUTION TIMES. Table 7.8 indicates 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.

7.2.6. SHIFT/ROTATE INSTRUCTION EXECUTION TIMES. Table 7.9 indicates 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.

7.2.7. BIT MANIPULATION INSTRUCTION EXECUTION TIMES. Table 7.10 indicates the number of clock periods required for the bit manipulation instructions. The number of bus read and write cycles is shown in parethesis 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 7.7 : 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) +
ADDO	Byte, Word	**4**(1/0)	**8**(1/1)*	**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	41/0)	**8**(1/0)*	**8**(1/1)+
	Long	**8**(1/0)	**8**(1/0)	**12**(1/2)+

+ add effective address calculation time
* word only

Table 7.8 : 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**(1/1)+
NEGX	Byte, Word	**4**(1/0)	**8**(1/2)+
	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**(1/2)+
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

Table 7.9 : 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

Table 7.10 : 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 address calculation time
* indicates maximum value

7.2.8. CONDITIONAL INSTRUCTION EXECUTION TIMES. Table 7.11 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.

Table 7.11 : 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)

+ add effective address calculation time
* indicates maximum value

7.2.9. JMP, JSR, LEA, PEA, AND MOVEM INSTRUCTION EXECUTION TIMES. Table 7.12 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 7.12 : JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times.

Instr	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) does not affect the instruction's execution time

7.2.10. MULTI-PRECISION INSTRUCTION EXECUTION TIMES. Table 7.13 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 7-13, the headings have the following meanings : Dn = data register operand and M = memory operand.

Table 7.13 : 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)

7.2.11. MISCELLANEOUS INSTRUCTION EXECUTION TIMES. Tables 7.14 and 7.15 indicate the number of clock periods for the following miscellaneous instructions. The number of bus read and write cycles 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.

Table 7.14 : 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 7.15 : 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)

7.2.12. EXCEPTION PROCESSING EXECUTION TIMES. Table 7.16 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 7.16 : 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 RESET and HALT are first sampled as negated to when instruction execution starts.

SECTION 8

ELECTRICAL SPECIFICATIONS

This section contains electrical specifications and associated timing information for the TS68000. These specifications represent an improvement over previously published specifications for the 8,10,12.5, 16 MHz.TS68000 and are valid only for products bearing date codes of 8922 and later.

8.1. ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	– 0.3 to 7	V
V_{IN}	Input Voltage	– 0.3 to 7	V
T_A	Operating Temperature Range TS68000C TS68000V TS68000M	T_L to T_H 0 to 70 – 40 to 85 – 55 to 125	°C
T_{stg}	Storage Temperature	– 55 to 150	°C

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})

8.2. THERMAL DATA

Package	T_A Range	θ_{JA} (°C/W)	P_D (W) @ T_A Min.	T_J (°C) @ T_A Min.	P_D (W) @ T_A Max.	T_J (°C) @ T_A Max.
Plastic DIL	0 °C to 70 °C	30	1.2	36	1.0	100
PLCC	0 °C to 70 °C	45	1.2	54	1.0	115
Ceramic PGA	0 °C to 70 °C	33	1.2	40	1.0	103
	0 °C to 85 °C	33	1.2	40	1.0	118
	– 40 °C to 85 °C	33	1.5	10	1.0	118

8.3. DC ELECTRICAL CHARACTERISTICS
(V_{CC} = 5 Vdc ± 5 % ; GND = 0 Vdc ; T_A = T_L to T_H ; see figures 8.1, 8.2 and 8.3)

Symbol	Parameter		Min.	Max.	Unit
V_{IH}	Input High Voltage		2	V_{CC}	V
V_{IL}	Input Low Voltage		GND – 0.3	0.8	V
I_{IN}	Input Leakage Current @ 5.25 V	$\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$ CLK, $\overline{IPL0}$-$\overline{IPL2}$, $\overline{VPA}$ $\overline{HALT}$, $\overline{RESET}$	 – –	 2.5 20	µA
I_{TSI}	Three-state (off state) Input Current @ 2.4 V/0.4 V	$\overline{AS}$, A1-A23, D0-D15, FC0-FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	–	20	µA
V_{OH}	Output High Voltage (I_{OH} = – 400 µA)	E* E, $\overline{AS}$, A1-A23, $\overline{BG}$, D0-D15, FC0-FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	V_{CC} – 0.75 2.4	– –	V
V_{OL}	Output Low Voltage (I_{OL} = 1.6 mA) (I_{OL} = 3.2 mA) (I_{OL} = 5 mA) (I_{OL} = 5.3 mA)	 $\overline{HALT}$ A1-A23, $\overline{BG}$, FC0-FC2 $\overline{RESET}$ E, $\overline{AS}$, D0-D15, $\overline{LDS}$, R/$\overline{W}$ $\overline{UDS}$, $\overline{VMA}$	 – – – –	 0.5 0.5 0.5 0.5	V
P_D***	Power Dissipation (see 8.4 POWER CONSIDERATIONS)		–	–	W
C_{IN}	Capacitance (V_{in} = 0 V, T_A = 25 °C ; Frequency = 1 MHz)**		–	20	pF

* With external pullup resistor of 1.1kΩ.
* * Capacitance is periodically sampled rather than 100% tested.
* * * During normal operation instantaneous VCC current requirements may be as high as 1.5A.

Figure 8.1 : $\overline{\text{RESET}}$ Test Load.

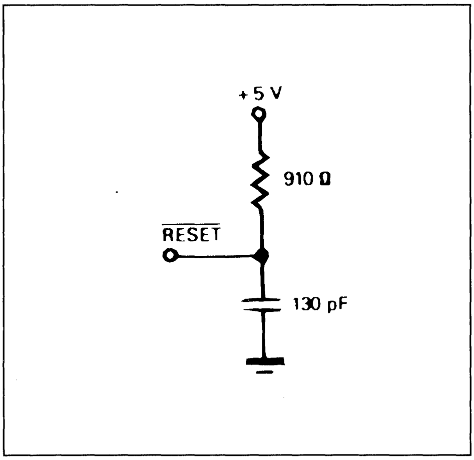


Figure 8.2 : $\overline{\text{HALT}}$ Test Load.

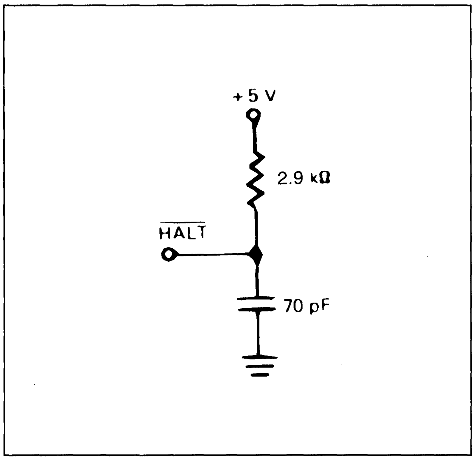
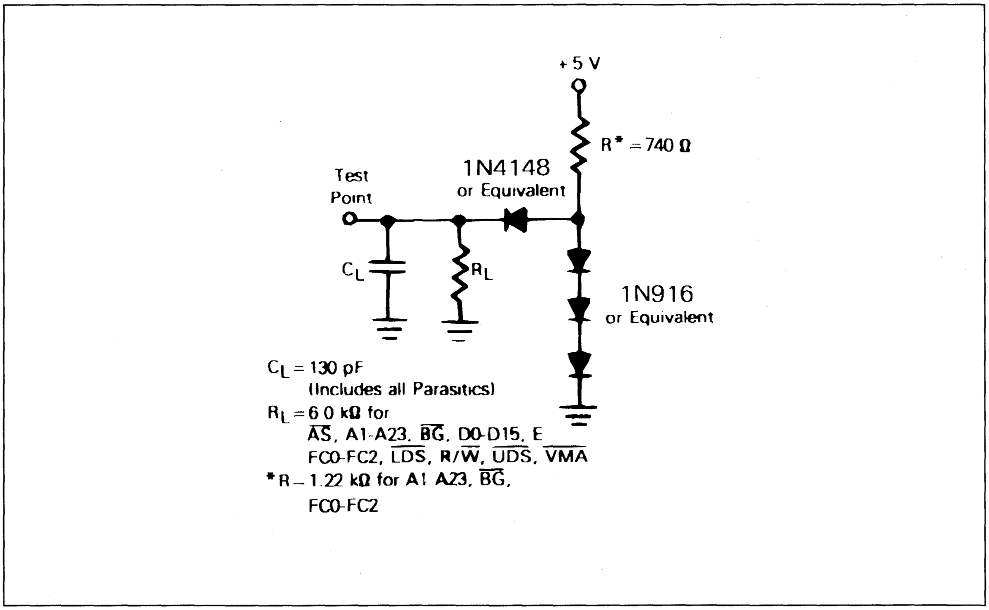


Figure 8.3 : Test Loads.


+5 V
R* = 740 Ω
1N4148
or Equivalent
Test
Point
C_L
R_L
1N916
or Equivalent
C_L = 130 pF
(Includes all Parasitics)
R_L = 6.0 kΩ for
$\overline{\text{AS}}$, A1-A23, $\overline{\text{BG}}$, D0-D15, E
FC0-FC2, $\overline{\text{LDS}}$, R/$\overline{\text{W}}$, $\overline{\text{UDS}}$, $\overline{\text{VMA}}$
*R = 1.22 kΩ for A1-A23, $\overline{\text{BG}}$,
FC0-FC2

8.4. POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from :

$T_J = T_A + (P_D \cdot \theta_{JA})$

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 :

$P_D = K \div (T_J + 273°C)$

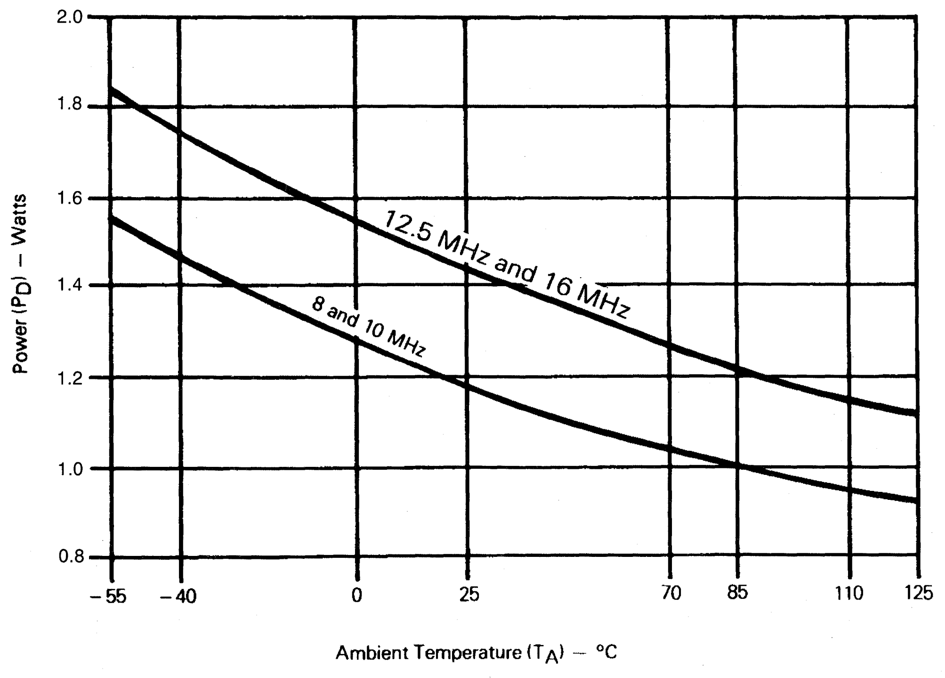
Solving equations 1 and 2 for K gives :

$K = PD \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2$

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 8.4 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 the various TS68000 packages. However, actual θJA's in the range of 30°C to 50°C/watt only change the curves slightly.

Figure 8.4 : TS68000 Power Dissipation (P_D) vs Ambient Temperature (T_A).



The total thermal resistance of a package (ΘJA) 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 :

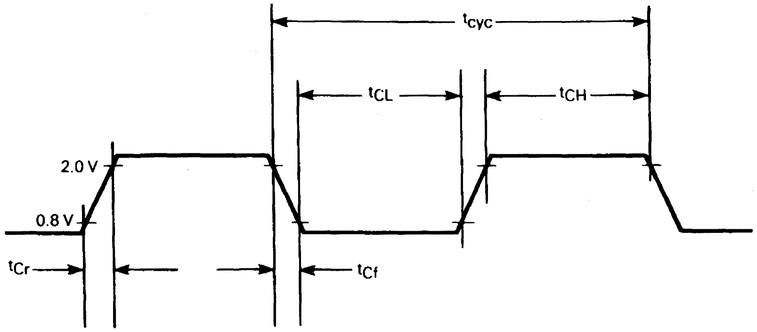
$$\theta_{JA} = \theta_{JC} + \theta_{CA}$$

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.

8.5. AC ELECTRICAL SPECIFICATIONS – CLOCK TIMING (see figure 8-5)

Symbol	Parameter	8 MHz		10 MHz		12.5 MHz		16 MHz		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
f	Frequency of Operation	4	8	4	10	8	12.5	8	16.7	MHz
t_{cyc}	Cycle Time	125	250	100	250	80	125	60	125	ns
t_{CL} t_{CH}	Clock Pulse Width	55 55	125 125	45 45	125 125	35 35	62.5 62.5	25 25	62.5 62.5	ns
t_{Cr} t_{Cf}	Rise and Fall Times	– –	10 10	– –	10 10	– –	5 5	– –	5 5	ns

Figure 8.5 : Clock Input Timing Diagram.



Note : Timing measurements are referenced to and from a low voltage of 0.8 volt and high a voltage of 2.0 volts, 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 0.8 volt and 2.0 volts.

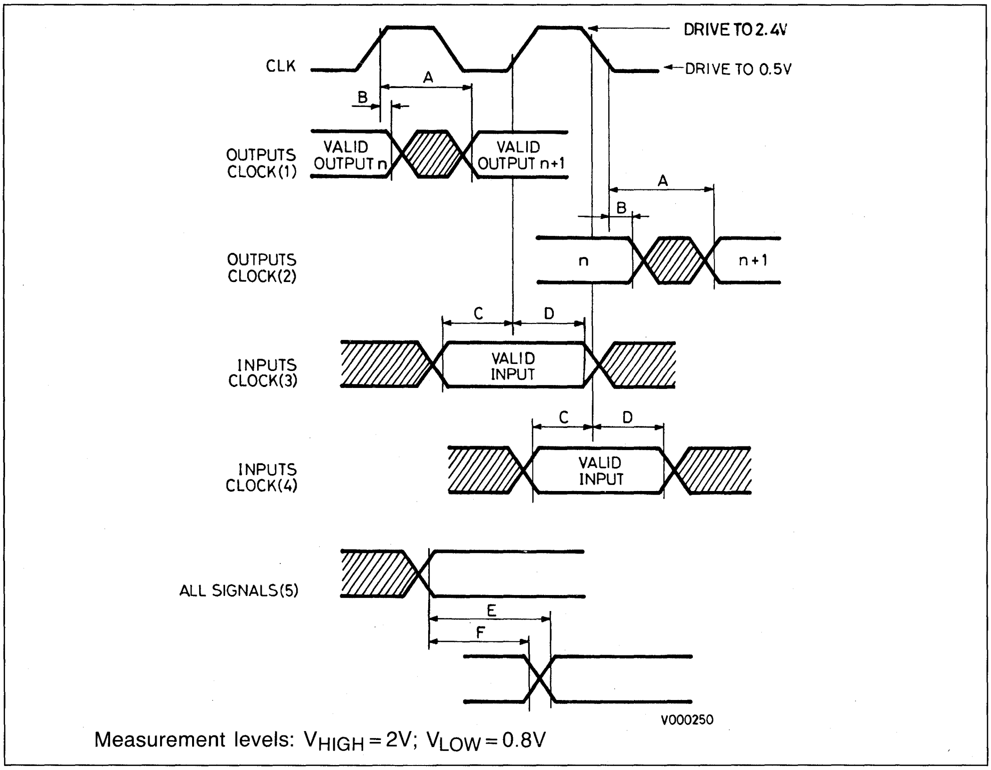
8.6. AC ELECTRICAL SPECIFICATION DEFINITIONS

The AC specifications presented consist of output delays, input setup and hold times, and signal skew times. All signals are specified relative to an appropriate edge of the clock and possibly to one or more other signals.

The measurement of the AC specifications is defined by the waveforms shown in figure 8.6 in order to test the parameters guaranteed by SGS-THOMSON. Inputs must be driven to the voltage levels specified in the figure 8.6. Outputs are specified with minimum and/or maximum limits, as appropriate, and are measured as shown in the same figure. Inputs are specified with minimum setup and hold times, and are measured as shown. Finally the measurement for signal-to-signal specifications are also shown.

Note that the testing levels used to verify conformance to the AC specifications does not affect the guaranteed DC operation of the device as specified in the DC electrical characteristics.

Figure 8.6 : Drive Levels and Test Points for AC Specifications.



Notes :
1. This output timing is applicable to all parameters specified relative to the rising edge of the clock.
2. This output timing is applicable to all parameters specified relative to the falling edge of the clock.
3. This input timing is applicable to all parameters specified relative to the rising edge of the clock.
4. This input timing is applicable to all parameters specified relative to the falling edge of the clock.
5. This timing is applicable to all parameters specified relative to the assertion negation of another signal.

Legend :
A. Maximum output delay specification.
B. Minimum output hold time.
C. Minimum input setup time specification.
D. Minimum input hold time specification.
E. Signal valid to signal valid specification (max. or min.).
F. Signal valid to signal invalid specification (max. or min.).

8.7. AC ELECTRICAL SPECIFICATIONS – READ CYCLES

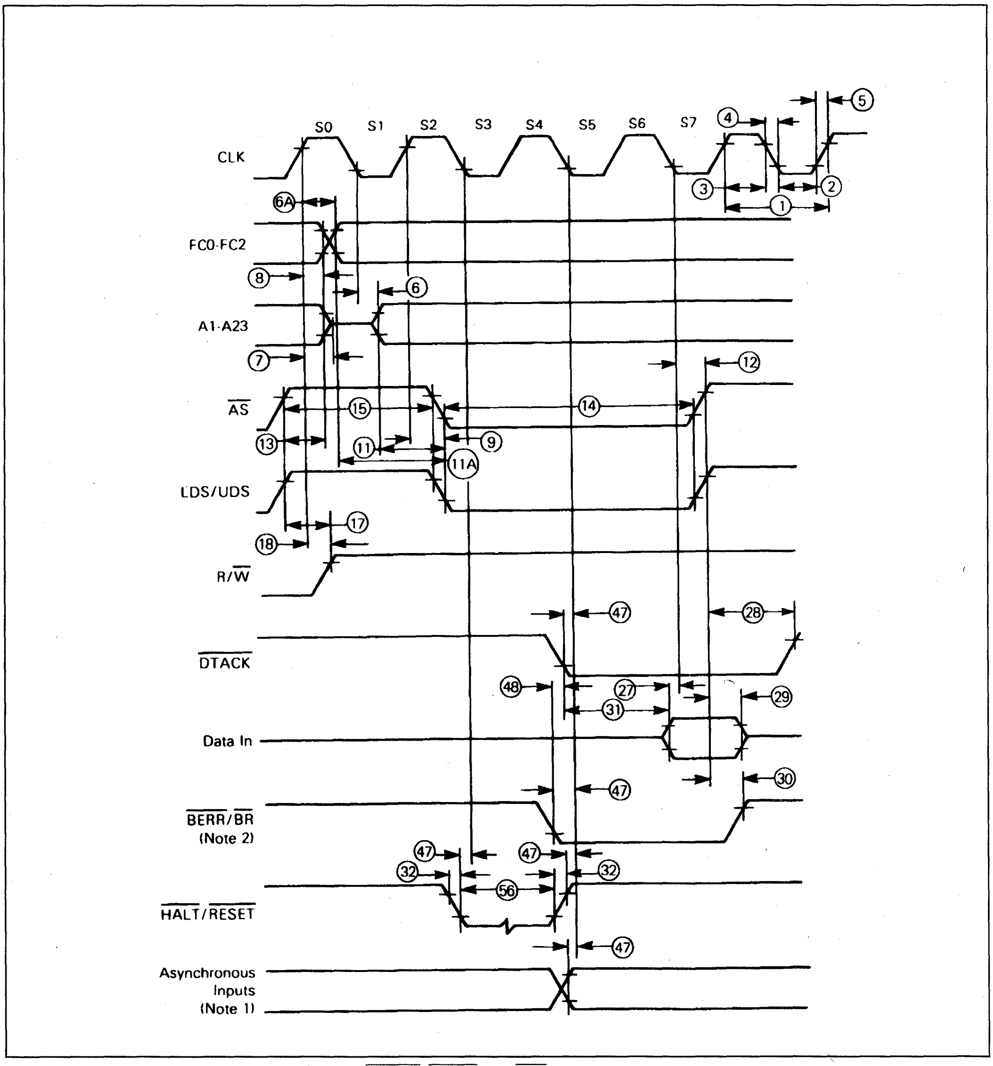
(V_{CC} = 5Vdc ± 5% ; GND = 0Vdc ; T_A = T_L to T_H ; see figure 8.7)

N°	Symbol	Parameter	8 MHz		10 MHz		12.5 MHz		16 MHz		Unit
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
1	t_{CYC}	Clock Period	125	250	100	250	80	125	60	125	ns
2	t_{CL}	Clock Width Low	55	125	45	125	35	62	25	62	ns
3	t_{CH}	Clock Width High	55	125	45	125	35	62	25	62	ns
4	t_{Cf}	Clock Fall Time	–	10	–	10	–	5	–	5	ns
5	t_{Cr}	Clock Rise Time	–	10	–	10	–	5	–	5	ns
6	t_{CLAV}	Clock Low to Address Valid	–	60	–	50	–	50	–	40	ns
6A	t_{CHFCV}	Clock High to FC Valid	–	60	–	50	–	50	–	40	ns
7	t_{CHADZ}	Clock High to Address, Data Bus High Impedance (maximum)	–	70	–	60	–	50	–	50	ns
8	t_{CHAFI}	Clock High to Address, FC Invalid (minimum)	0	–	0	–	0	–	0	–	ns
9(1)	t_{CHSL}	Clock High to $\overline{AS}$, $\overline{DS}$ Low	0	60	0	55	0	50	0	45	ns
11(2)	t_{AVSL}	Address Valid to $\overline{AS}$, $\overline{DS}$ Low	20	–	15	–	10	–	5	–	ns
11A(2)	t_{FCVSL}	FC Valid to $\overline{AS}$, $\overline{DS}$ Low	50	–	40	–	35	–	30	–	ns
12(1)	t_{CLSH}	Clock Low to $\overline{AS}$, $\overline{DS}$ High	–	50	–	45	–	40	–	30	ns
13²	t_{SHAFI}	$\overline{AS}$, $\overline{DS}$ High to Address/FC Invalid	30	–	20	–	10	–	10	–	ns
14(2)	t_{SL}	$\overline{AS}$, $\overline{DS}$ Width Low	240	–	195	–	160	–	120	–	ns
15(2)	t_{SH}	$\overline{AS}$, $\overline{DS}$ Width High	150	–	105	–	65	–	50	–	ns
17(2)	t_{SHRH}	$\overline{AS}$, $\overline{DS}$ High to R/$\overline{W}$ High	40	–	20	–	10	–	10	–	ns
18(1)	t_{CHRH}	Clock High to R/$\overline{W}$ High	0	55	0	45	0	40	0	45	ns
27(5)	t_{DICL}	Data in to Clock low (setup time)	15	–	10	–	10	–	10	–	ns
28(2)	t_{SHDAH}	$\overline{AS}$, $\overline{DS}$ High to $\overline{DTACK}$ High	0	240	0	190	0	150	0	80	ns
29	t_{SHDII}	$\overline{AS}$, $\overline{DS}$ High to Data In Invalid (hold time)	0	–	0	–	0	–	0	–	ns
30	t_{SHBEH}	$\overline{AS}$, $\overline{DS}$ High to $\overline{BERR}$ High	0	–	0	–	0	–	0	–	ns
31(2,5)	t_{DALDI}	$\overline{DTACK}$ Low to Data in (setup time)	–	70	–	50	–	40	–	30	ns
32	$t_{RHr,f}$	$\overline{HALT}$ and $\overline{RESET}$ Input Transition Time	0	200	0	200	0	150	0	150	ns
47(5)	t_{ASI}	Asynchronous Input Setup Time	20	–	20	–	10	–	10	–	ns
48(3)	t_{BELDAL}	$\overline{BERR}$ Low to $\overline{DTACK}$ Low	20	–	20	–	10	–	10	–	ns
56(4)	t_{HRPW}	$\overline{HALT}$/$\overline{RESET}$ Pulse Width	10	–	10	–	10	–	10	–	Clk.Per.

Notes :
1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the value given in the maximum columns.
2. Actual value depends on clock period.
3. If 47 is satisfied for both $\overline{DTACK}$ and $\overline{BERR}$, 48 may be 0 nanosecond.
4. For power up, the MPU must be held in $\overline{RESET}$ state for 100 milliseconds 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 setup time (47) requirements are satisfied, the $\overline{DTACK}$ low-to-data setup time (31) requirement can be ignored. The data must only satisfy the data-in to clock-low setup time (27) for the following cycle.

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 8.7 : Read Cycle Timing Diagram.



Notes : 1. Setup time for the synchronous inputs $\overline{\text{BGACK}}$, $\overline{\text{IPLO-2}}$, and $\overline{\text{VPA}}$ 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 volt and a high voltage 2.0 volts, 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 0.8 volt and 2.0 volts.

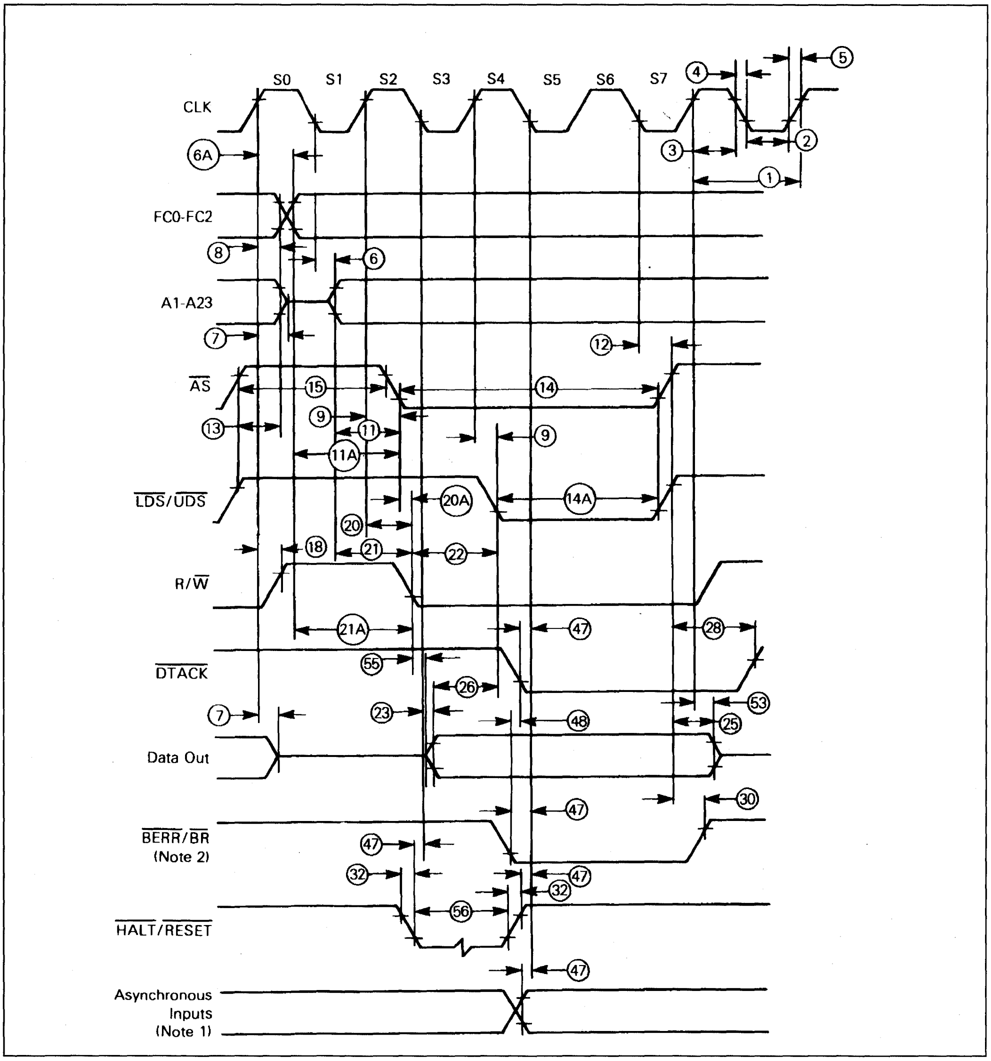
8.8. AC ELECTRICAL SPECIFICATIONS – WRITE CYCLES
(V_{CC} = 5 Vdc ± 5 % ; GND = 0 Vdc ; $T_A = T_L$ to T_H ; see figure 8.8)

N°	Symbol	Parameter	8 MHz		10 MHz		12.5 MHz		16 MHz		Unit
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
1	t_{CYC}	Clock Period	125	250	100	250	80	125	60	125	ns
2	t_{CL}	Clock Width Low	55	125	45	125	35	62	25	62	ns
3	t_{CH}	Clock Width High	55	125	45	125	35	62	25	62	ns
4	t_{Cf}	Clock Fall Time	–	10	–	10	–	5	–	5	ns
5	t_{Cr}	Clock Rise Time	–	10	–	10	–	5	–	5	ns
6	t_{CLAV}	Clock Low to Address Valid	–	60	–	50	–	50	–	40	ns
6A	t_{CHFCV}	Clock High to FC Valid	–	60	–	50	–	50	–	40	ns
7	t_{CHADZ}	Clock High to Address, Data Bus High Impedance (maximum)	–	70	–	60	–	50	–	50	ns
8	t_{CHAFI}	Clock High to Address, FC Invalid (minimum)	0	–	0	–	0	–	0	–	ns
9(1)	t_{CHSL}	Clock High to $\overline{AS}$, $\overline{DS}$ Low	0	60	0	55	0	50	0	45	ns
11(2)	t_{AVSL}	Address Valid to $\overline{AS}$, $\overline{DS}$ Low	20	–	15	–	10	–	5	–	ns
11A(2)	t_{FCVSL}	FC Valid to $\overline{AS}$, Low	50	–	40	–	35	–	30	–	ns
12(1)	t_{CLSH}	Clock Low to $\overline{AS}$, $\overline{DS}$ High	–	50	–	45	–	40	–	30	ns
13(2)	t_{SHAFI}	$\overline{AS}$, $\overline{DS}$ High to Address/FC Invalid	30	–	20	–	10	–	10	–	ns
14(2)	t_{SL}	$\overline{AS}$ Low	240	–	195	–	160	–	120	–	ns
14A(2)	t_{DSL}	$\overline{DS}$ Width Low	115	–	95	–	80	–	60	–	ns
15(2)	t_{SH}	$\overline{AS}$, $\overline{DS}$ Width High	150	–	105	–	65	–	50	–	ns
18(1)	t_{CHRH}	Clock High to R/$\overline{W}$ High	0	55	0	45	0	40	0	40	ns
20(1)	t_{CHRL}	Clock High to R/$\overline{W}$ Low	–	55	–	45	–	40	–	30	ns
20A(6)	t_{ASRV}	$\overline{AS}$, Low to R/$\overline{W}$ Valid	–	20	–	20	–	20	–	20	ns
21(2)	t_{AVRL}	Address Valid to R/$\overline{W}$ Low	20	–	0	–	0	–	0	–	ns
21A(2)	t_{FCVRL}	FC Valid to R/$\overline{W}$ Low	60	–	50	–	30	–	20	–	ns
22(2)	t_{RLSL}	R/$\overline{W}$ Low to $\overline{DS}$ Low	80	–	50	–	30	–	20	–	ns
23	t_{CLDO}	Clock Low to Data out Valid	–	60	–	50	–	50	–	40	ns
25(2)	t_{SHDOI}	$\overline{AS}$, $\overline{DS}$ High to Data Out Invalid	30	–	20	–	15	–	10	–	ns
26(2)	t_{DOSL}	Data out Valid to $\overline{DS}$ Low	30	–	20	–	15	–	10	–	ns
28(2)	t_{SHDAH}	$\overline{AS}$, $\overline{DS}$ High to $\overline{DTACK}$ high	0	240	0	190	0	150	0	80	ns
30	t_{SHBEH}	$\overline{AS}$, $\overline{DS}$ High to $\overline{BERR}$ high	0	–	0	–	0	–	0	–	ns
32	$t_{RHr,f}$	$\overline{HALT}$ and $\overline{RESET}$ Input Transition Time	0	200	0	200	0	150	0	150	ns
47(5)	t_{ASI}	Asynchronous Input Setup Time	20	–	20	–	10	–	10	–	ns
48(3)	t_{BELDAL}	$\overline{BERR}$ Low to $\overline{DTACK}$ Low	20	–	20	–	20	–	10	–	ns
53	t_{CHDOI}	Clock High to Data Out Invalid	0	–	0	–	0	–	0	–	ns
55	t_{RLDBD}	R/$\overline{W}$ to Data Bus Driven	30	–	20	–	10	–	0	–	ns
56(4)	t_{HRPW}	$\overline{HALT}$/$\overline{RESET}$ Pulse Width	10	–	10	–	10	–	10	–	ns

Notes : 1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the value given in the maximum 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 100 milliseconds 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 setup time (47) requirements are satisfied, the $\overline{DTACK}$ low-to-data setup time (31) requirement can be ignored. The data must only satisfy the data-in to clock-low setup time (27) for the following cycle.
6. When $\overline{AS}$, and R/$\overline{W}$ are equally loaded (± 20%), Subtract 10 nanoseconds from the values in these columns.

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 8.8 : Write Cycle Timing Diagram.



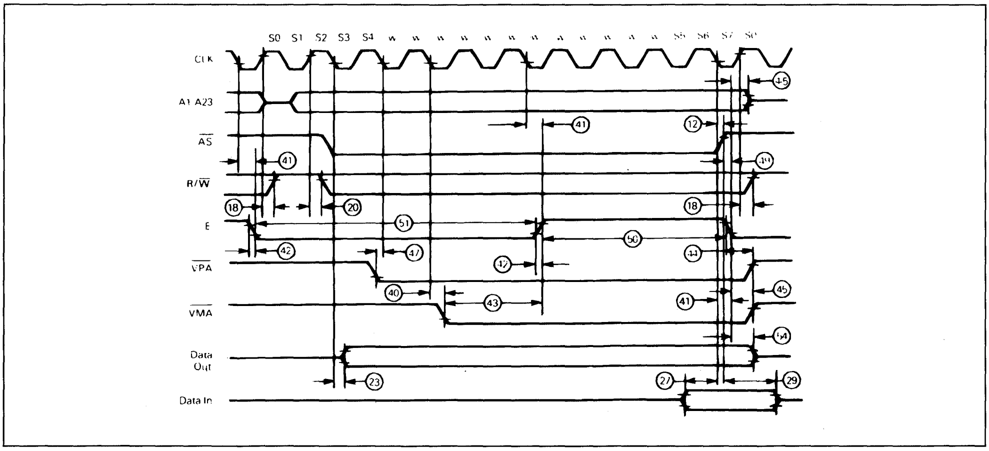
Notes :
1. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 volt and 2.0 volts.
2. Because of loading variations, R/$\overline{W}$ may be valid after $\overline{AS}$ even through both are initiated by the rising edge of S2 (specification 20A).

8.9. AC ELECTRICAL SPECIFICATIONS – TS68000 to 6800 PERIPHERAL
(V_{CC} = 5 Vdc ± 5 % ; GND = 0 Vdc ; T_A = T_L to T_H ; refer to figures 8.9 and 8.10)

N°	Symbol	Parameter	8 MHz		10 MHz		12.5 MHz		16 MHz		Unit
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
12(1)	t_{CLSH}	Clock Low to $\overline{AS}$, $\overline{DS}$ High	–	50	–	45	–	40	–	30	ns
18(1)	t_{CHRH}	Clock High to R/$\overline{W}$ High	0	55	0	45	0	40	0	40	ns
20(1)	t_{CHRL}	Clock High to R/$\overline{W}$ Low (write)	–	55	–	45	–	40	–	30	ns
23	t_{CLDO}	Clock Low to Data out Valid (write)	–	60	–	50	–	50	–	40	ns
27(2)	t_{CLDO}	Data in to Clock Low (setup time on read)	15	–	10	–	10	–	10	–	ns
29	t_{SHDII}	$\overline{AS}$, $\overline{DS}$ high to Data in Invalid (hold time on read)	0	–	0	–	0	–	0	–	ns
40	t_{CLVML}	Clock Low to $\overline{VMA}$ Low	–	70	–	70	–	70	–	50	ns
41	t_{CLET}	Clock Low to E Transistion	–	55	–	45	–	35	–	35	ns
42	$t_{Er,f}$	E Output Rise and Fall Time	–	25	–	15	–	15	–	15	ns
43	t_{VMLEH}	$\overline{VMA}$ Low to E high	200	–	150	–	90	–	70	–	ns
44	t_{SHVPH}	$\overline{AS}$, $\overline{DS}$ High to $\overline{VPA}$ High	0	120	0	90	0	70	0	50	ns
45	t_{ELCAI}	E Low to Control, Address Bus Invalid (address hold time)	30	–	10	–	10	–	10	–	ns
47(2)	t_{ASI}	Asynchronous Input Setup Time	20	–	20	–	10	–	10	–	ns
49(3)	t_{SHEL}	$\overline{AS}$, $\overline{DS}$ High to E Low	– 70	70	– 55	55	– 45	45	– 35	+ 35	ns
50	t_{EH}	E Width High	450	–	350	–	280.	–	210	–	ns
51	t_{EL}	E Width low	700	–	550	–	440	–	330	–	ns
54	t_{ELDOI}	E Low to Data out Invalid	30	–	20	–	15	–	10	–	ns

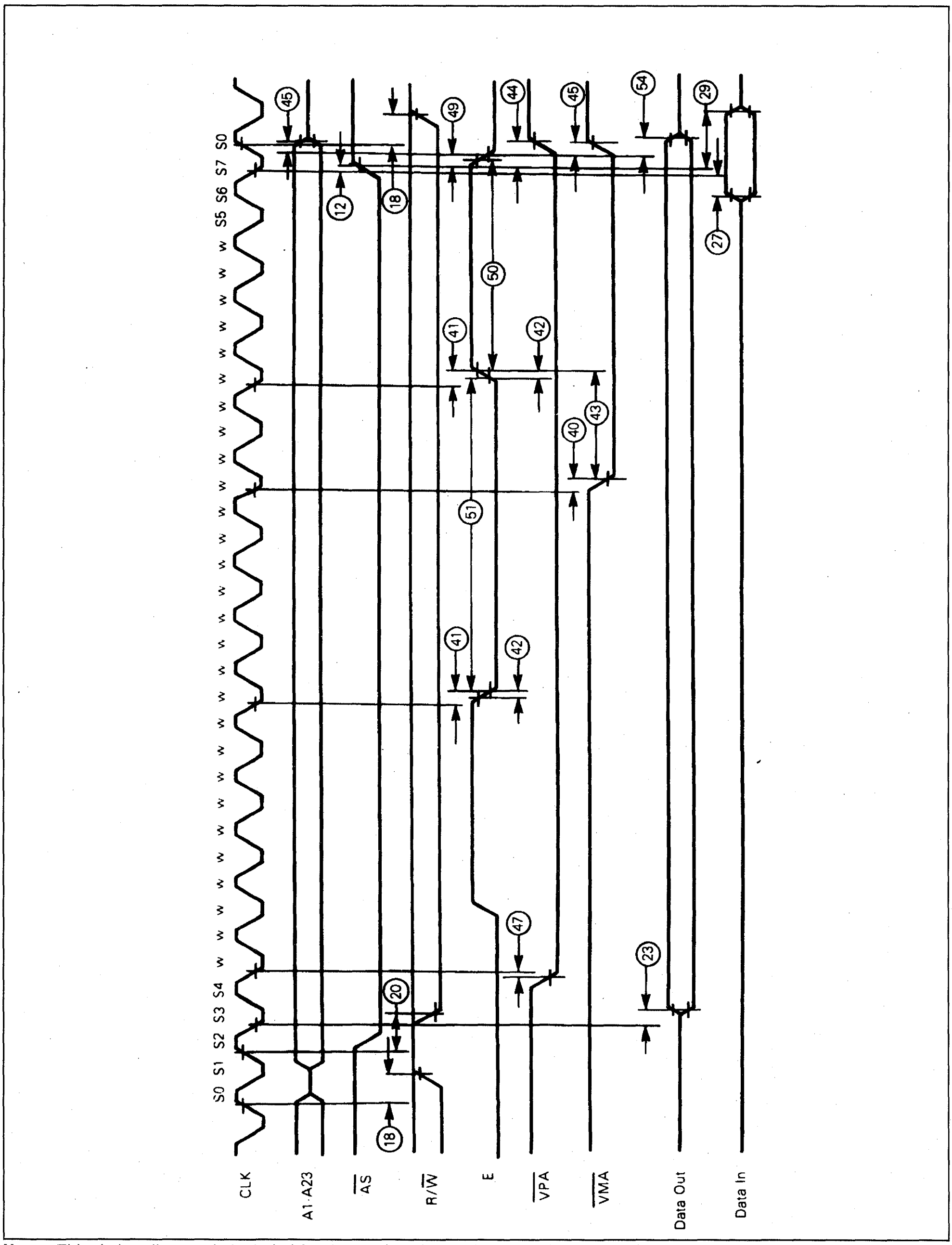
Notes : 1. For a loading capacitance of less than or equal to 50 pF, subtract 5 nanoseconds from the value given in the maximum columns.
2. If the asynchronous setup time (47) requirements are satisfied, the $\overline{DTACK}$ low-to-data setup time (31) requirement can be ignored. The data must only satisfy the date-in clock-low setupt time (27) for the following cycle.
3. The falling edge of S6 triggers both the negation of the strobes (AS, and x DS) 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.

Figure 8.9 : TS68000 to 6800 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 best case possibly attainable.

Figure 8.10 : TS68000 to 6800 Peripheral Timing Diagram – Worst Case.


CLK
A1-A23
AS
R/W
E
VPA
VMA
Data Out
Data In
S0 S1 S2 S3 S4 w w w w w w w w w w w w w w w w w w w w w w w w w w w w w S5 S6 S7 S0
12
18
20
23
27
29
40
41
42
43
44
45
47
49
50
51
54

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.

8.10. AC ELECTRICAL SPECIFICATIONS – BUS ARBITRATION
(V_{CC} = 5 Vdc ± 5 % ; GND = 0 Vdc ; $T_A = T_L$ to T_H ; see figures 8.11, 8.12 and 8.13)

N°	Symbol	Parameter	8 MHz		10 MHz		12.5 MHz		16 MHz		Unit
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
7	t_{CHADZ}	Clock high to Address, Data Bus High Impedance	–	70	–	60	–	50	–	50	ns
16	t_{CHCZ}	Clock High to Control Bus High Impedance	–	80	–	70	–	60	–	50	ns
33	t_{CHGL}	Clock High to $\overline{BG}$ Low	–	60	–	50	–	40	–	40	ns
34	t_{CHGH}	Clock High to $\overline{BG}$ High	–	60	–	50	–	40	–	40	ns
35	t_{BRLGL}	$\overline{BR}$, Low to $\overline{BG}$ Low	1.5	90ns + 3.5	1.5	80ns + 3.5	1.5	60ns + 3.5	1.5	60ns + 3.5	Clk Per.
36(1)	t_{BKHGH}	$\overline{BR}$, High to $\overline{BG}$ High	1.5	90ns + 3.5	1.5	80ns + 3.5	1.5	70ns + 3.5	1.5	60ns + 3.5	Clk Per.
37	T_{GALGH}	$\overline{BGACK}$ Low to $\overline{BG}$ High	1.5	90ns + 3.5	1.5	80ns + 3.5	1.5	70ns + 3.5	1.5	60ns + 3.5	Clk Per.
37A(2)	t_{GALBRH}	$\overline{BGACK}$ Low to $\overline{BG}$ High	20	1.5 Clocks	20	1.5 Clocks	20	1.5 Clocks	10	1.5 Clocks	ns
38	t_{GLZ}	$\overline{BG}$ Low to Control, Address, Data Bus High Impedance ($\overline{AS}$ High)	–	80	–	70	–	60	–	50	ns
39	T_{GH}	$\overline{BG}$ Width High	1.5	–	1.5	–	1.5	–	1.5	–	Clk Per.
46	t_{GAL}	$\overline{BGACK}$ Width Low	1.5	–	1.5	–	1.5	–	1.5	–	Clk Per.
47(3)	t_{ASI}	Asynchronous Input Setup Time	20	–	20	–	10	–	10	–	ns
57	t_{GABD}	$\overline{BGACK}$ High to Control Bus Driven AS, UDS, LDS	1.5	–	1.5	–	1.5	–	1.5	–	Clk Per.
		FC_x, R/$\overline{W}$, $\overline{VMA}$	1		1		1		1		
58(1)	t_{RHBD}	$\overline{BR}$ 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.
3. If the asynchronous setup time (47) requirements are satisfied, the $\overline{DTACK}$ low-to-data setup time (31) requirement can ban ign red. The data must only satisfy the date-in clock-low setup time (27) for the following cycle.

Figures 8.11, 8.12, and 8.13 depict the three bus arbitration cases that can arise. Figure 8.11 shows the timing where $\overline{AS}$ is negated when the processor asserts BG (Idle Bus Case). Figure 8.12 shows the timing where $\overline{AS}$ is asserted when the processor asserts $\overline{BG}$ (Active Bus Case). Figure 8.13 shows the timing where more than one bus master are requesting the bus. Refer to **4.2.2. Bus Arbitration** for a complete discussion of bus arbitration.

The waveforms shown in figures 8.11, 8.12, and 8.13 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 8.11 : Bus Arbitration Timing Diagram – Idle Bus Case.

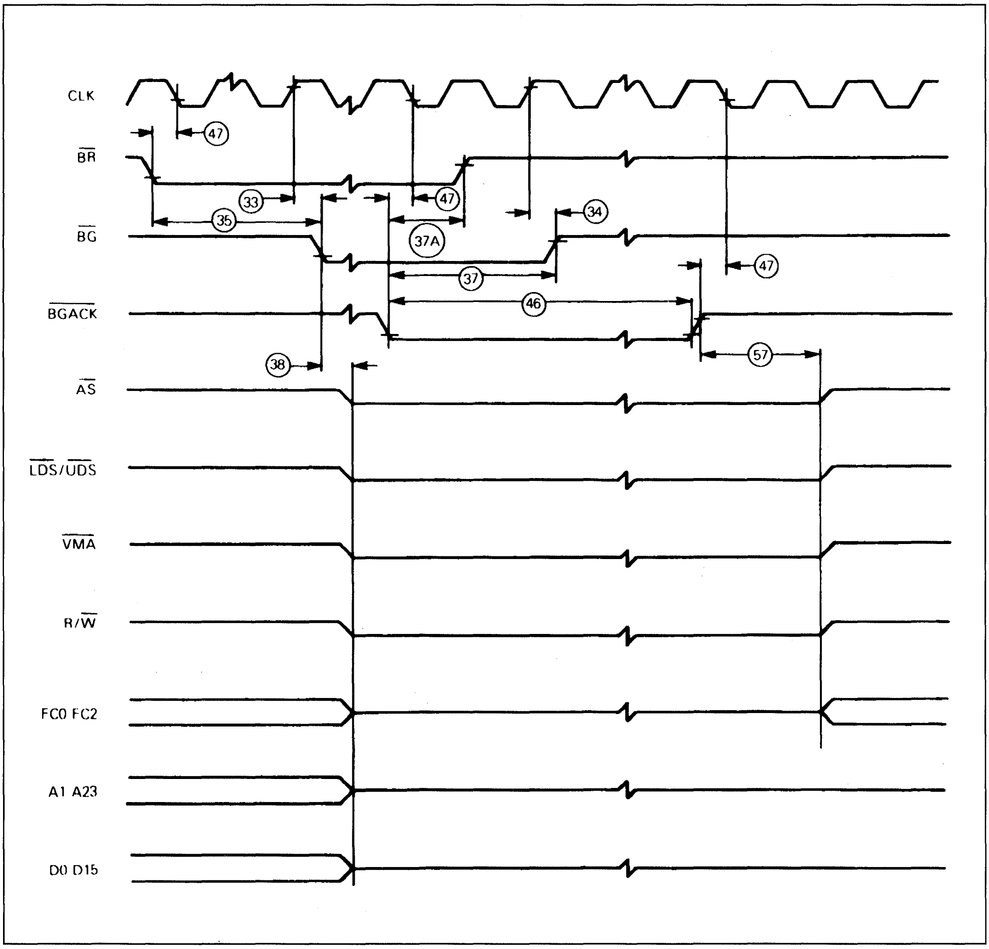


Figure 8.12 : Bus Arbitration Timing Diagram – Active Bus Case.

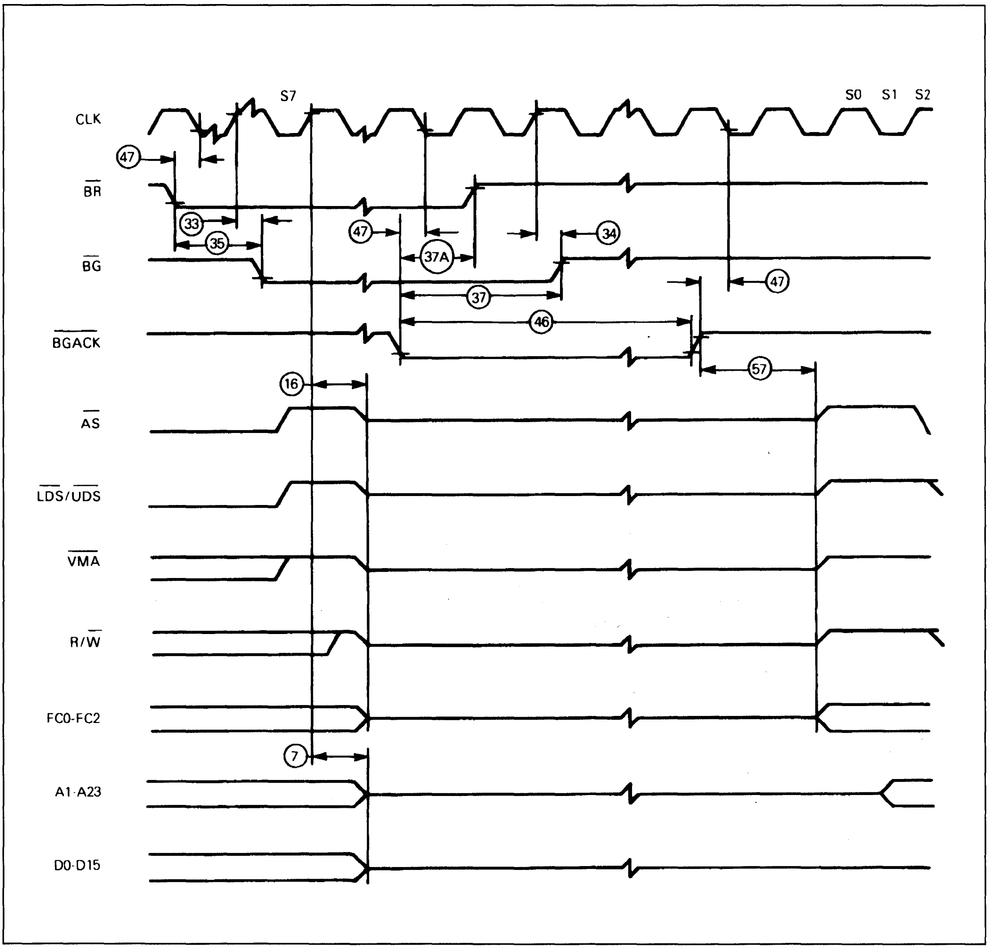
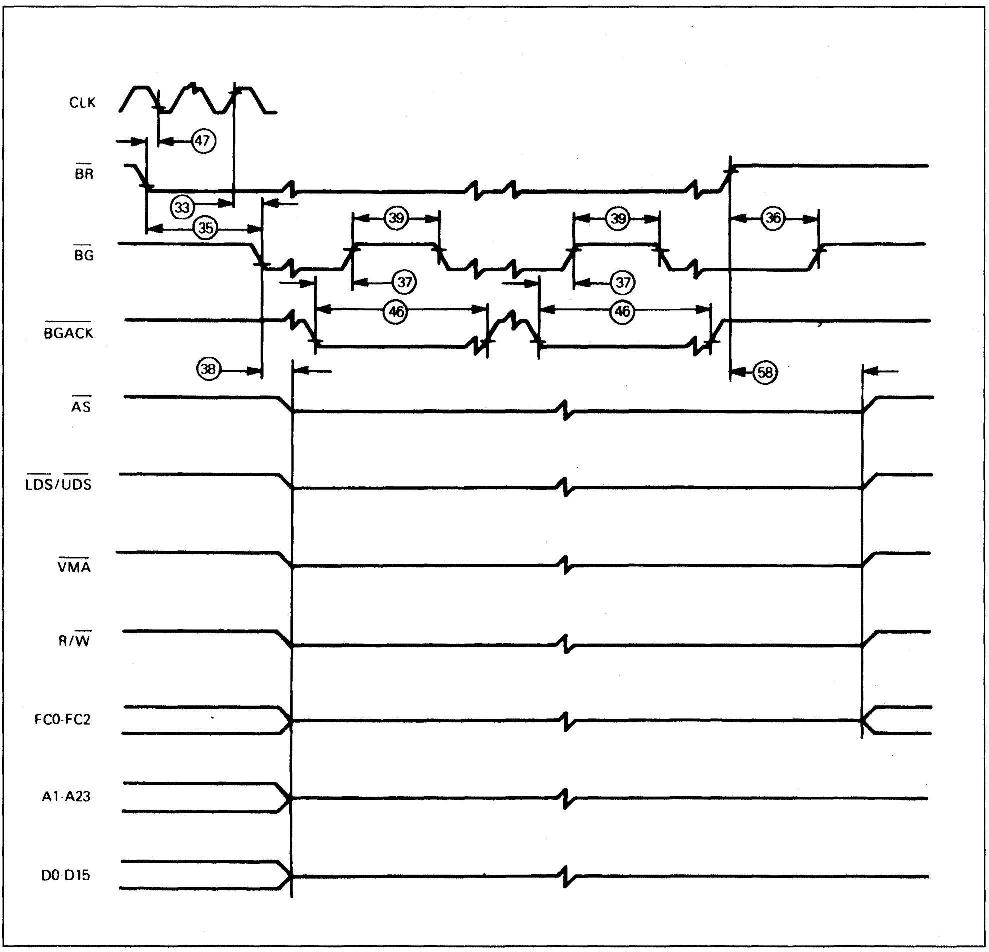


Figure 8.13 : Bus Arbitration Timing Diagram – Multiple Bus Requests.



SECTION 9

ORDER CODES

This section contains detailed information to be used as a guide when ordering the TS68000

9.1. STANDARD VERSIONS

Part Number	Frequency (MHz)	Temperature Range	Package Type
TS68000 CP8	8.0	0°C to +70°C	Plastic DIL
TS68000 VP8	8.0	−40°C to +85°C	P Suffix
TS68000 CP10	10.0	0°C to +70°C	
TS68000 VP10	10.0	−40°C to +85°C	
TS68000 CP12	12.5	0°C to +70°C	
TS68000 CP16	16.0	0°C to +70°C	
TS68000 CFN8	8.0	0°C to +70°C	PLCC
TS68000 VFN8	8.0	−40°C to +85°C	FN Suffix
TS68000 CFN10	10.0	0°C to +70°C	
TS68000 VFN10	10.0	−40°C to +85°C	
TS68000 CFN12	12.5	0°C to +70°C	
TS68000 CFN16	16.0	0°C to +70°C	
TS68000 CR8	8.0	0°C to +70°C	Pin Grid Array
TS68000 CR10	10.0	0°C to +70°C	R Suffix
TS68000 CR12	12.5	0°C to +70°C	
TS68000 CR16	16.0	0°C to +70°C	

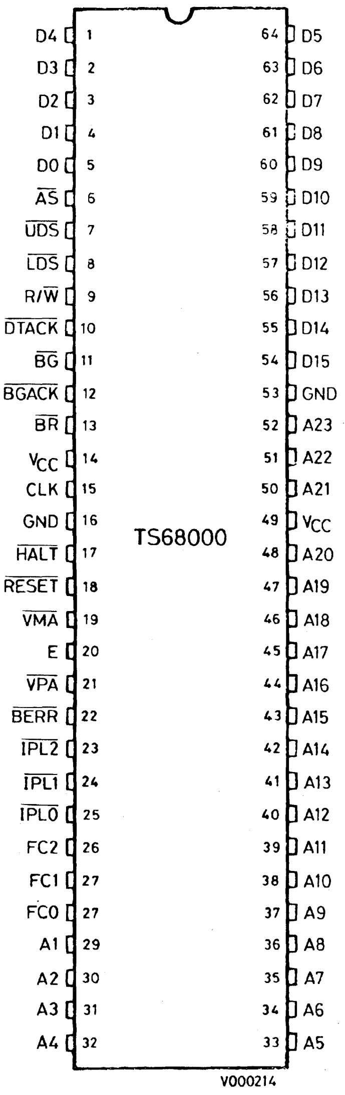
SECTION 10

MECHANICAL DATA

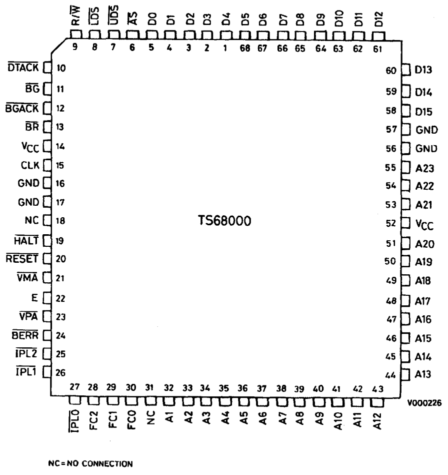
This section contains the pin assignments and package dimensions for the 64–pin dual–in–line, the 68–terminal chip carrier (LCCC), the 68–pin grid array, and the 68–pin quad pack (PLCC), versions of the TS68000.

10.1. PIN ASSIGNMENTS

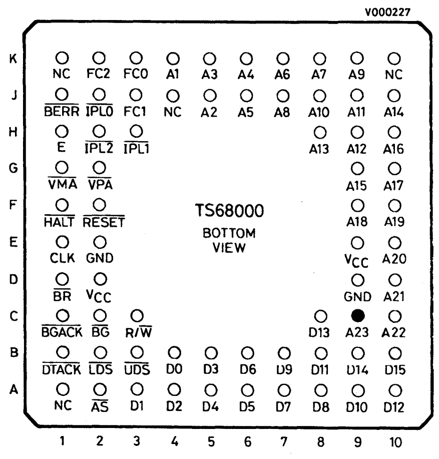
64–Pin Dual–in–Line Package



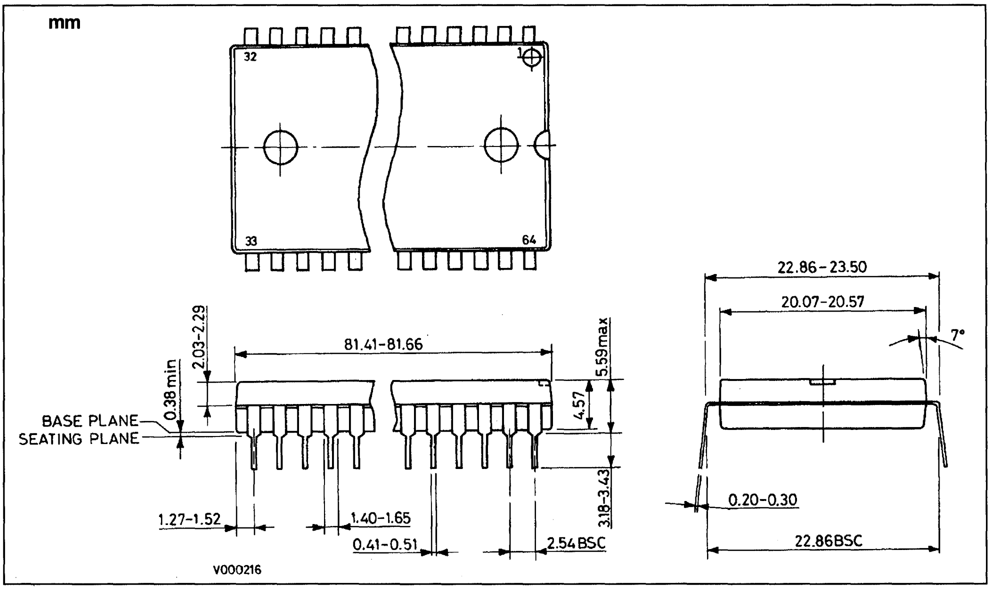
68–Pin Quad Pack (PLCC)

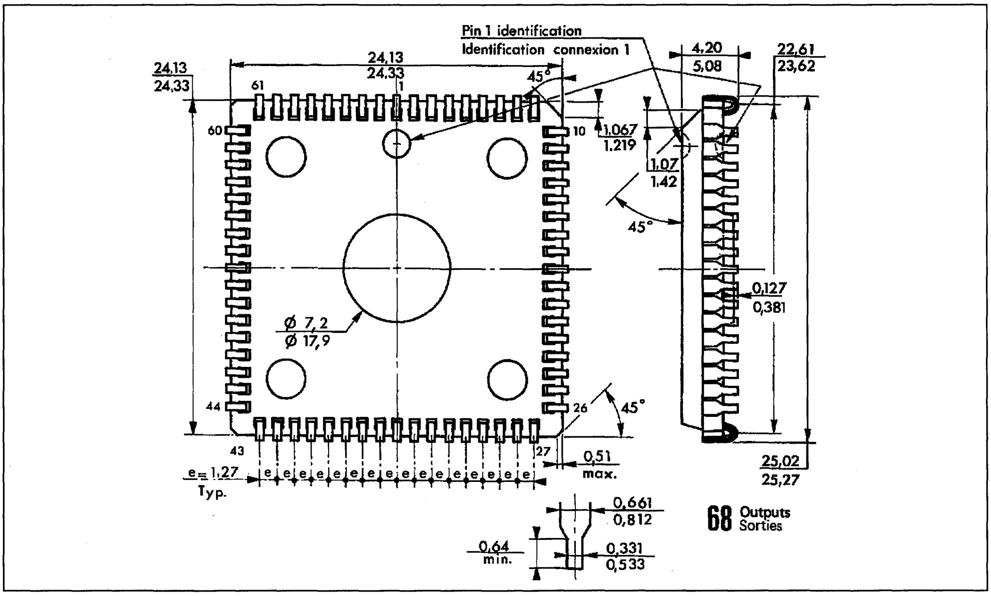


68–Pin Grid Array

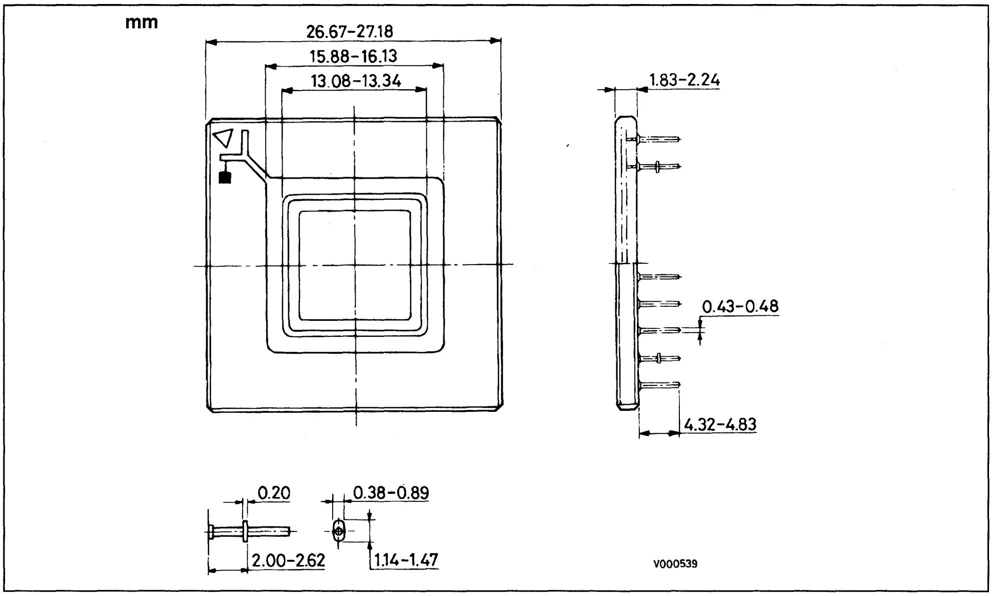

V000227
K: NC FC2 FC0 A1 A3 A4 A6 A7 A9 NC
J: BERR IPL0 FC1 NC A2 A5 A8 A10 A11 A14
H: E IPL2 IPL1 A13 A12 A16
G: VMA VPA A15 A17
F: HALT RESET A18 A19
E: CLK GND VCC A20
D: BR VCC GND A21
C: BGACK BG R/W D13 A23 A22
B: DTACK LDS UDS D0 D3 D6 D9 D11 D14 D15
A: NC AS D1 D2 D4 D5 D7 D8 D10 D12
1 2 3 4 5 6 7 8 9 10
TS68000
BOTTOM VIEW

10.2. PACKAGE DIMENSIONS





10.2. PACKAGE DIMENSIONS (continued)



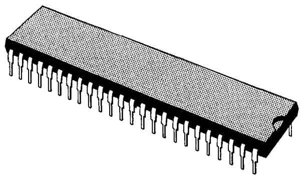
This is advanced information and specifications are subject to change without notice. Please inquire with our sales offices about the availability to the different packages.

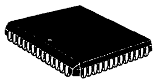
SGS-THOMSON
MICROELECTRONICS

TS68008

8/16–BIT MICROPROCESSOR WITH 8–BIT DATA BUS

- 17 32-BIT DATA AND ADDRESS REGISTERS
- 56 BASIC INSTRUCTION TYPES
- EXTENSIVE EXCEPTION PROCESSING
- MEMORY MAPPED I/O
- 14 ADDRESSING MODES
- 1 MBYTE LINEAR ADDRESSING SPACE
- COMPLETE CODE COMPATIBILITY WITH THE TS68000


P
(Plastic Package)

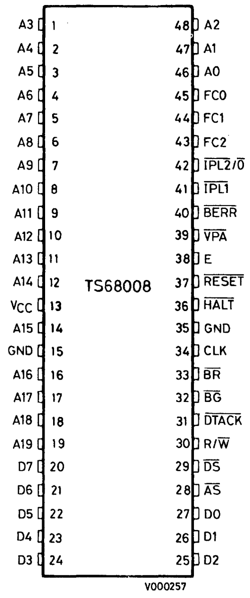

FN
(PLCC 52)

DESCRIPTION

The TS68008 is a member of the TS68000 family of advanced microprocessors. This device allows the design of cost effective systems using 8-bit data buses while providing the benefits of a 32-bit microprocessor architecture. The performance of the TS68008 is greater than any 8-bit microprocessor and superior to several 16-bit microprocessors.

A system implementation based on an 8-bit data bus reduces system cost in comparison to 16-bit systems due to a more effective use of components and the fact that byte-wide memories and peripherals can be used much more effectively. In addition, the non-multiplexed address and data buses eliminate the need for external demultiplexers, thus further simplifying the system.

PIN CONNECTIONS



SECTION1

INTRODUCTION

The TS68008 is a member of the 68000 Family of advanced microprocessors. This device allows the design of cost effective systems using 8-bit data buses while providing the benefits of a 32-bit microprocessor architecture. The performance of the TS68008 is greater than any 8-bit microprocessor and superior to several 16-bit microprocessors.

The resources available to the TS68008 user consist of the following :

- 17 32-Bit Data and Address Registers
- 56 Basic Instruction Types
- Extensive Exception Processing
- Memory Mapped I/O
- 14 Addressing Modes
- Complete Code Compatibility with the TS68000

A system implementation based on an 8-bit data bus reduces system cost in comparison to 16-bit systems due to a more effective use of components and the fact that byte-wide memories and peripherals can be used much more effectively. In addition, the non-multiplexed address and data buses eliminate the need for external demultiplexers, thus further simplifying the system.

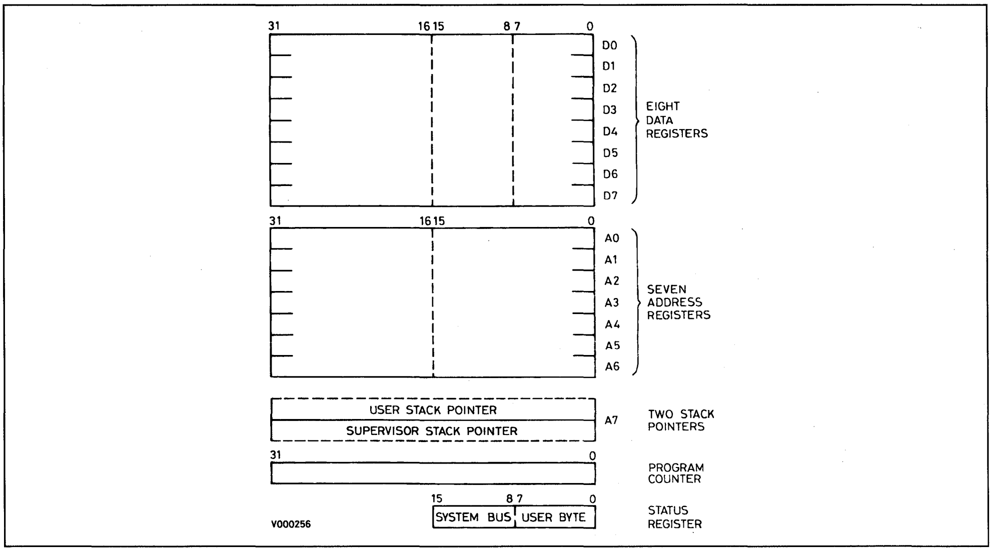
The TS68008 has full code compatibility (source and object) with the TS68000 which allows programs to be run on either MPU, depending on performance requirements and cost objectives.

The TS68008 is available in a 48-pin dual-in-line package (plastic or ceramic) and a 52-pin quad plastic package. Among the four additional pins of the 52-pin package, two additional address lines are included beyond the 20 address lines of the 48-pin package. The address range of the TS68008 is one of four megabytes with the 48- or 52-pin package, respectively.

The large non-segmented linear address space of the TS68008 allows large modular programs to be developed and executed efficiently. A large linear address space allows program segment sizes to be determined by the application rather than forcing the designer to adopt an arbitrary segment size without regard to the application's individual requirements.

The programmer's model is identical to that of the TS68000, as shown in figure 1.1, with seventeen 32-bit registers, a 32-bit program counter, and a 16-bit status register. 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), the user stack pointer (A7), and the system stack pointer (A7') may be used as software stack pointers and base address registers. In addition, the registers may be used for some simple word and long word data operations. All of the 17 registers may be used as index registers.

Figure 1.1 : Progamming Model.



While all of the address registers can be used to create stacks and queues, the A7 address register, by convention, is used as the system stack pointer. Supplementing this convention is another address register, A7', also referred to as the system stack pointer. This powerful concept allows the supervisor mode and user mode of the TS68008 to each have their own system stack pointer (consistently referred to as SP) without needing to move pointers for each context of use when the mode is switched.

The system stack pointer (SP) is either the supervisor stack pointer (A7' ≡ SSP) or the user stack point (A7 ≡ USP), depending on the state of the S bit in the status register. If the S bit is set, indicating that the processor is in the supervisor state, when the SSP is the active system stack pointer and the USP is not used. If the S bit is clear, indicating that the processor is in the user state, then the USP is the active system stack pointer and the SSP is protected from user modification.

The status register, shown in figure 1.2, may be considered as two bytes : the user byte and the system byte. The user byte contains five bits defining the overflow (V), zero (Z), negative (N), carry (C), and extended (X) condition codes. The system byte contains five defined bits. Three bits are used to define the current interrupt priority ; any interrupt level higher than the current mask level will be recognized. (Note that level 7 interrupts are non-maskable - that is, level 7 interrupts are always processed). Two additional bits indicate whether the processor is in the trace (T) mode and/or in the supervisor (S) state.

1.1. 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.

Most instructions can use any of the 14 addressing modes which are listed in table 1.1. These addressing modes consist of six basic types :

- Register Direct
- Register Indirect
- Absolute
- Program Counter Relative
- Immediate
- Implied

The register indirect addressing modes also have the capability to perform postincrementing, predecrementing, offsetting, and indexing. The program counter relative mode may be used in combination with indexing and offsetting for writing relocatable programs.

Figure 1.2 : Status Register

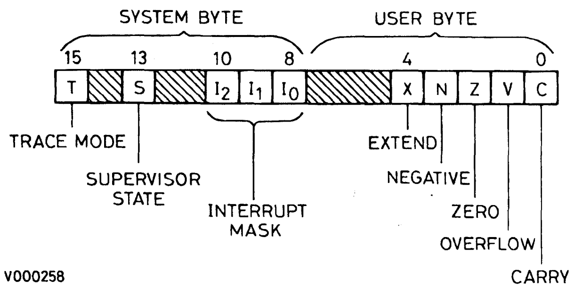


Table 1.1 : Addressing Modes.

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 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
( ) = Contents of
d_8 = 8-Bit Offset (Displacement)
d_{16} = 16-Bit Offset (Displacement)
#xxx = Immediate Data

1.2. INSTRUCTION SET OVERVIEW

The TS68008 is completely code compatible with the TS68000. This means that programs developed for the TS68000 will run on the TS68008 and vice versa. This applies equally to either source code or object code.

The instruction set was designed to minimize the number of mnemonics remembered by the programmer. To further reduce the programmer's burden, the addressing modes are orthogonal.

The instruction set, shown in table 1.2, forms a set of programming tools that include all processor functions to perform data movement, integer arithmetic, logical operations, shift and rotate operations, bit manipulation, BCD operations, and both program and system control. Some additional instructions are variations or subsets of these and appear in table 1.3.

Table 1.2 : Instruction Set Summary.

Mnemonic	Description
ABCD ADD AND ASL ASR	Add Decimal with Extend Add Logical And Arithmetic Shift Left Arithmetic Shift Right
Bcc BCHG BCLR BRA BSET BSR BTST	Branch Conditionally Bit Test and Change Bit Test and Clear Branch always Bit Test and Set Branch to Subroutine Bit Test
CHK CLR CMP	Check Register against Bounds Clear Operand Compare
DBcc DIVS DIVU	Test Condition, Decrement and Branch Signed Divide Unsigned Divide
EOR EXG EXT	Exclusive Or Exchange Registers Sign Extend
JMP JSR	Jump Jump to Subroutine
LEA LINK LSL LSR	Load Effective Address Link Stack Logical Shift Left Logical Shift Right
MOVE MULS MULU	Move Signed Multiply Unsigned Mulitply
NBCD NEG NOP NOT	Negate Decimal with Extend Negate No Operation One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET ROL ROR ROXL ROXR RTE RTR RTS	Reset External Devices Rotate Left without Extend Rotate Right without Extend Rotate Left with Extend Rotate Right with Extend Return from Exception Return and Restore Return from Subroutine
SBCD Scc STOP SUB SWAP	Subtract Decimal with Extend Set Conditional Stop Subtract Swap Data Register Halves
TAS TRAP TRAPV TST	Test and Set Operand Trap Trap on Overflow Test
UNLK	Unlink

Table 1.3 : Variations of Instruction Types.

Instruction Type	Variation	Description
ADD	ADD ADDA ADDQ ADDI ADDX	Add Add Address Add Quick Add Immediate Add with Extend
AND	AND ANDI ANDI to CCR ANDI to SR	Logical And And Immediate And Immediate to Condition Codes And Immediate to Status Register
CMP	CMP CMPA CMPM CMPI	Compare Compare Address Compare Memory Compare Immediate
EOR	EOR EORI EORI to CCR EORI to SR	Exclusive Or Exclusive Or Immediate Exclusive Or Immediate to Condition Codes Exclusive Or Immediate to Status Register
MOVE	MOVE MOVEA MOVEC MOVEM MOVEP MOVEQ MOVES MOVE from SR MORE to SR MOVE from CCR MOVE to CCR MOVE USP	Move Move Address Move Control Register Move Multiple Registers Move Peripheral Data Move Quick Move Alternate Address Space Move from Status Register Move to Status Register Move from Condition Codes Move to Condition Codes Move User Stack Pointer
NEG	NEG NEGX	Negate Negate with Extend
OR	OR ORI ORI to CCR ORI to SR	Logical Or Or Immediate Or Immediate to Condition Codes Or Immediate to Status Register
SUB	SUB SUBA SUBI SUBQ SUBX	Subtract Subtract Address Subtract Immediate Subtract Quick Subtract with Extend

SECTION 2

DATA ORGANIZATION AND ADDRESSING CAPABILITIES

This section describes the registers and data organization of the TS68008.

2.1. OPERAND SIZE

Operand sizes are defined as follows : a byte equals eight bits, a word equals 16 bits (two bytes), and a long word equals 32 bits (four bytes). 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. When fetching instructions, the TS68008 always fetches pairs of bytes (words) thus guaranteeing compatibility with the TS68000.

2.2. 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.

2.2.1. DATA REGISTERS. Each data register is 32 bits wide. Byte operands occupy the low order eight 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.

2.2.2. 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 byte sized operand. 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.

2.3. DATA ORGANIZATION IN MEMORY

The data types supported by the TS68008 are : bit data, integer data of 8, 16, or 32 bits, and 32-bit addresses. Figure 2.1 shows the organization of these data types in memory.

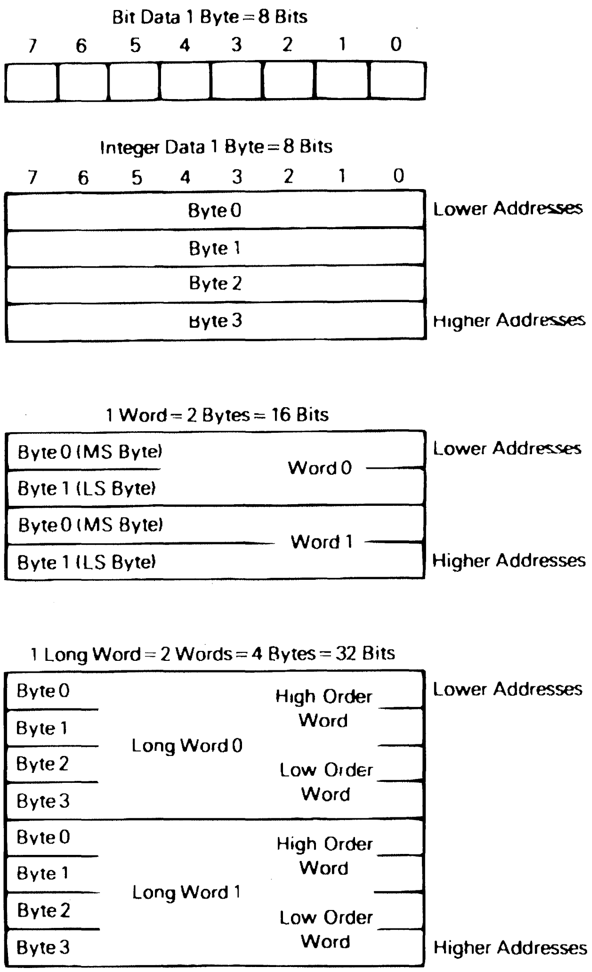
2.4. ADDRESSING

Instructions for the TS68008 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 the instruction.
Effective Address	– use of the different effective address modes.
Implicit Reference	– the definition of certain instructions implies the use of specific registers.

Figure 2.1 : Memory Data Organization.



2.5. INSTRUCTION FORMAT

Instructions are from one to five words (two to ten bytes) in length as shown in figure 2.2. Instructions always start on a word boundary thus guaranteeing compatibility with the TS68000. 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.

2.6. PROGRAM/DATA REFERENCES

The TS68008 separates memory references into two classes : program references, and data references. Program references, as the name implies, are references to that section of memory containing the program being executed. Data references refer to that section of memory containing 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. The function codes are used to indicate the address space being accessed during a bus cycle.

2.7. 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.

2.8. EFFECTIVE ADDRESS

Most instructions specify the location of an operand by using the effective address field in the operation word. For example, figure 2.3 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 2.2. The effective address modes are

Figure 2.2 : Instruction Operation Word General Format.

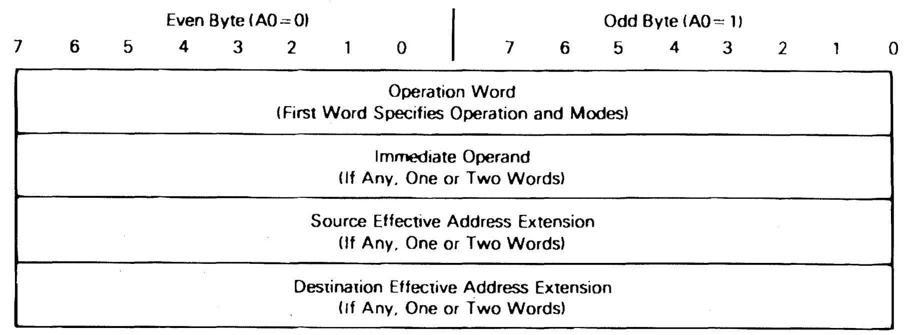
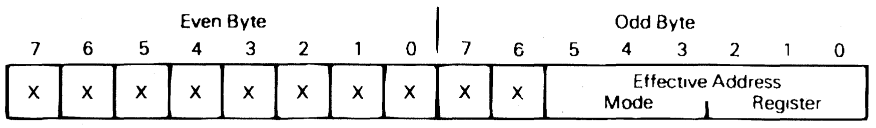

Even Byte (A0 = 0)
Odd Byte (A0 = 1)
7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
Operation Word
(First Word Specifies Operation and Modes)
Immediate Operand
(If Any, One or Two Words)
Source Effective Address Extension
(If Any, One or Two Words)
Destination Effective Address Extension
(If Any, One or Two Words)

Figure 2.3 : Single–Effective–Address Instruction Operation Word.



2.8. EFFECTIVE ADDRESS (continued)

grouped into three categories : register direct, memory addressing, and special.

2.8.1. REGISTER DIRECT MODES. These effective addressing modes specify that the operand is in one of sixteen multifunction registers.

2.8.1.1. Data Register Direct.

The operand is in the data register specified by the effective address register field.

2.8.1.2. Address Register Direct.

The operand is in the address register specified by the effective address register field.

2.8.2. MEMORY ADDRESS MODES. These effective addressing modes specify that the operand is in memory and provide the specific address of the operand.

2.8.2.1. 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.

2.8.2.2. 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 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.

2.8.2.3. Address Register Indirect With Predecrement.

The address of the operand will be in the address register specified by the register field. Before the address register is used for operand access, 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.

2.8.2.4. Address Register Indirect With Displacement.

This address 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.

2.8.2.5. Address Register Indirect With Index.

This address 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.

2.8.3. SPECIAL ADDRESS MODES. The special address modes use the effective address register field to specify the special addressing mode instead of a register number.

2.8.3.1. Absolute Short Address.

This address 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.

2.8.3.2. Absolute Long Address.

This address 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.

2.8.3.3. Program Counter With Displacement.

This address 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.

2.8.3.4. Program Counter With Index.

This address mode requires one word of extension. This 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.

2.8 EFFECTIVE ADDRESS (continued)

2.8.3.5. Immediate Data.

This address 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.

2.8.3.6. 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	EORI to SR	MOVE to CCR
ANDI to SR	ORI to CCR	MOVE to SR
EORI to CCR	ORI to SR	MOVE from SR

2.9. EFFECTIVE ADDRESS ENCODING SUMMARY

Table 2.1 is a summary of the effective addressing modes discussed in the previous paragraphs.

2.10. 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 is not used. 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 2.1 : 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

SECTION 3

INSTRUCTION SET SUMMARY

This section contains an overview of the form and structure of the TS68008 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 Coded 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.

3.1. 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 3.1 is a summary of the data movement operations.

Table 3.1 : Data Movement Operations.

Instruction	Operand Size	Operation
EXG	32	$R_X \leftrightarrow R_Y$
LEA	32	EA → An
LINK	–	An → – (SP) SP → An SP + Displacement → SP
MOVE	8, 26, 32	(EA)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   - = indirect with predecrement
d = destination   + = indirect with postdecrement
[ ] = bit number   # = immediate data

3.2. 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 3.2 is a summary of the integer arithmetic operations.

Table 3.2 : 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) – #xxxx (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	O – (EA) – X → (EA)
SUB	8, 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
– ( ) = indirect with predecrement
( ) + = indirect with postdecrement
# = immediate data

3.3. 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 3.3 is a summary of the logical operations.

Table 3.3 : Logical Operations.

Instruction	Operand Size	Operation
AND	8, 16, 32	Dn Λ (EA) → Dn (EA) Λ Dn → (EA) (EA) Λ #xxx → (EA)
OR	8, 16, 32	Dn V (EA) → Dn (EA) V Dn → (EA) (EA) V #xxx → (EA)
EOR	8, 16, 32	(EA) ⊕ Dy → (EA) (EA) ⊕ #xxx → (EA)
NOT	8, 16, 32	~ (EA) → (EA)

Notes : # = immediate data
~ = invert
Λ = logical AND
V = logical OR
⊕ = logical exclusive OR

3.4. 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 and provide single-bit shifts or rotates.

Table 3.4 is a summary of the shift and rotate operations.

Table 3.4 : Shift and Rotate Operations.

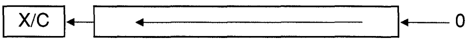
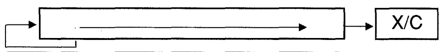
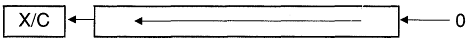
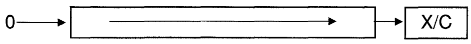
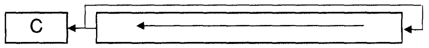
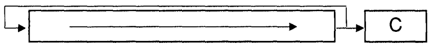
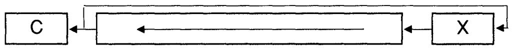
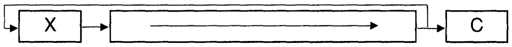
Instruction	Operand Size	Operation
ASL	8, 16, 32	
ASR	8, 16, 32	
LSL	8, 16, 32	
LSR	8, 16, 32	

Table 3.4 : Shift and Rotate Operations (continued).

Instruction	Operand Size	Operation
ROL	8, 16, 32	
ROR	8, 16, 32	
ROXL	8, 16, 32	
ROXR	8, 16, 32	

3.5. 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 3.5 is a summary of the bit manipulation operations. (Z is bit 2 of the status register).

Table 3.5 : Bit Manipulation Operations.

Instruction	Operand Size	Operation
BTST	8, 32	~ bit of (EA) $\rightarrow$ Z
BSET	8, 32	~ bit of (EA) $\rightarrow$ Z 1 $\rightarrow$ bit of EA
BCLR	8, 32	~ bit of (EA) $\rightarrow$ Z 0 $\rightarrow$ bit of EA
BCHG	8, 32	~ bit of (EA) $\rightarrow$ Z ~ bit of (EA) $\rightarrow$ bit of EA

Note : ~ = Invert

3.6. 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 3.6 is a summary of the binary coded decimal operations.

Table 3.6 : 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	$0 - (EA)_{10} - X \rightarrow (EA)$

3.7. PROGRAM CONTROL OPERATIONS

Program control operations are accomplished using a series of conditional and unconditional branch instructions, jump instructions, and return instructions. These instructions are summarized in table 3.7.

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 3.7 : 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 condtions)
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

3.8. 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 3.8.

Table 3.8 : 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	Chek 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

SECTION 4

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.

4.1. SIGNAL DESCRIPTION

The TS68008 is available in two package sizes (48-pin and 52-pin). The additional four pins of the 52-pin quad package allow for additional signals : A20, A21, BGACK, and IPL2.

Throughout this document, references to the address bus pins (A0-A19) and the interrupt priority level pins (IPL0/IPL2, IPL1) refer to A0-A21 and IPL0, IPL1, and IPL2 for the 52-pin version of the TS68008.

The input and output signals can be functionally organized into the groups shown in figure 4.1(a) for the 48-pin version and in figure 4.1(b) for the 52-pin version. The following paragraphs provide a brief description of the signals and a reference (if applicable) to other paragraphs that contain more information about the function being performed.

4.1.1. ADDRESS BUS (48-PIN : A0 THROUGH A19. 52-PIN : A0 THROUGH A21). This unidirectional three-state bus provides the address for bus operation during all cycles except interrupt acknowledge cycles. During interrupt acknowledge cycles, address lines A1, A2, and A3 provide information about what level interrupt is being serviced while address lines A0 and A4 through A19 (A21) are all driven high.

4.1.2. DATA BUS (D0 THROUGH D7). This 8-bit, bidirectional, three-state bus is the general purpose data path. During an interrupt acknowledge cycle, the external device supplies the vector number on data lines D0-D7.

4.1.3. ASYNCHRONOUS BUS CONTROL. Asynchronous data transfers are handled using the following control signals : address strobe, read/write, data strobe, and data transfer acknowledge. These signals are explained in the following paragraphs.

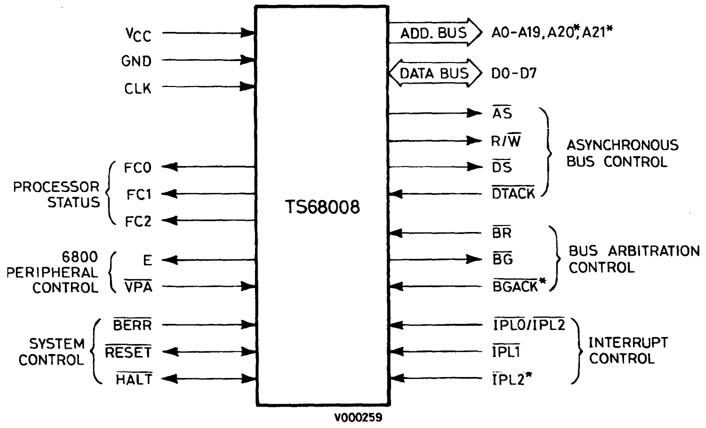
4.1.3.1. Address Strobe ($\overline{AS}$).

This three-state signal indicates that there is a valid address on the address bus. It is also used to "lock" the bus during the read-modify-write cycle used by the test and set (TAS) instruction.

4.1.3.2. Read/Write (R/$\overline{W}$).

This three-state signal defines the data bus transfer as a read or write cycle. The R/$\overline{W}$ signal also works in conjunction with the data strobe as explained in the following paragraph.

Figure 4.1 : Input and Output Signals.



* 52 PIN VERSION ONLY

4.1.3.3. Data Strobe ($\overline{\text{DS}}$).

This three-state signal controls the flow of data on the data bus as shown in table 4.1. When the R/$\overline{\text{W}}$ line is high, the processor will read from the data bus as indicated. When the R/$\overline{\text{W}}$ line is low, the processor will write to the data bus as shown.

4.1.3.4. Data Transfer Acknowledge ($\overline{\text{DTACK}}$).

This input indicates that the data transfer is completed. When the processor recognizes $\overline{\text{DTACK}}$ during a read cycle, data is latched and the bus cycle is terminated. When $\overline{\text{DTACK}}$ is recognized during a write cycle, the bus cycle is terminated. (Refer to **4.4. Asynchronous Versus Synchronous Operation**).

4.1.4. BUS ARBITRATION CONTROL. The 48-pin TS68008 contains a simple two-wire arbitration circuit and the 52-pin TS68008 contains the full three-wire TS68000 bus arbitration control. Both versions are designed to work with daisy-chained networks, priority encoded networks, or a combination of these techniques. This circuit is used in determining which device will be the bus master device.

4.1.4.1. Bus Request ($\overline{\text{BR}}$).

This input is wire ORed with all other devices that could be bus masters. This device indicates to the processor that some other device desires to become the bus master. Bus requests may be issued at any time in a cycle or even if no cycle is being performed.

4.1.4.2. Bus Grant ($\overline{\text{BG}}$).

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.

4.1.4.3. Bus Grant Acknowledge ($\overline{\text{BGACK}}$).

This input, available on the 52-pin version only, 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 master-ship.

Notes :

1. There is a two-clock interval straddling the transition of $\overline{\text{AS}}$ from the inactive state to the active state during which $\overline{\text{BG}}$ cannot be issued.
2. If an existing TS68000 system is retrofitted to use the TS68008, 48-pin version (using $\overline{\text{BR}}$ and $\overline{\text{BG}}$ only), the existing $\overline{\text{BR}}$ and $\overline{\text{BGACK}}$ signals should be ANDed and the resultant signal connected to the TS68008's $\overline{\text{BR}}$.

4.1.5. INTERRUPT CONTROL (48-PIN : $\overline{\text{IPL0/IPL2}}$, $\overline{\text{IPL1}}$. 52-PIN : $\overline{\text{IPL0}}$, $\overline{\text{IPL1}}$, $\overline{\text{IPL2}}$). These input pins indicate the encoded priority level of the device requesting an interrupt. The TS68000 and the 52-pin TS68008 MPUs use three pins to encode a range of 0-7 but, for the 48-pin TS68008 only two pins are available. By connecting the $\overline{\text{IPL0/IPL2}}$ pin to both the $\overline{\text{IPL0}}$ and $\overline{\text{IPL2}}$ inputs internally, the 48-pin encodes values of 0, 2, 5, and 7. Level zero is used to indicate that there are no interrupts pending and level seven is a non-maskable edge-triggered interrupt. Except for level seven, the requesting level must be greater than the interrupt mask level contained in the processor status register before the processor will acknowledge the request.

The level presented to these inputs is continually monitored to allow for the case of a requesting level that is less than or equal to the processor status register level to be followed by a request that is greater than the processor status register level. A satisfactory interrupt condition must exist for two successive clocks before triggering an internal interrupt request. An interrupt acknowledge sequence is indicated by the function codes.

4.1.6. 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 signals are explained in the following paragraphs.

Table 4.1 : Data Strobe Control of Data Bus.

$\overline{\text{DS}}$	R/$\overline{\text{W}}$	D0 - D7
1	–	No Valid Data
0	1	Valid Data Bits 0-7 (read cycle)
0	0	Valid Data Bits 0-7 (write cycle)

4.1.6.1. Bus Error ($\overline{\text{BERR}}$).

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 me mory management unit, or
- 4. various 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 **4.2.3. Bus Error and Halt Operation** for a detailed description of the interaction which is summarized below.

$\overline{\text{BERR}}$	$\overline{\text{HALT}}$	**Resulting Operation**
High	High	Normal operation
High	Low	Single bus cycle operation
Low	High	Bus error - exception processing
Low	Low	Bus error - re-run current cycle

4.1.6.2. Reset ($\overline{\text{RESET}}$).

This bidirectional signal line acts to reset (start a system initialization sequence) the processor in response to an external $\overline{\text{RESET}}$ 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 $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$ signals applied at the same time. Refer to **4.2.4. Reset Operation** for further information.

4.1.6.3. Halt ($\overline{\text{HALT}}$).

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 **4.2.3. Bus Error and Halt Operation** for additional information about the interaction between the halt and bus error signals.

When the processor has stopped executing instructions, such as in a double bus fault condition, the halt line is driven by the processor to indicate to external devices that the processor has stopped.

4.1.7. 6800 PERIPHERAL CONTROL. These control signals are used to allow the interfacing of synchronous 6800 peripheral devices with the asynchronous TS68008. These signals are explained in the following paragraphs.

The TS68008 does not supply a valid memory address (VMA) signal like that of the TS68000. The VMA signal indicates to the 6800 peripheral devices that there is a valid address on the address bus and that the processor is synchronized to the enable clock. This signal can be produced by a TTL circuit (see a sample circuit in figure 4.2). The $\overline{\text{VMA}}$ signal, in this circuit, only responds to a valid peripheral address (VPA) input which indicates that the peripheral is an 6800 Family device. Timing for this circuit is shown in figure 6.2.

The VPA decode shown in figure 4.2 is an active high decode indicating that address strobe ($\overline{\text{AS}}$) has been asserted and the address bus is addressing an 6800 peripheral. The $\overline{\text{VPA}}$ output of the circuit is used to indicate to the TS68008 that the data transfer should be synchronized with the enable (E) signal.

4.1.7.1. Enable (E).

This signal is the standard enable signal common to all 6800 type peripheral devices. The period for this output is 10 TS68008 clock periods (six clocks low, four clocks high).

4.1.7.2. Valid Peripheral Address ($\overline{\text{VPA}}$).

This input indicates that the device or region addressed is a 6800 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 **6.0 Interface with 6800 Peripherals.**

4.1.8. PROCESSOR STATUS (FC0, FC1, FC2). These function code outputs indicate the state (user

Figure 4.2 : External $\overline{\text{VMA}}$ Generation.

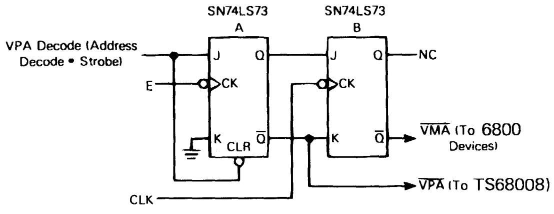


Table 4.2 : Function Code Outputs.

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

or supervisor) and the cycle type currently being executed, as shown in table 4.2. The information indicated by the function code outputs is valid whenever address strobe ($\overline{AS}$) is active.

4.1.9. 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 shall be a constant frequency.

4.1.10. V_{CC} and GND. Power is supplied to the processor using these two signals. V_{CC} is power and GND is the ground connection.

4.1.11. SIGNAL SUMMARY. Table 4.3 is a summary of all the signals discussed in the previous paragraphs.

Table 4.3 : Signal Summary.

Signal Name	Mnemonic	Input/Output	Active State	Hi-Z	
				On $\overline{HALT}$	On $\overline{BGACK}$
Address Bus	A0-A19	Output	High	Yes	Yes
Data Bus	D0-D7	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
Data Strobes	$\overline{DS}$	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	$\overline{IPLx}$	Input	Low	No	No
Bus Error	$\overline{BERR}$	Input	Low	No	No
Reset	$\overline{RESET}$	Input/Output	Low	No*	No*
Halt	$\overline{HALT}$	Input/Output	Low	No*	No*
Enable	E	Output	High	No	No
Valid Peripheral Address	$\overline{VPA}$	Input	Low	No	No
Function Code Output	FC0, FC1, FC2	Output	High	No	Yes
Clock	CLK	Input	High	No	No
Power Input	V_{CC}	Input	–	–	–
Ground	GND	Input	–	–	–

* Open Drain.
** 52-Pin Version Only.

4.2. 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.

4.2.1. DATA TRANSFER OPERATIONS. Transfer of data between devices involves the following leads :

- Address bus A0 through A19
- Data bus D0 through D7
- Control signals

The address and data buses are separate non-multiplexed parallel buses. Data transfer is accomplished with an asynchronous bus structure that uses handshakes to ensure the correct movement of data. 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 TS68008 for interlocked multiprocessor communications.

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 voltage is low or high. The term negate or negation is used to indicate that a signal is inactive or false.

4.2.1.1. 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 bytes. When the instruction specifies byte operation, the processor uses A0 to determine which byte to read and then issues data strobe.

A word read cycle flowchart is given in figure 4.3. A byte read cycle flowchart is given in figure 4.4. Read cycle timing is given in figure 4.5. Figure 4.6 details words and byte read cycle operations.

4.2.1.2. 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 A0 to determine which byte to write and then issues the data strobe. A word write cycle flowchart is given in figure 4.7. A byte write cycle flowchart is given in figure 4.8. Write cycle timing is given in figure 4.5. Figure 4.9 details word and byte write cycle operation.

4.2.1.3. Read-Modify-Write Cycle.

The read-modify-write cycle performs a byte read, modifies the data in the arithmetic-logic unit, and writes the data back to the same address. In the TS68008, 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 cycle flowchart is given in figure 4.10 and a timing diagram is given in figure 4.11.

4.2.2. BUS ARBITRATION. Bus arbitration on the 52-pin version of the TS68008 is identical to that on the TS68000.

Bus arbitration on the 48-pin version of the TS68008 has been modified from that on the TS68000. It is controlled by the same finite state machine as on the TS68000, but because the $\overline{\text{BGACK}}$ input signal is not bonded out to a pin and is, instead, permanently negated internally, the bus arbitration becomes a two-wire handshake circuit. Therefore, in reading the following paragraphs for a description of bus arbitration on the 48-pin version of the TS68008, the $\overline{\text{BGACK}}$ signal should be considered permanently negated.

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. on the 52-pin version of the TS68008 only, acknowledging that mastership has been assumed.

Figure 4.3 : Word Read Cycle Flowchart.

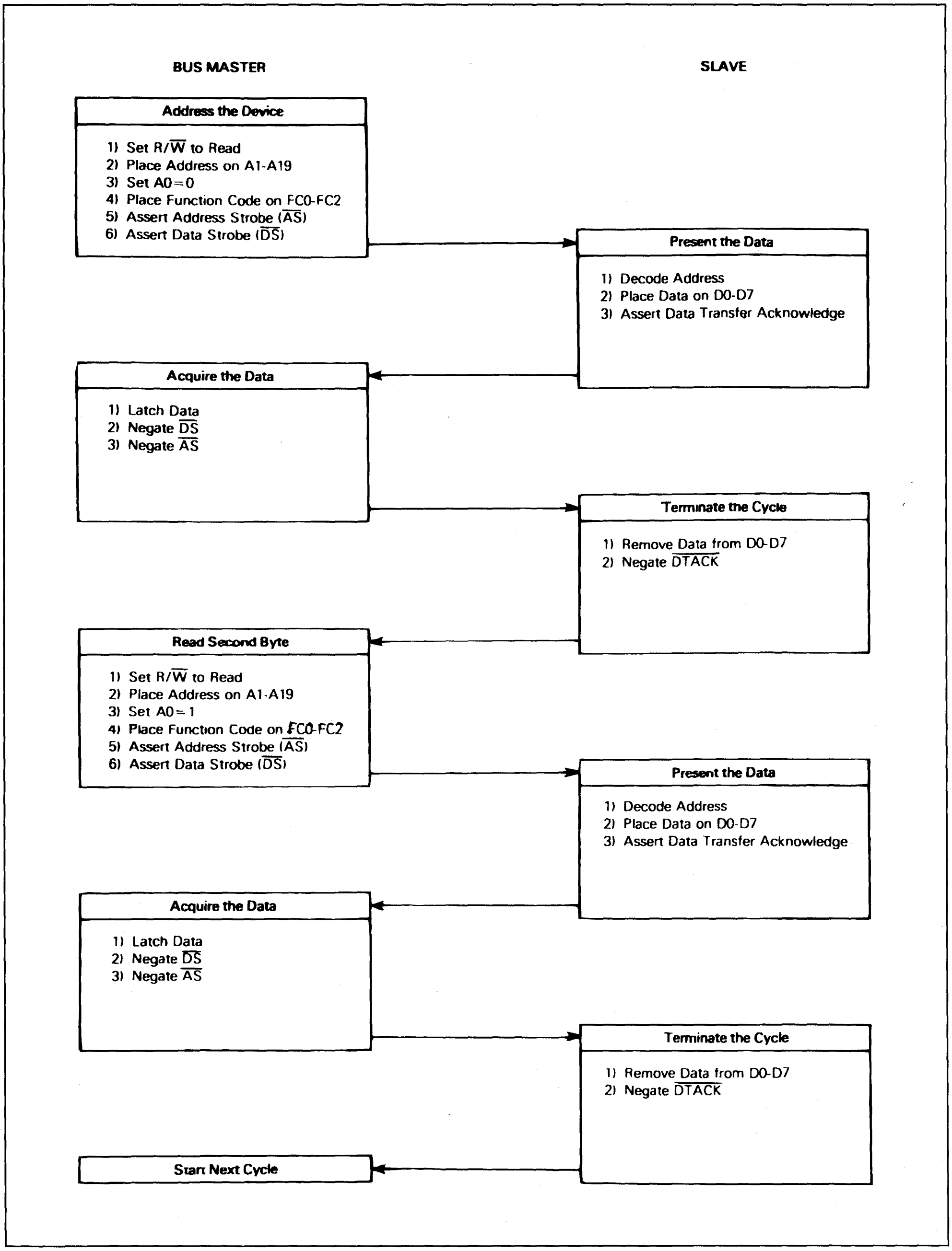

BUS MASTER
SLAVE
Address the Device
1) Set R/$\overline{W}$ to Read
2) Place Address on A1-A19
3) Set A0=0
4) Place Function Code on FC0-FC2
5) Assert Address Strobe ($\overline{AS}$)
6) Assert Data Strobe ($\overline{DS}$)
Present the Data
1) Decode Address
2) Place Data on D0-D7
3) Assert Data Transfer Acknowledge
Acquire the Data
1) Latch Data
2) Negate $\overline{DS}$
3) Negate $\overline{AS}$
Terminate the Cycle
1) Remove Data from D0-D7
2) Negate $\overline{DTACK}$
Read Second Byte
1) Set R/$\overline{W}$ to Read
2) Place Address on A1-A19
3) Set A0=1
4) Place Function Code on FC0-FC2
5) Assert Address Strobe ($\overline{AS}$)
6) Assert Data Strobe ($\overline{DS}$)
Present the Data
1) Decode Address
2) Place Data on D0-D7
3) Assert Data Transfer Acknowledge
Acquire the Data
1) Latch Data
2) Negate $\overline{DS}$
3) Negate $\overline{AS}$
Terminate the Cycle
1) Remove Data from D0-D7
2) Negate $\overline{DTACK}$
Start Next Cycle

Figure 4.4 : Byte Read Cycle Flowchart.

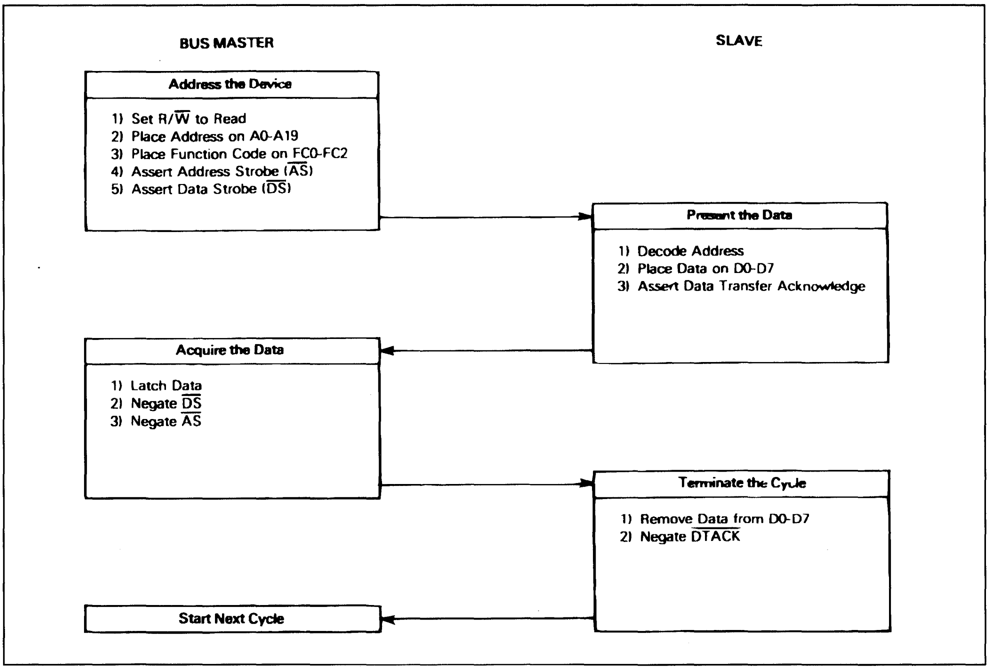

BUS MASTER
SLAVE
Address the Device
1) Set R/$\overline{W}$ to Read
2) Place Address on A0-A19
3) Place Function Code on FC0-FC2
4) Assert Address Strobe ($\overline{AS}$)
5) Assert Data Strobe ($\overline{DS}$)
Present the Data
1) Decode Address
2) Place Data on D0-D7
3) Assert Data Transfer Acknowledge
Acquire the Data
1) Latch Data
2) Negate $\overline{DS}$
3) Negate $\overline{AS}$
Terminate the Cycle
1) Remove Data from D0-D7
2) Negate $\overline{DTACK}$
Start Next Cycle

Figure 4.5 : Read and Write Cycle Timing Diagram.

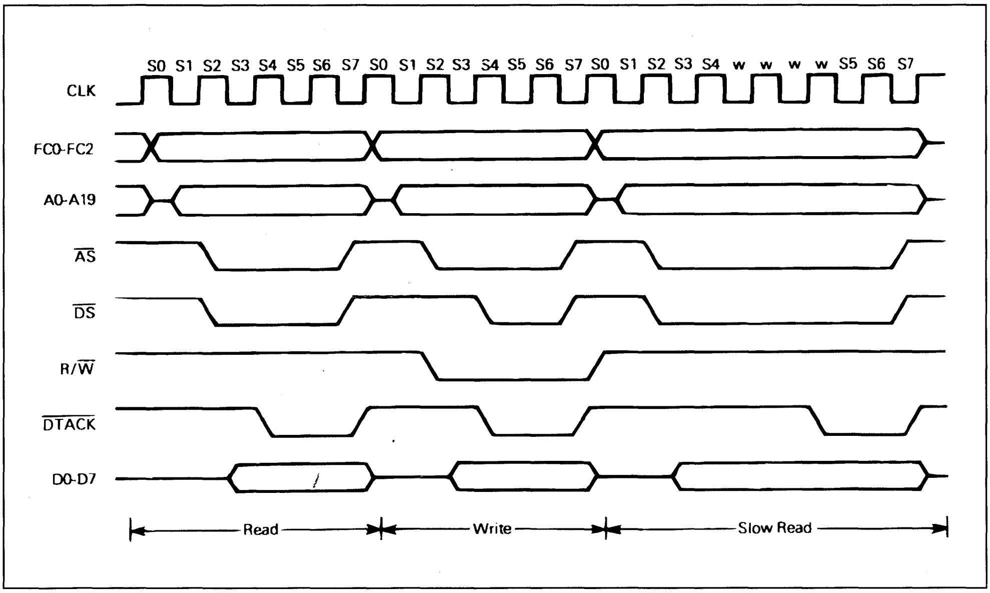


Figure 4.6 : Word and Byte Read Cycle Timing.

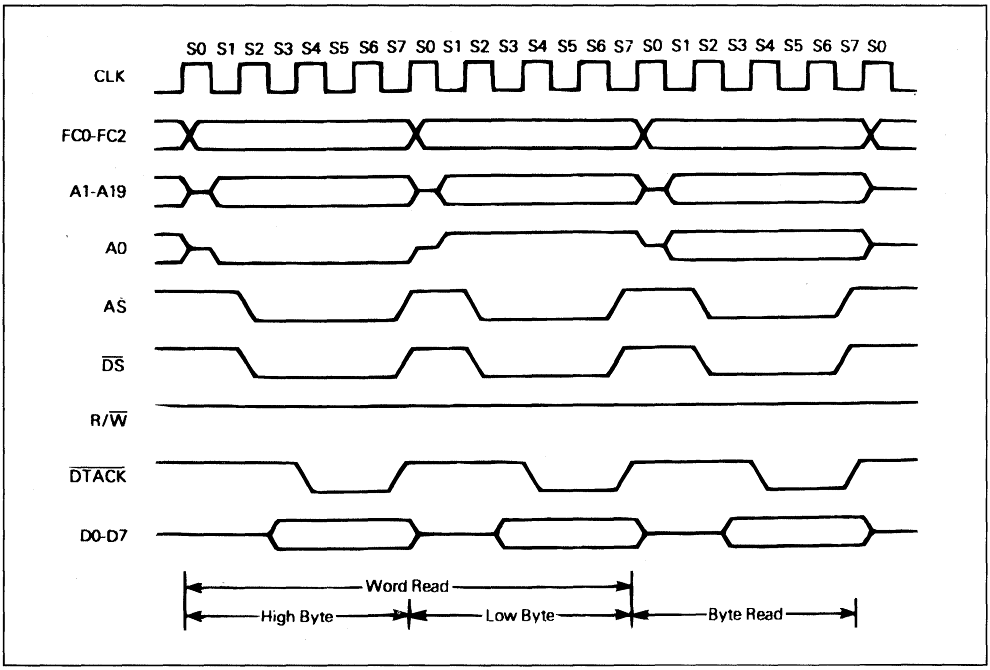

S0 S1 S2 S3 S4 S5 S6 S7 S0 S1 S2 S3 S4 S5 S6 S7 S0 S1 S2 S3 S4 S5 S6 S7 S0
CLK
FC0-FC2
A1-A19
A0
AS
DS
R/W
DTACK
D0-D7
Word Read
High Byte
Low Byte
Byte Read

Figure 4.7 : Word Write Cycle Flowchart.

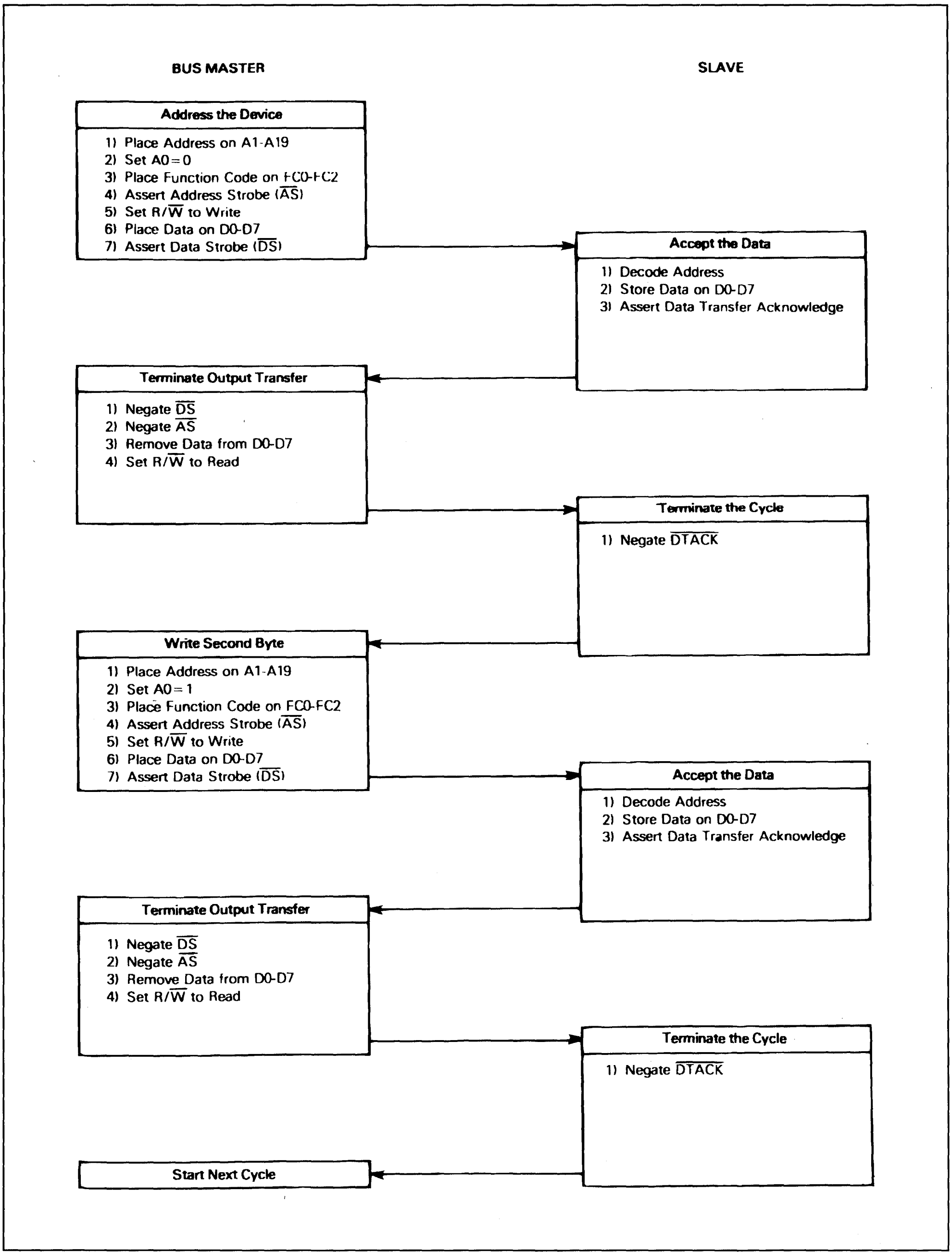

BUS MASTER
SLAVE
Address the Device
1) Place Address on A1-A19
2) Set A0=0
3) Place Function Code on FC0-FC2
4) Assert Address Strobe ($\overline{AS}$)
5) Set R/$\overline{W}$ to Write
6) Place Data on D0-D7
7) Assert Data Strobe ($\overline{DS}$)
Accept the Data
1) Decode Address
2) Store Data on D0-D7
3) Assert Data Transfer Acknowledge
Terminate Output Transfer
1) Negate $\overline{DS}$
2) Negate $\overline{AS}$
3) Remove Data from D0-D7
4) Set R/$\overline{W}$ to Read
Terminate the Cycle
1) Negate $\overline{DTACK}$
Write Second Byte
1) Place Address on A1-A19
2) Set A0=1
3) Place Function Code on FC0-FC2
4) Assert Address Strobe ($\overline{AS}$)
5) Set R/$\overline{W}$ to Write
6) Place Data on D0-D7
7) Assert Data Strobe ($\overline{DS}$)
Accept the Data
1) Decode Address
2) Store Data on D0-D7
3) Assert Data Transfer Acknowledge
Terminate Output Transfer
1) Negate $\overline{DS}$
2) Negate $\overline{AS}$
3) Remove Data from D0-D7
4) Set R/$\overline{W}$ to Read
Terminate the Cycle
1) Negate $\overline{DTACK}$
Start Next Cycle

Figure 4.8 : Byte Write Cycle Flowchart.

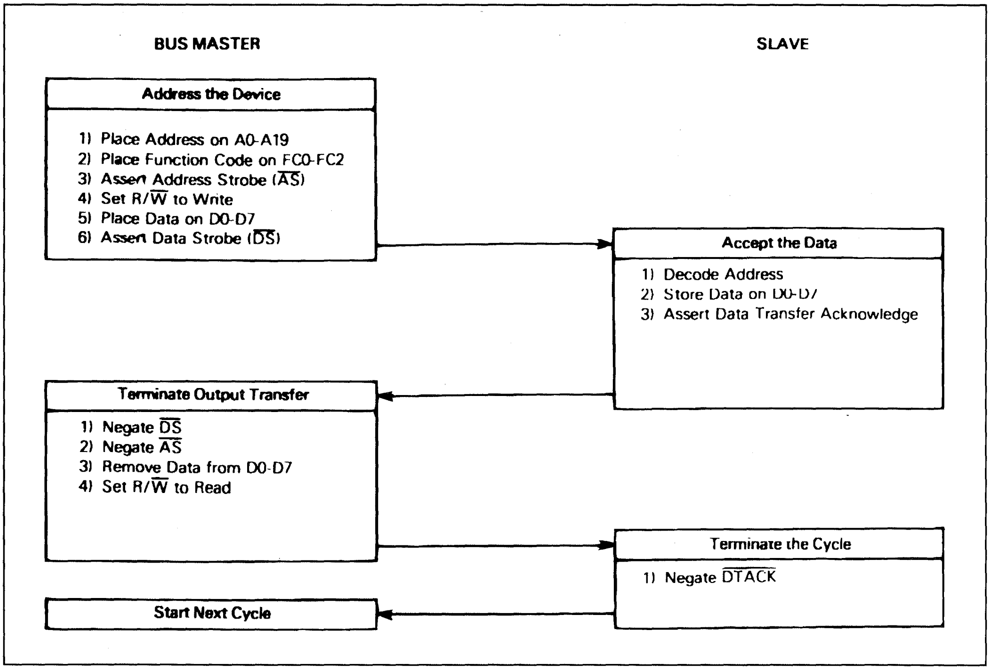

BUS MASTER
SLAVE
Address the Device
1) Place Address on A0-A19
2) Place Function Code on FC0-FC2
3) Assert Address Strobe ($\overline{AS}$)
4) Set R/$\overline{W}$ to Write
5) Place Data on D0-D7
6) Assert Data Strobe ($\overline{DS}$)
Accept the Data
1) Decode Address
2) Store Data on D0-D7
3) Assert Data Transfer Acknowledge
Terminate Output Transfer
1) Negate $\overline{DS}$
2) Negate $\overline{AS}$
3) Remove Data from D0-D7
4) Set R/$\overline{W}$ to Read
Terminate the Cycle
1) Negate $\overline{DTACK}$
Start Next Cycle

Figure 4.9 : Word and Byte Write Cycle Timing.

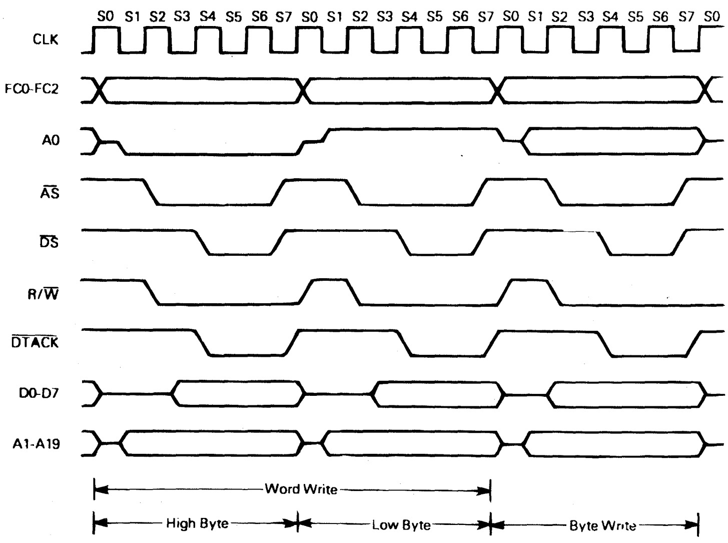


Figure 4.10 : Read-Modify-Write Cycle Flowchart.

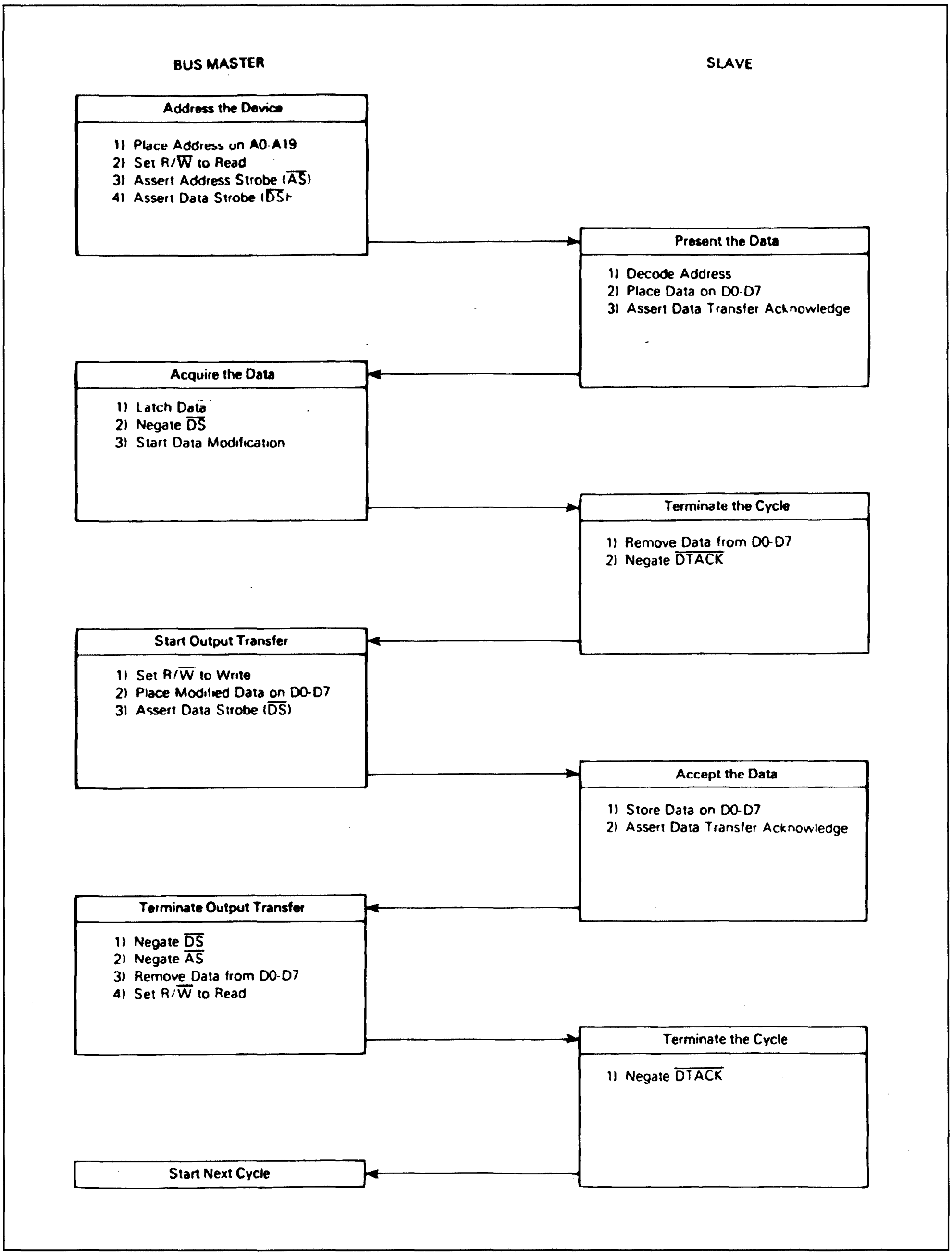
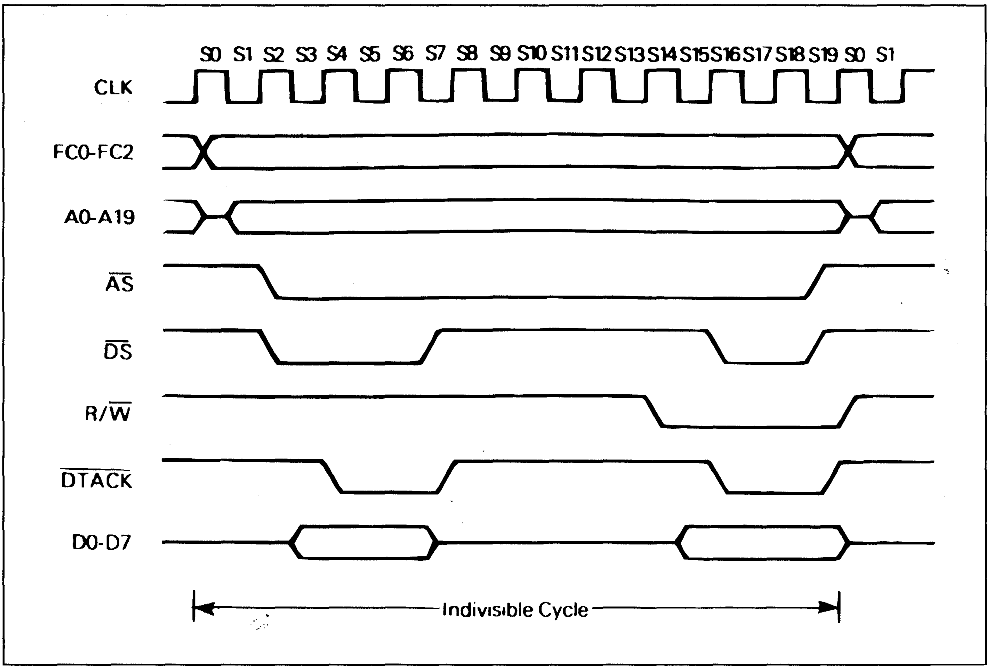

BUS MASTER
SLAVE
Address the Device
1) Place Address on A0-A19
2) Set R/$\overline{W}$ to Read
3) Assert Address Strobe ($\overline{AS}$)
4) Assert Data Strobe ($\overline{DS}$)
Present the Data
1) Decode Address
2) Place Data on D0-D7
3) Assert Data Transfer Acknowledge
Acquire the Data
1) Latch Data
2) Negate $\overline{DS}$
3) Start Data Modification
Terminate the Cycle
1) Remove Data from D0-D7
2) Negate $\overline{DTACK}$
Start Output Transfer
1) Set R/$\overline{W}$ to Write
2) Place Modified Data on D0-D7
3) Assert Data Strobe ($\overline{DS}$)
Accept the Data
1) Store Data on D0-D7
2) Assert Data Transfer Acknowledge
Terminate Output Transfer
1) Negate $\overline{DS}$
2) Negate $\overline{AS}$
3) Remove Data from D0-D7
4) Set R/$\overline{W}$ to Read
Terminate the Cycle
1) Negate $\overline{DTACK}$
Start Next Cycle

Figure 4.11 : Read-Modify-Write Cycle Timing.


S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19 S0 S1
CLK
FC0-FC2
A0-A19
AS
DS
R/W
DTACK
D0-D7
Indivisible Cycle

Flowcharts showing the detail involved in a request from a single device are illustrated in figure 4.12 for the 48-pin version and figure 4.13 for the 52-pin version. Timing diagrams for the same operation are given in figure 4.14 and figure 4.15. 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 (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.

4.2.2.1. Requesting The Bus.

External devices capable of becoming bus masters request the bus by asserting the bus request (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.

On the 52-pin version, 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.

Figure 4.12 : Bus Arbitration Cycle Flowchart for the 48-Pin Version.

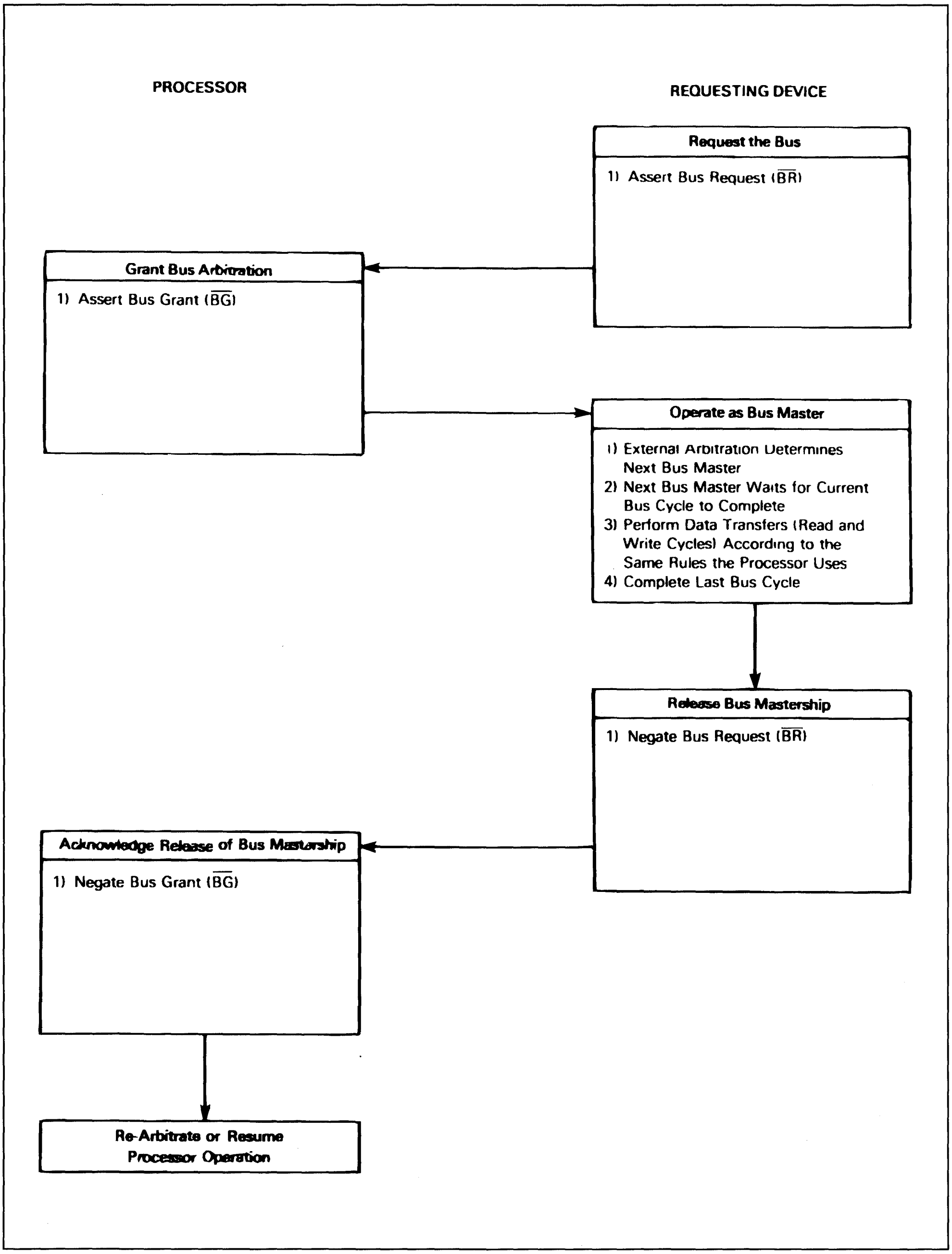
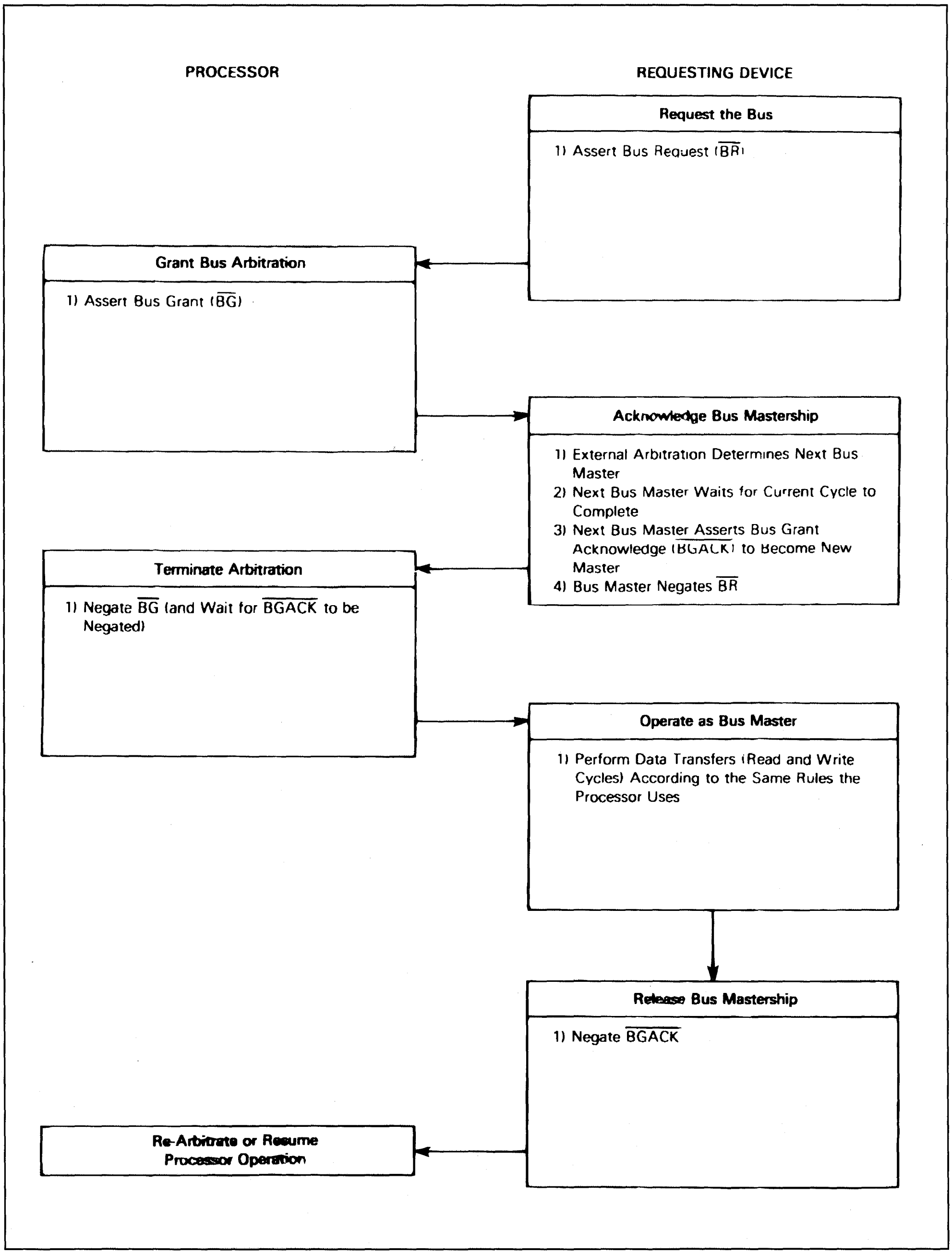

PROCESSOR
REQUESTING DEVICE
Request the Bus
1) Assert Bus Request ($\overline{BR}$)
Grant Bus Arbitration
1) Assert Bus Grant ($\overline{BG}$)
Operate as Bus Master
1) External Arbitration Determines Next Bus Master
2) Next Bus Master Waits for Current Bus Cycle to Complete
3) Perform Data Transfers (Read and Write Cycles) According to the Same Rules the Processor Uses
4) Complete Last Bus Cycle
Release Bus Mastership
1) Negate Bus Request ($\overline{BR}$)
Acknowledge Release of Bus Mastership
1) Negate Bus Grant ($\overline{BG}$)
Re-Arbitrate or Resume Processor Operation

Figure 4.13 : Bus Arbitration Cycle Flowchart for the 52-Pin Version.


PROCESSOR
REQUESTING DEVICE
Request the Bus
1) Assert Bus Request ($\overline{BR}$)
Grant Bus Arbitration
1) Assert Bus Grant ($\overline{BG}$)
Acknowledge Bus Mastership
1) External Arbitration Determines Next Bus Master
2) Next Bus Master Waits for Current Cycle to Complete
3) Next Bus Master Asserts Bus Grant Acknowledge ($\overline{BGACK}$) to Become New Master
4) Bus Master Negates $\overline{BR}$
Terminate Arbitration
1) Negate $\overline{BG}$ (and Wait for $\overline{BGACK}$ to be Negated)
Operate as Bus Master
1) Perform Data Transfers (Read and Write Cycles) According to the Same Rules the Processor Uses
Release Bus Mastership
1) Negate $\overline{BGACK}$
Re-Arbitrate or Resume Processor Operation

SGS-THOMSON MICROELECTRONICS

Figure 4.14 : Bus Arbitration Timing for the 48-Pin Version.

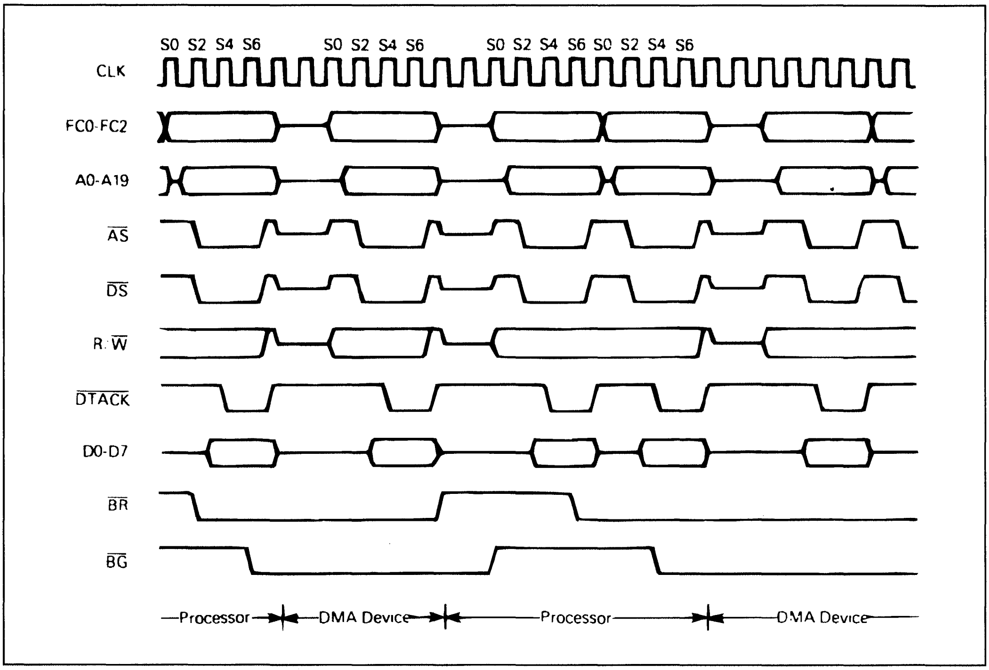
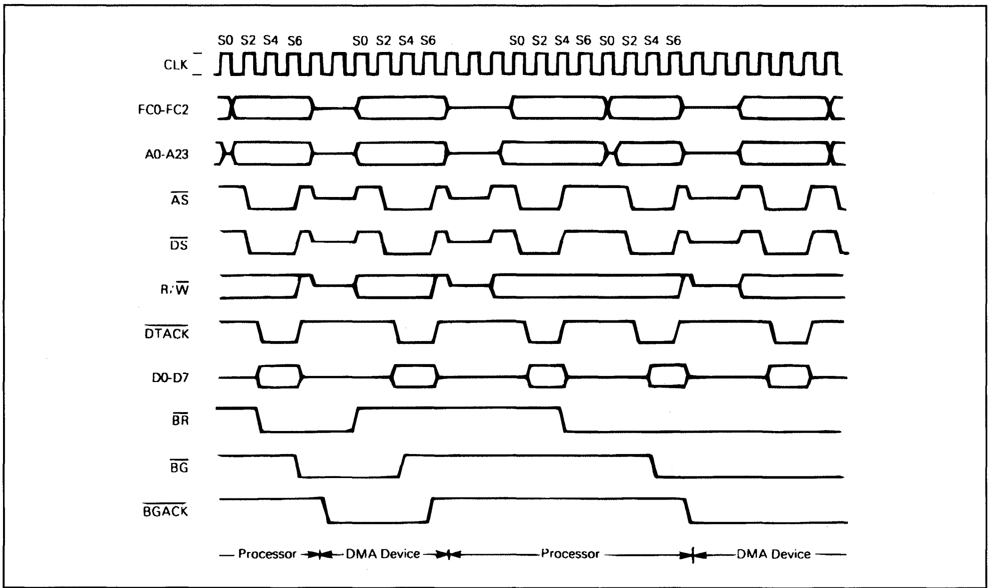


Figure 4.15 : Bus Arbitration Timing for the 52-Pin Version.



4.2.2.2. 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.

4.2.2.3. Acknowledgement Of Mastership (52-pin version of TS68008 only).

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.

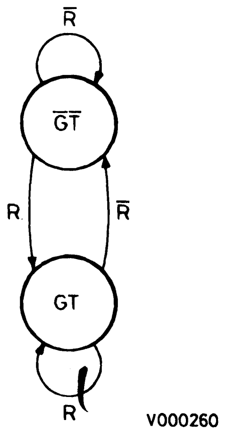
The bus request from the granted device should be dropped after bus grant acknowledge is asserted. If a bus request is still pending, another bus grant will be asserted within a few clocks of the negation of the bus grant. Refer to **4.2.3. Bus Arbitration Control Unit**. Note that the processor does not perform any external bus cycles before it re-asserts bus grant.

4.2.3. BUS ARBITRATION CONTROL. The bus arbitration control unit in the TS68008 is implemented with a finite state machine. A state diagram of this machine is shown in figure 4.16 for both pin versions of the TS68008. All asynchronous signals to the TS68008 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 setup time (#47) has been met (see figure 4.17). The input signal is sampled on the falling edge of the clock and is valid internally after the next falling edge.

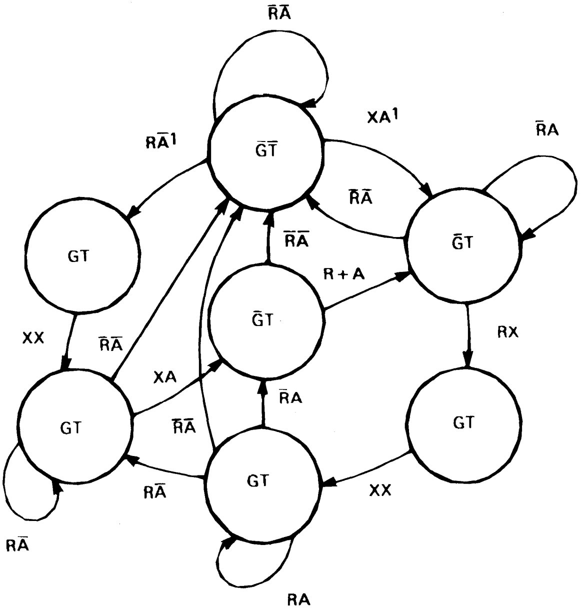
As shown in figure 4.16, input signals labeled R and A are internally synchronized on the bus request and bus grant acknowledge pins respectively. The bus grant output is labeled G and the internal three-state control signal T. If T is true, the address, data, and control buses are placed in a high-impedance 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.

Figure 4.16 : TS68008 Bus Arbitration Unit State Diagram.

(a) State Diagram for the 48-Pin Version of **TS68008**



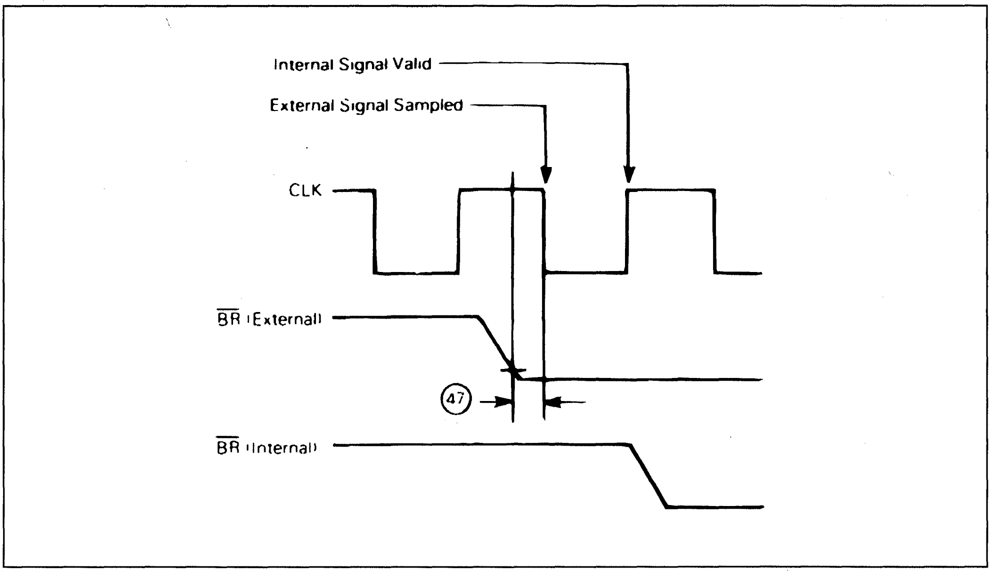
(b) State Diagram for the 52-Pin Version of **TS68008**



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

Notes : 1. State machine will not change if the bus is S0 or S1. Refer to **4.2.3 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 4.17 : Timing Relationship of External Asynchronous Inputs to Internal Signals.



A timing diagram of the bus arbitration sequence during a processor bus cycle is shown in figure 4.18. The bus arbitration sequence while the bus is inactive (i.e., executing internal operations such as a multiply instruction) is shown in figure 4.19.

Figure 4.18 : Bus Arbitration Timing Diagram-Processor Active.

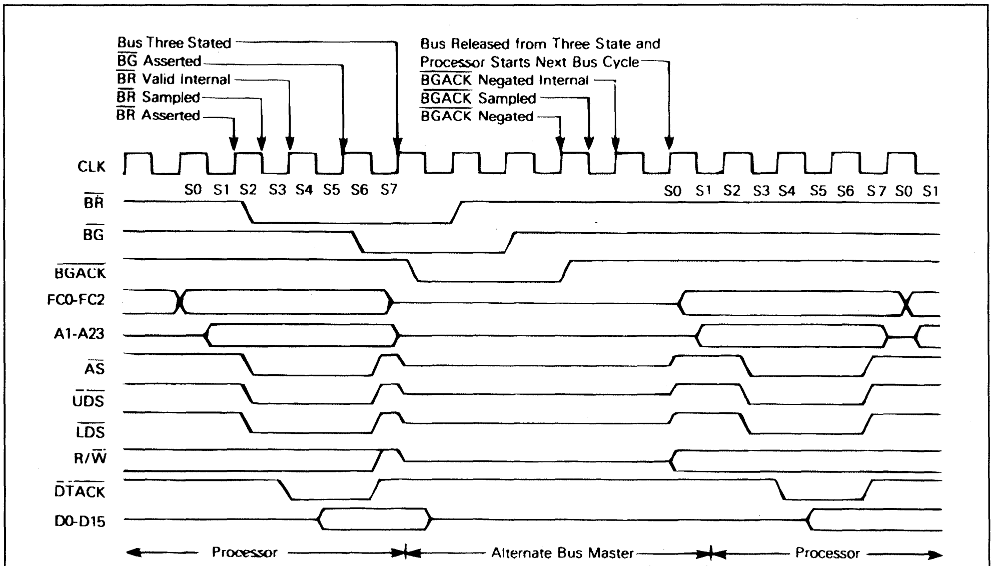
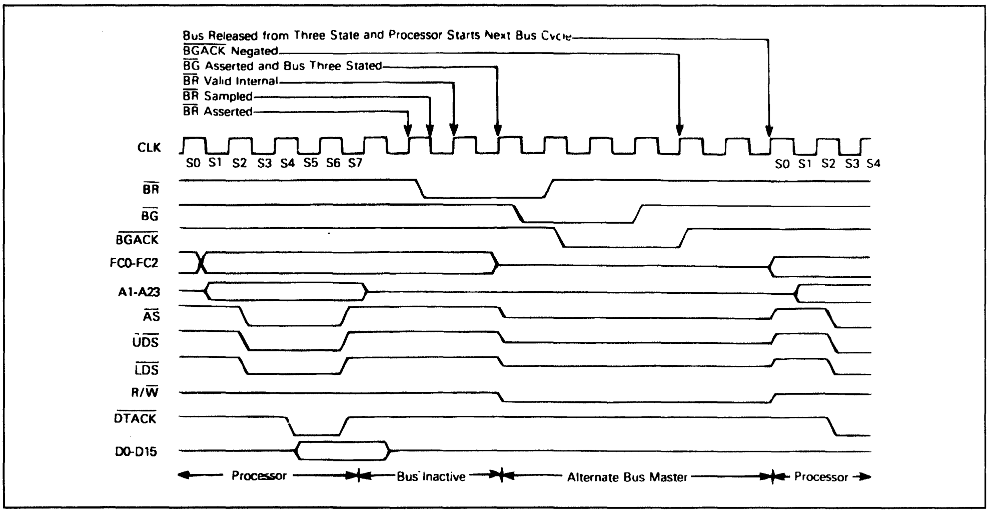


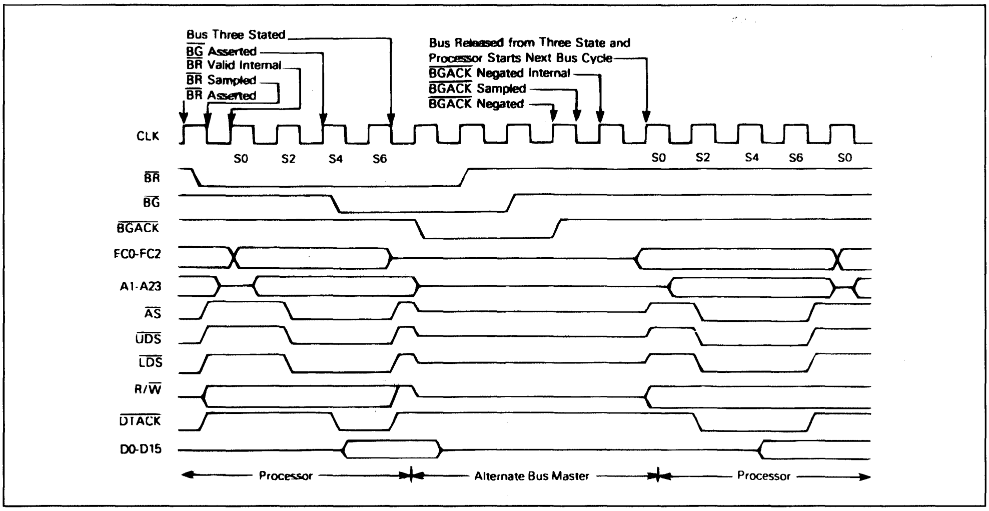
Figure 4.19 : Bus Arbitration Timing Diagram-Bus Inactive.



If a bus request is made at a time when the MPU has already begun as bus cycle but AS has not been asserted (bus state S0), BG will not be asserted on the next rising edge. Instead, BG will be delayed until the second rising edge following its internal assertion. This sequence is shown in figure 4.20.

4.2.4. 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 4.20 : Bus Arbitration Timing Diagram-Special Case.



4.2.4.1. Exception Sequence.

When the bus error signal is asserted, the current bus cycle is terminated. AS will be negated 2.5 clock periods after BERR is recognized. See 4.4 ASYNCHRONOUS VERSUS SYNCHRONOUS OPERATION for more information. As long as BERR remains asserted, the data and address buses will be in the high-impedance state. When BERR is negated, the processor will begin stacking for exception processing. 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.
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 processor loads the new program counter from the bus error vector. A software bus error handler routine is then executed by the processor. Refer to **5.2 Exception Processing** for additional information.

4.2.4.2. Re-running the Bus Cycle.

When the processor receives a bus error signal during a bus cycle and the HALT pin is being driven by an external device, the processor enters the re-run sequence. Figure 4.21 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 AS. If BERR and HALT are asserted during a read-modify-write bus cycle, a bus error operation results.

4.2.4.3. Halt Operation With No Bus Error.

The halt input signal to the TS68008 performs a halt/run/single-step function in a similar fashion to the 6800 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). HALT operation timing is shown in figure 4.22.

Figure 4.21 : Re-Run Bus Cycle Timing Information.

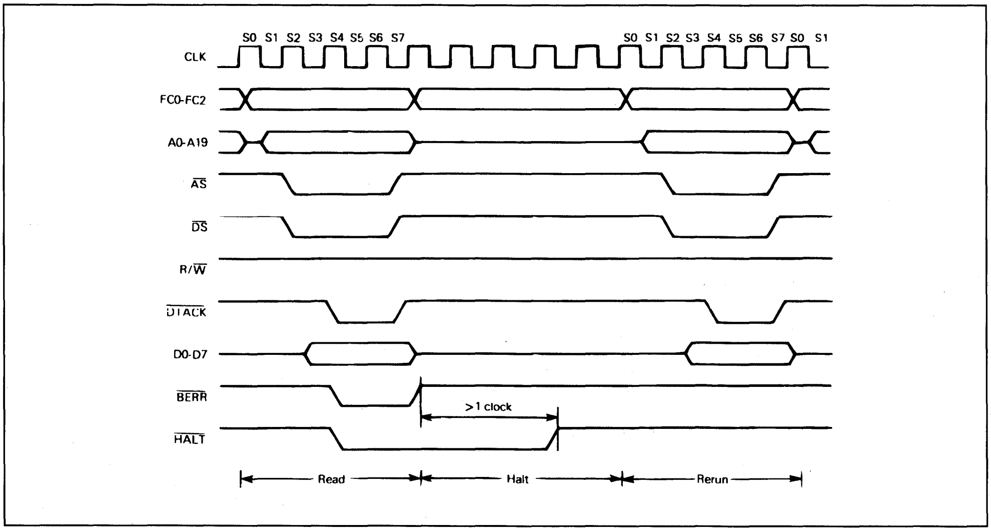
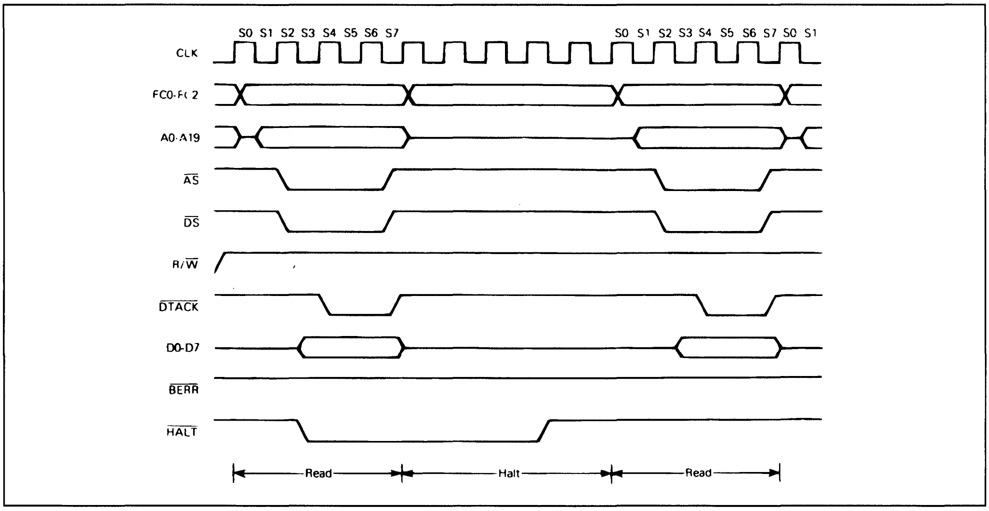


Figure 4.22 : $\overline{\text{HALT}}$ Operation Timing Diagram.



The 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 4.23 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 $\overline{\text{HALT}}$ 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 (see **4.2.4. Reset Operation**).

When the processor completes a bus cycle after recognizing that the halt signal is active, the address and data bus signals are put in the high-impedance state.

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 (i.e., three-states) 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.

4.2.4.4. 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. Figure 4.24 is a diagram of the bus error timing.

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.

4.2.5. RESET OPERATION. The reset signal is a bidirectional signal that allows either the processor or an external signal to reset the system. Figure 4.25 is a timing diagram for processor generated reset operation.

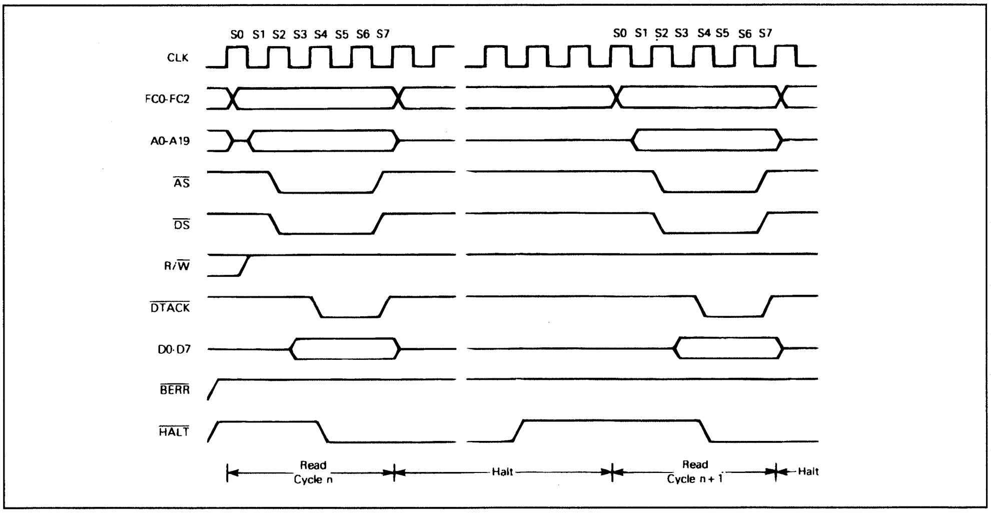


Figure 4.23 : $\overline{\text{HALT}}$ Signal Single-Step Operation Timing Characteristics.

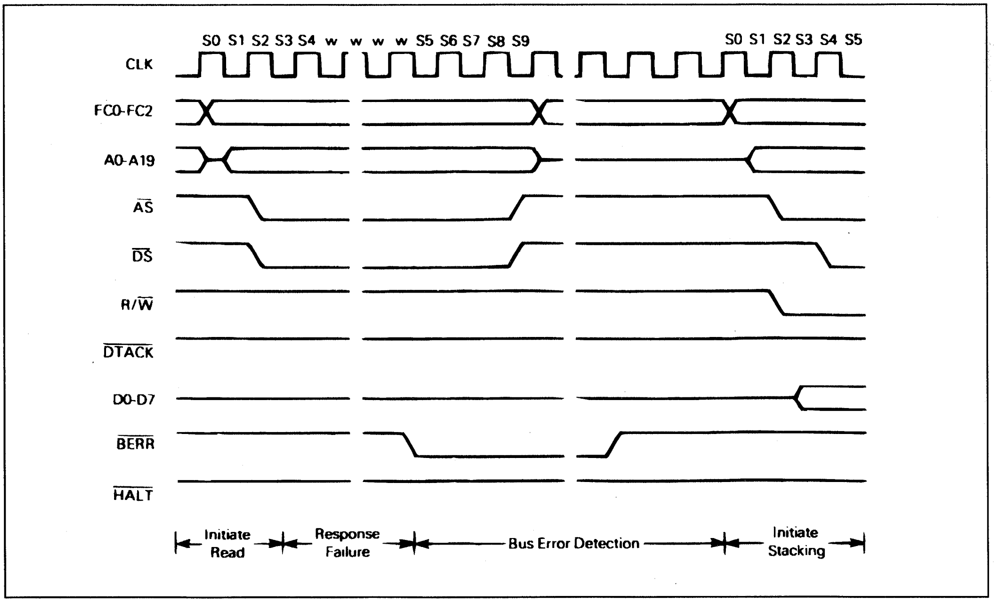
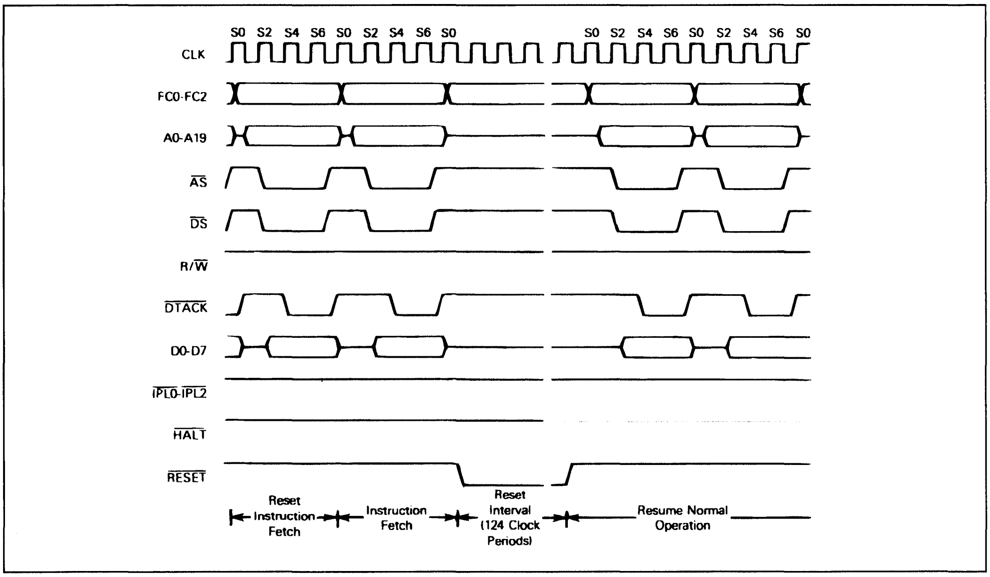


Figure 4.24 : Bus Error Timing Diagram.

Figure 4.25 : Reset Operation Timing Diagram.



When the reset and halt lines are driven it is recognized as an entire system reset, including the processor. For an external reset, both the HALT and RESET lines must be asserted to ensure total reset of the processor. Timing diagram for system reset is shown in figure 4.26. 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.

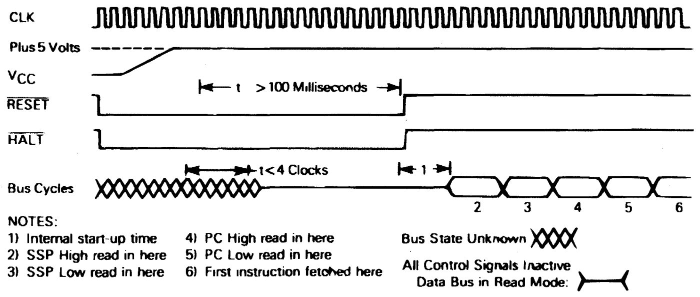
Asserting the reset and halt lines for 10 clock cycles will cause a processor reset, except when V_{CC} is initially applied to the processor. In this case, an external reset must be applied for at least 100 milliseconds allowing stabilization of the on-chip circuitry and system clock.

4.3. THE RELATIONSHIP OF DTACK, BERR, AND HALT

In order to properly control termination of a bus cycle for a re-run or a bus error condition, DTACK, BERR, and HALT should be asserted and negated on the rising edge of the TS68008 clock. This will assure that when two signals are asserted simultaneously, the required setup 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 TS68008. Parameter #48 is intended to ensure this operation in a totally asynchronous system, and may be ignored if the above conditions are met.

Figure 4.26 : System Reset Timing Diagram.



The preferred bus cycle terminations may be summarized as follows (case numbers refer to table 4.4) :

Normal Termination :	DTACK occurs first (case 1).
Halt Termination :	HALT is asserted at same time, or precedes DTACK (no BERR) cases 2 and 3.
Bus Error Termination :	BERR is asserted in lieu of, at same time, or preceding DTACK (case 4) ; BERR negated at same time, or after DTACK.
Re-Run Termination :	HALT and BERR asserted at the same time, or before DTACK (cases 5 and 6) ; HALT must be held at least one cycle after BERR.

Table 4.4 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 4.5 (DTACK is assumed to be negated normally in all cases ; for correct results, both DTACK and 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 DTACK and BERR simultaneously after time out (case 4).

EXAMPLE B :

A system uses error detection on RAM contents. Designer may (a) delay DTACK until data verified, and return BERR and HALT simultaneously to re-run error cycle (case 5), or if valid, return DTACK (b) delay DTACK until data verified, and return BERR at same time as DTACK if data in error (case 4) ; (c) return DTACK prior to data verification, as described in previous section. If data invalid, BERR is asserted (case 1) in next cycle. Error-handling software must know how to recover error cycle.

4.4. ASYNCHRONOUS VERSUS SYNCHRONOUS OPERATION

4.4.1. ASYNCHRONOUS OPERATION.

To achieve clock frequency independence at a system level, the TS68008 can be used in an asynchronous manner. This entails using only the bus handshake line (AS, DS, DTACK, BERR, HALT, and VPA) to control the data transfer. Using this method, 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 (DTACK) to terminate the bus cycle. If no slave responds or the access is invalid, external control logic asserts the BERR, or BERR and HALT signal to abort or rerun the bus cycle.

The 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 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.

Table 4.4 : $\overline{DTACK}$, $\overline{BERR}$ and $\overline{HALT}$ Assertion Results.

Case N°	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 re-run.
5	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	X X A	X S S	Terminate and re-run.
6	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	NA NA A	X A S	Terminate and re-run when $\overline{HALT}$ removed.

Legend : 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 4.5 : $\overline{BERR}$ and $\overline{HALT}$ Negation Results.

Conditions of Termination in Table 4.4	Control Signal	Negated an 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.

4.4.2. 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 setup time is given as parameter #47. If this setup 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 setup time it is not guaranteed not to be recognized. In addition, if 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 setup time given as parameter #27. Given this, parameter #31 may be ignored. Note that if $\overline{DTACK}$ is asserted, with the required setup 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, $\overline{VPA}$ 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 BERR is asserted in the absence of $\overline{DTACK}$, in which case it will terminate one clock cycle later in S9.

SECTION 5

PROCESSING STATES

This section describes the actions of the TS68008 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 is detailed.

The TS68008 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 memory 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.

5.1. PRIVILEGE STATES

The processor operates in one of two states of privilege : the "user" state or the "supervisor" state. The privilege state determines which operations are legal, is used by the external memory management device to control and translate accesses, and is used to choose between the supervisor stack pointer and the user stack pointer in instruction references.

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.

5.1.1. 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) or exception processing is invoked, 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.

5.1.2. USER STATE. The user state is the lower state of privilege and is controlled by the S bit of the status register. If the S bit is negated (low), the processor is executing instructions in the user state. 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.

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 to user stack pointer (MOVE USP) and move from user stack pointer (MOVE from USP) instructions are also privileged.

5.1.3. 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.

5.1.4. 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 states, such as interrupt acknowledge. Table 5.1 lists the classification of references.

Table 5.1 : Reference 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

5.2. 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.

5.2.1. 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 5.1), 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 vectored interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (figure 5.2) to the processor on data bus lines D0 through D7. The processor translates the vector number into a full 32-bit address, as shown in figure 5.3. The memory layout for exception vectors is given in table 5.2.

Figure 5.1 : Format of Vector Table Entries.

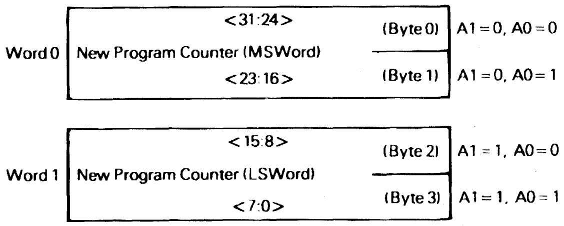
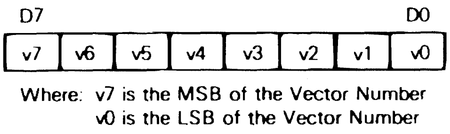


Figure 5.2 : Vector Number Format.



D7
D0
v7
v6
v5
v4
v3
v2
v1
v0
Where: v7 is the MSB of the Vector Number
v0 is the LSB of the Vector Number

Figure 5.3 : Vector Number Translated to an Address.

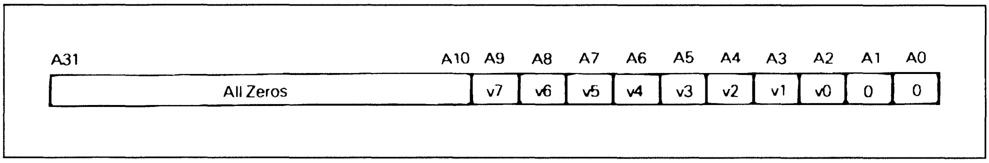


Table 5.2 : Vector Table.

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		–
64–225	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 SGS-THOMSON Microelectronics. No user peripheral devices should be assigned these numbers.

As shown in table 5.2, 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.

5.2.2. 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 register against 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.

5.2.3. 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 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. 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.

5.2.4. 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, address error, and bus 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. The trace and interrupt exceptions allow the current instruction to execute to completion, but preempt 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 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 5.3.

Table 5.3 : 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.

5.3. 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.

5.3.1. 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.

5.3.2. INTERRUPTS. Seven levels of interrupts are provided by the 68000 architecture. The TS68008 supports three interrupt levels : two, five, and seven, level seven being the highest. Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. 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 execution 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. First a copy of the status register is saved, and the privilege state is set to supervisor, 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 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 instruction execution commences in the interrupt handling routine. A flowchart for the interrupt acknowledge sequence is given in figure 5.4, a timing diagram is given in figure 5.5, and the interrupt processing sequence is shown in figure 5.6.

Figure 5.4 : Vector Acquisition Flowchart.

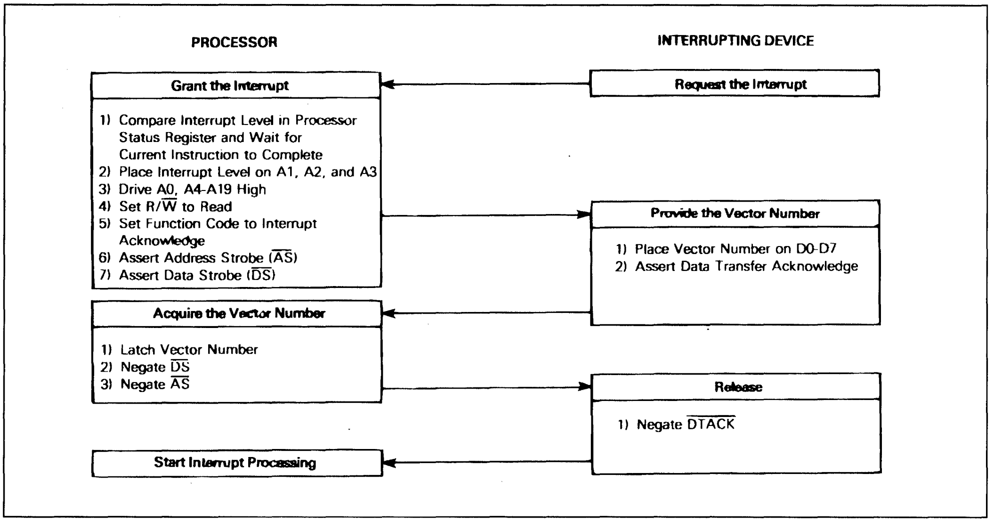
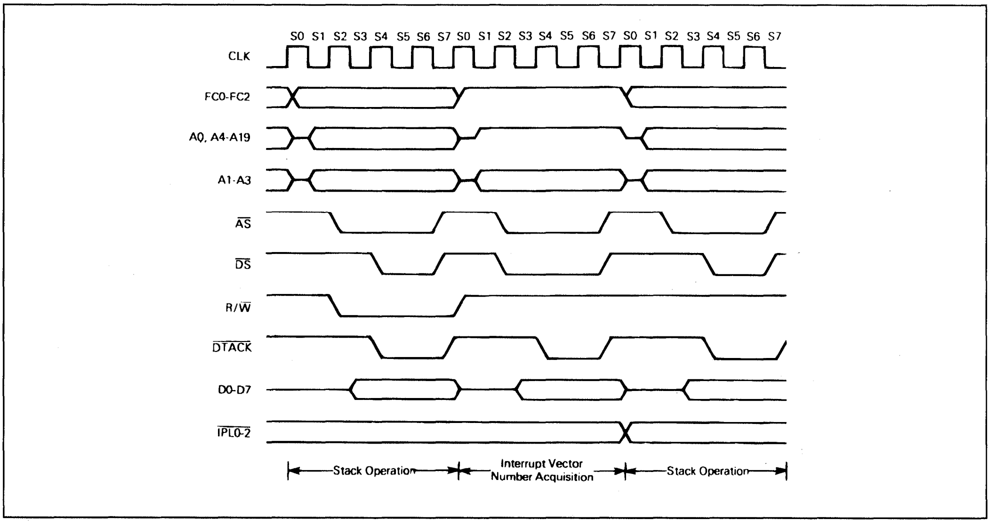

PROCESSOR
INTERRUPTING DEVICE
Grant the Interrupt
1) Compare Interrupt Level in Processor Status Register and Wait for Current Instruction to Complete
2) Place Interrupt Level on A1, A2, and A3
3) Drive A0, A4-A19 High
4) Set R/$\overline{W}$ to Read
5) Set Function Code to Interrupt Acknowledge
6) Assert Address Strobe ($\overline{AS}$)
7) Assert Data Strobe ($\overline{DS}$)
Request the Interrupt
Provide the Vector Number
1) Place Vector Number on D0-D7
2) Assert Data Transfer Acknowledge
Acquire the Vector Number
1) Latch Vector Number
2) Negate $\overline{DS}$
3) Negate $\overline{AS}$
Release
1) Negate $\overline{DTACK}$
Start Interrupt Processing

Figure 5.5 : Interrupt Acknowledge Cycle.



1. Acquire vector number via interrupt acknowledge.
2. Convert vector number to a full 32-bit address.
3. Stack the SR and PC by successive write cycles. Refer to figure 4-7 for word write cycle operation.
4. Place vector table address on A0-A19. Refer to figure 5-3 for address translation.
5. Read upper half of program counter (PC). Refer to figure 4-3 for word read cycle operation.
6. Increment vector table address by 2 and place it on A0-A19.
7. Read lower half of program counter (PC).
8. Load new program counter (PC).
9. Place contents of PC on A0-A19.
10. Read first instruction of service routine.

SGS-THOMSON MICROELECTRONICS

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.

5.3.3. UNITIALIZED INTERRUPT. An interrupting device asserts $\overline{\text{VPA}}$ or provides an interrupt vector during an interrupt acknowledge cycle to the TS68008. If the vector register of the peripheral has not been initialized, the responding 68000 Family peripheral will provide vector 15 ($0F), the unitialized interrupt vector. This provides a uniform way to recover from a programming error.

5.3.4. SPURIOUS INTERRUPT. If during the interrupt acknowledge cycle no device responds by asserting $\overline{\text{DTACK}}$ or $\overline{\text{VPA}}$, 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.

5.3.5. 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. 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 divide (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

5.3.6. 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. SGS-THOMSON Microelectronics 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 68000 Family compatible microprocessors. They are : $4AFA, $4AFB and $4AFC. Two of the patterns, $4AFA and $4AFB, are reserved for SGS-THOMSON system 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.

5.3.7. 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
RESET
RTE
MOVE to SR
AND Immediate to SR
EOR Immediate to SR
OR Immediate to SR
MOVE USP

5.3.8. TRACING. To aid in program development, the TS68008 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.

5.3.9. 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. Regardless of whether the processor was doing instruction or exception processing, that processing is terminated, and the processor immediately begins exception processing.

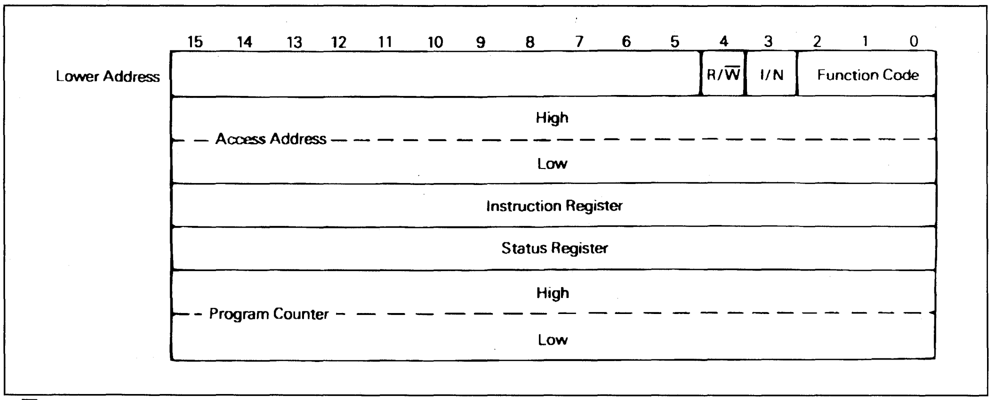
Exception processing for bus error follows the usual sequence of steps. The status register is copied, the supervisor state is entered, and the trace state is turned off. The vector number is generated to refer to the bus error vector. Since the processor was not between instructions when the bus error 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 (refer to figure 5.7). 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, two to ten bytes 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 the processor was processing an instruction or not, 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 5.7 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 the vector. 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 all memory contents. Only the $\overline{\text{RESET}}$ pin can restart a halted processor.

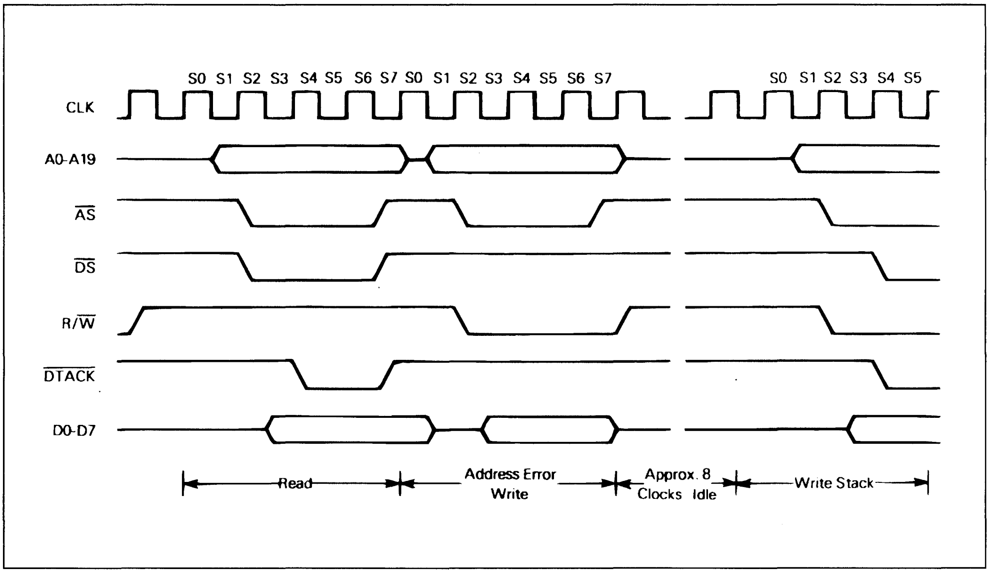
5.3.10. ADDRESS ERROR. Address error exceptions occur when the processor attempts to access a word or a long word operand or an instruction at an odd address. When the TS68008 detects an address error it prevents assertion of $\overline{\text{DS}}$ but asserts $\overline{\text{AS}}$ to provide proper bus arbitration support. The effect is much like an internally generated bus error, in that the bus cycle is aborted, and the processor ceases whatever processing it is currently doing and 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 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 5.8, an address error will execute a short bus cycle followed by exception processing.

Figure 5.7 : Supervisor Stack Order (Group 0).



R/$\overline{W}$ (read/write) : write = 0, read = 1. I/N (instruction/not) : instruction = 0, not = 1.

Figure 5.8 : Address Error Timing.



SECTION 6

INTERFACE WITH 6800 PERIPHERALS

SGS-THOMSON extensive line of 6800 peripherals are compatible with the TS68008. Some of these devices that are particularly useful are :

EF6821 Peripheral Interface Adapter

EF6840 Programmable Timer Module

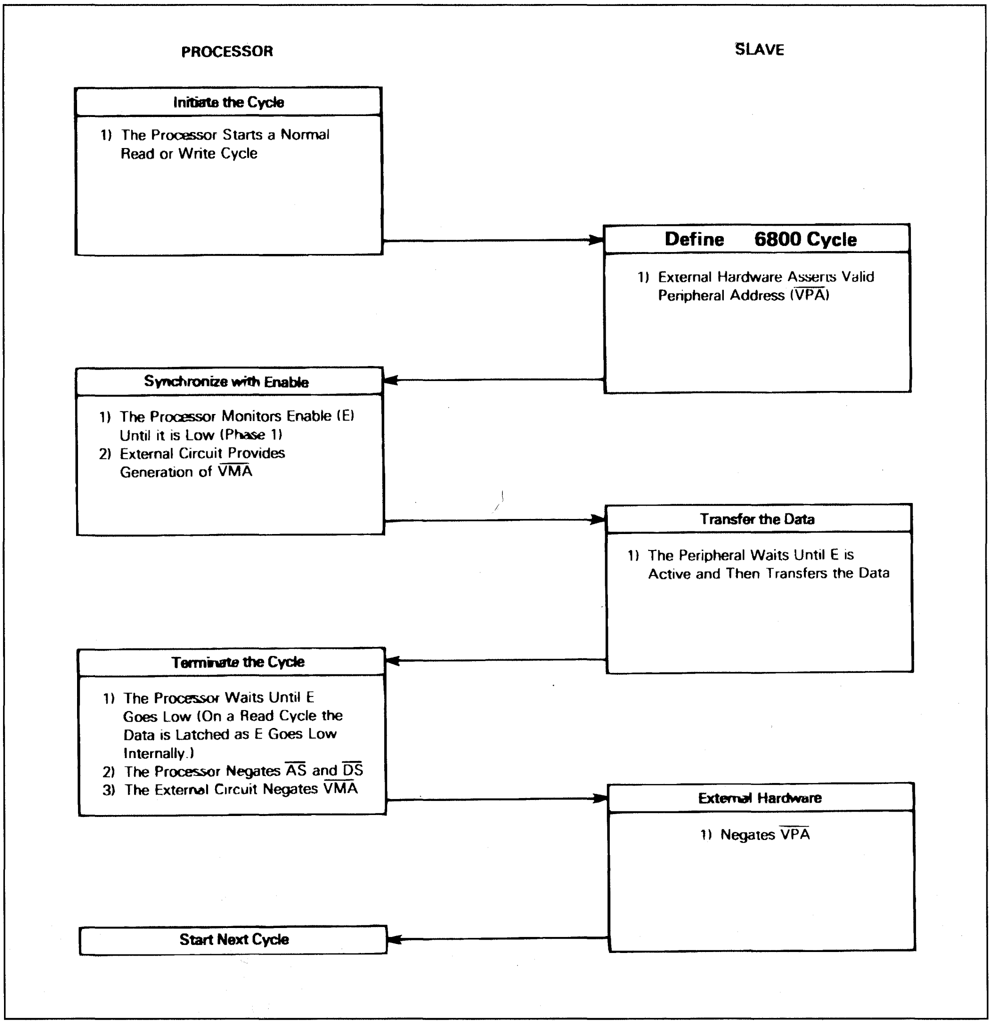
EF9345, EF9367 CRT Controllers

EF6850 Asynchronous Communications Interface Adapter

EF6854 Advanced Data Link Controller

To interface the synchronous 6800 peripherals with the asynchronous TS68008, the processor modifies its bus cycle to meet the 6800 cycle requirements whenever an 6800 device address is detected. This is possible since both processors use memory mapped I/O. Figure 6.1 is a flowchart of the interface operation between the processor and 6800 devices.

Figure 6.1 : 6800 Cycle Flowchart.



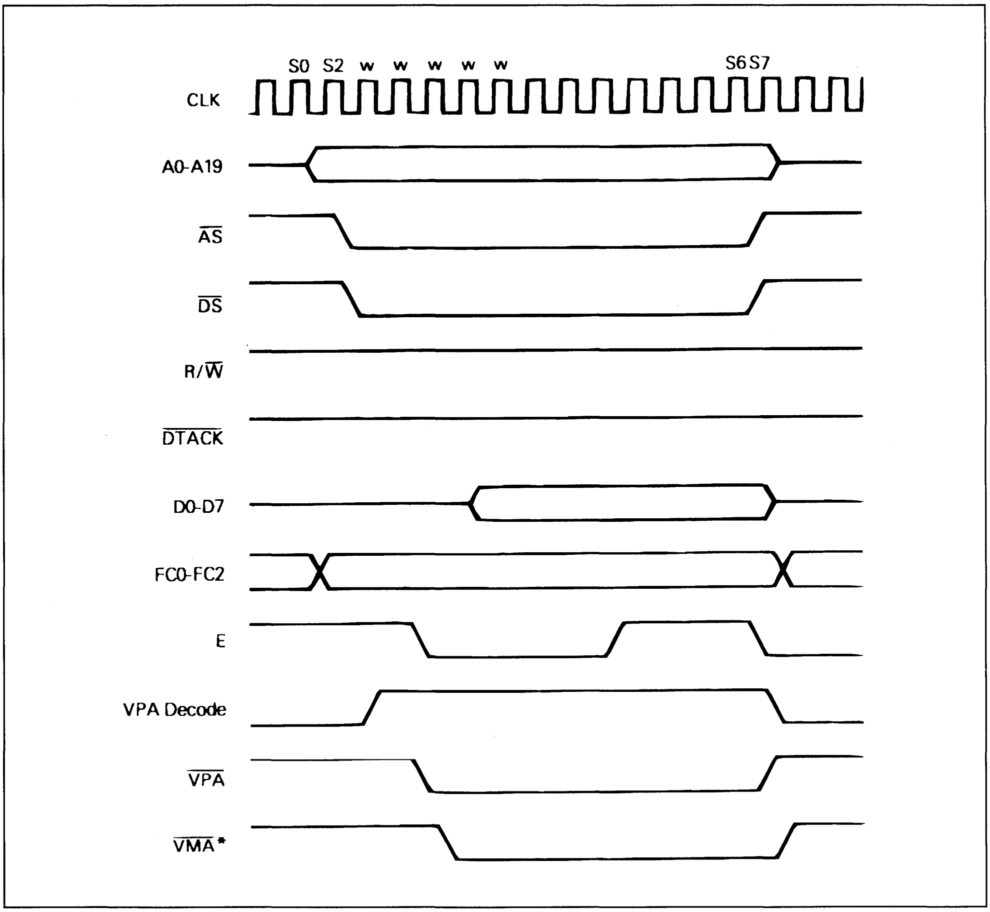
PROCESSOR
SLAVE
Initiate the Cycle
1) The Processor Starts a Normal Read or Write Cycle
Define 6800 Cycle
1) External Hardware Asserts Valid Peripheral Address ($\overline{VPA}$)
Synchronize with Enable
1) The Processor Monitors Enable (E) Until it is Low (Phase 1)
2) External Circuit Provides Generation of $\overline{VMA}$
Transfer the Data
1) The Peripheral Waits Until E is Active and Then Transfers the Data
Terminate the Cycle
1) The Processor Waits Until E Goes Low (On a Read Cycle the Data is Latched as E Goes Low Internally.)
2) The Processor Negates $\overline{AS}$ and $\overline{DS}$
3) The External Circuit Negates $\overline{VMA}$
External Hardware
1) Negates $\overline{VPA}$
Start Next Cycle

6.1. DATA TRANSFER OPERATION

Two signals on the processor provide the 6800 interface. They are : enable (E), and valid peripheral address (VPA). In addition, a valid memory address (VMA) signal must be provided (see 4.1.7. **6800 Peripheral Control**). Enable corresponds to the E signal in existing 6800 systems. The E clock frequency is one tenth of the incoming TS68008 clock frequency. The timing of E allows 1 megahertz peripherals to be used with an 8 megahertz TS68008. Enable has a 60/40 duty cycle ; that is, it is low for six input clocks and high for four input clocks.

6800 cycle timing is given in Section 8. At state zero 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 one, the address bus is released from the high-impedance state.

Figure 6.2 : 6800 Cycle Timing.



* Externally Generated.

During state two, 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 data strobe is also asserted in state two. If the bus cycle is a write cycle the read/write (R/$\overline{W}$) signal is switched to low (write) during state two. One half clock later, in state three, the write data is placed on the data bus, and in state four the data strobe is 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 an 6800 device (or an area reserved for 6800 devices) and that the bus should conform to the transfer characteristics of the 6800 bus. Valid peripheral address is derived by decoding the address bus, conditioned by address strobe. Chip select for the 6800 peripherals should be derived by decoding the address bus conditioned by $\overline{VMA}$ (not $\overline{AS}$).

After recognition of $\overline{VPA}$, the processor assures that the enable (E) is low, by waiting if necessary. Valid memory address (provided by an external circuit similar to that of figure 4.2) is then used as part of the chip select equation of the peripheral. This ensures that the 6800 peripherals are selected and deselected at the correct time. The peripheral now runs its cycle during the high portion of the E signal. Figure 6.2 depicts the 6800 cycle timing using the VMA generation circuit shown in figure 4.2. This cycle length is dependent strictly upon when $\overline{VPA}$ is asserted in relationship to the E clock.

During a read cycle, the processor latches the peripheral data in state six. For all cycles, the processor negates the address and data strobes one half clock cycle later in state seven, 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 address strobe is negated.

$\overline{DTACK}$ should not be asserted while $\overline{VPA}$ is asserted. Notice that $\overline{VMA}$ is active low, contrasted with the active high 6800 VMA. Refer to figure 4.2.

6.2. AC ELECTRICAL SPECIFICATIONS

The electrical specifications for interfacing the TS68008 to 6800 Family peripherals are located in Section 8.

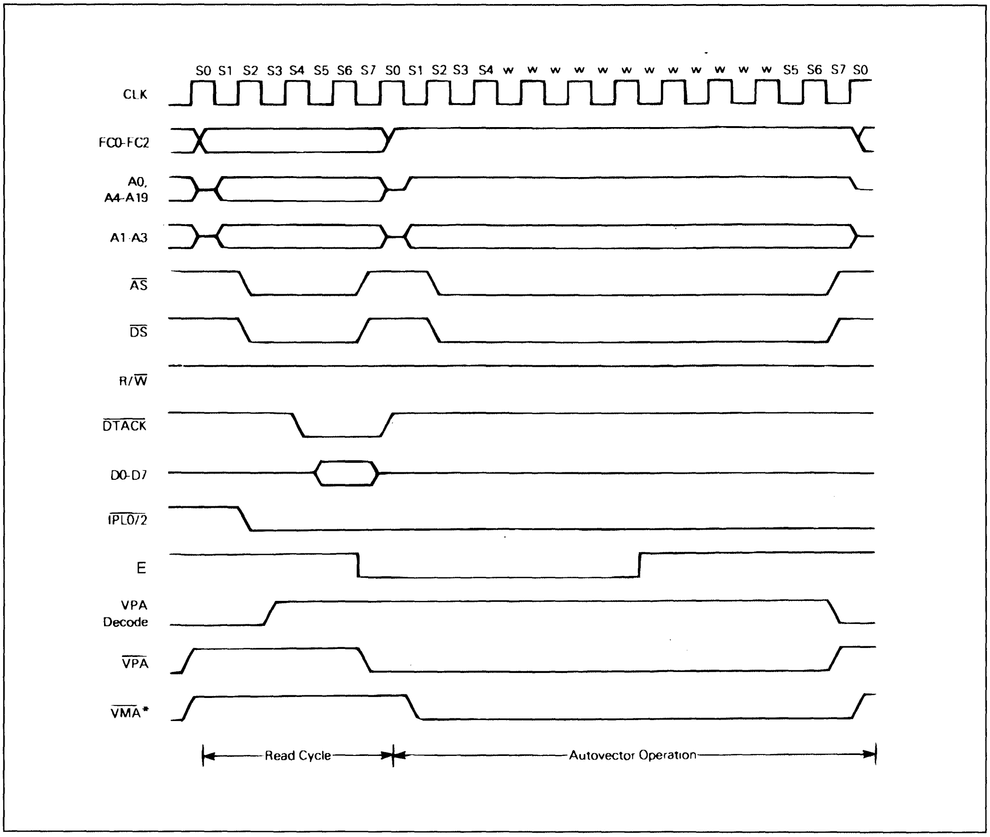
6.3. INTERRUPT INTERFACE OPERATION

During an interrupt acknowledge cycle while the processor is fetching the vector, the $\overline{VPA}$ is asserted, the TS68008 will complete a normal 6800 read cycle as shown in figure 6.3. 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 6800 interrupt sequence. The basic difference is that there are six normal interrupt vectors and one NMI type vector. As with both the 6800 and the TS68008'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 6800 peripheral address decoding should prevent unintended accesses.

Figure 6.3 : Autovector Operation Timing Diagram.



* Externally Generated.

SECTION 7

INSTRUCTION SET AND EXECUTION TIMES

7.1. INSTRUCTION SET

The following paragraphs provide information about the addressing categories and instruction set of the TS68008.

7.1.1. 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 by 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.

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 7.1 shows the various categories to which each of the effective address modes belong. Table 7.2 is the instruction set summary.

Table 7.1 : 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	–	–

Table 7.2 : Instruction Set.

Mnemonic	Description	Operation	Conditions 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) Λ (source) → Destination	–	*	*	0	0
ANDI	AND Immediate	(destination) Λ Immediate Data → Destination	–	*	*	0	0
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 + 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) → Destiantion	–	*	*	*	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
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 + Displacemment → 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 the Status Register	(source) → SR	*	*	*	*	*
MOVE from SR	Move from the Status Register	SR → Destination	–	–	–	–	–

SGS-THOMSON MICROELECTRONICS

Table 7.2 : Instruction Set (continued).

Mnemonic	Description	Operation	Conditions 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 → 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
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	*
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 and 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	Unlik	An → SP ; (SP) + → An	–	–	–	–	–

• logical exclusive OR	* affected
Λ logical AND	– unaffected
v logical OR	0 cleared
⊕ logical exclusive OR	1 set
~ logical complement	U undefined

7.1.2. INSTRUCTION PREFETCH. The TS68008 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 multiword 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 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. In the case of an interrupt or trace exception, neither word is used.
- 5. The program counter usually points to the last word fetched from the instruction stream.

7.2. 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. The number of periods includes instruction fetch and all applicable operand fetches and stores.

7.2.1. OPERAND EFFECTIVE ADDRESS CALCULATION TIMES. Table 7.3 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 parenthesis as (r/w). Note there are no write cycles involved in processing the effective address.

Table 7.3 : Effective Address Calculation Times.

Addressing Mode		Byte	Word	Long
	Register			
Dn	Data Register Direct	**0**(0/0)	**0**(0/0)	**0**(0/0)
An	Address Register Direct	**0**(0/0)	**0**(0/0)	**0**(0/0)
	Memory			
(An)	Address Register Indirect	**4**(1/0)	**8**(2/0)	**16**(4/0)
(An) +	Address Register Indirect with Postincrement	**4**(1/0)	**8**(2/0)	**16**(4/0)
– (An)	Address Register Indirect with Predecrement	**6**(1/0)	**10**(2/0)	**18**(4/0)
d(An)	Address Register Indirect with Displacement	**12**(3/0)	**16**(4/0)	**24**(6/0)
d(AN, ix)*	Address Register Indirect with Index	**14**(3/0)	**18**(4/0)	**26**(6/0)
xxx.W	Absolute Short	**12**(3/0)	**16**(4/0)	**24**(6/0)
xxx.L	Absolute Long	**20**(5/0)	**24**(6/0)	**32**(8/0)
d(PC)	Program Counter with Displacement	**12**(3/0)	**16**(4/0)	**24**(6/0)
d(PC, ix)	Program Counter with Index	**14**(3/0)	**18**(4/0)	**26**(6/0)
#xxx	Immediate	**8**(2/0)	**8**(2/0)	**16**(4/0)

* The size of the index register (ix) does not affect execution time.

Table 7.4 : Move Byte Instruction Execution Times.

Source	Destination								
	Dn	An	(An)	(An)+	–(An)	d(An)	d(An, x)*	xxx.W	xxx.L
Dn	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	20(4/1)	22(4/1)	20(4/1)	28(6/1)
An	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	20(4/1)	22(4/1)	20(4/1)	28(6/1)
(An)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	24(5/1)	26(5/1)	24(5/1)	32(7/1)
(An) +	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	24(5/1)	26(5/1)	24(5/1)	32(7/1)
– (An)	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	26(5/1)	28(5/1)	26(5/1)	34(7/1)
d(An)	20(5/0)	20(5/0)	24(5/1)	24(5/1)	24(5/1)	32(7/1)	34(7/1)	32(7/1)	40(9/1)
d(An, ix)*	22(5/0)	22(5/0)	26(5/1)	26(5/1)	26(5/1)	34(7/1)	36(7/1)	34(7/1)	42(9/1)
xxx.W	20(5/0)	20(5/0)	24(5/1)	24(5/1)	24(5/1)	32(7/1)	34(7/1)	32(7/1)	40(9/1)
xxx.L	28(7/0)	28(7/0)	32(7/1)	32(7/1)	32(7/1)	40(9/1)	42(9/1)	42(9/1)	48(11/11)
d(PC)	20(5/0)	20(5/0)	24(5/1)	24(5/1)	24(5/1)	32(7/1)	34(7/1)	32(7/1)	40(9/1)
d(PC, ix)*	22(5/0)	22(5/0)	26(5/1)	26(5/1)	26(5/1)	34(7/1)	36(7/1)	34(7/1)	42(9/1)
#xxx	16(4/0)	16(4/0)	20(4/1)	20(4/1)	20(4/1)	28(6/1)	30(6/1)	28(6/1)	36(8/1)

* The size of the index register (ix) does not affect execution time.

Table 7.5 : Move Word Instruction Execution Times.

Source	Destination								
	Dn	An	(An)	(An)+	– (An)	d(An)	d(An, ix)*	xxx.W	xxx.L
Dn	8(2/0)	8(2/0)	16(2/2)	16(2/2)	16(2/2)	24(4/2)	26(4/2)	20(4/2)	32(6/2)
An	8(2/0)	8(2/0)	16(2/2)	16(2/2)	16(2/2)	24(4/2)	26(4/2)	20(4/2)	32(6/2)
(An)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	32(6/2)	34(6/2)	32(6/2)	40(8/2)
(An) +	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	32(6/2)	34(6/2)	32(6/2)	40(8/2)
– (An)	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	34(6/2)	32(6/2)	34(6/2)	42(8/2)
d(An)	24(6/0)	24(6/0)	32(6/2)	32(6/2)	32(6/2)	40(8/2)	42(8/2)	40(8/2)	48(10/2)
d(An, ix)*	26(6/0)	26(6/0)	34(6/2)	34(6/2)	34(6/2)	42(8/2)	44(8/2)	42(8/2)	50(10/2)
xxx.W	24(6/0)	24(6/0)	32(6/2)	32(6/2)	32(6/2)	40(8/2)	42(8/2)	40(8/2)	48(10/2)
xxx.L	32(8/0)	32(8/0)	40(8/2)	40(8/2)	40(8/2)	48(10/2)	50(10/2)	48(10/2)	56(12/2)
d(PC)	24(6/0)	24(6/0)	32(6/2)	32(6/2)	32(6/2)	40(8/2)	42(8/2)	40(8/2)	48(10/2)
d(PC, ix)*	26(6/0)	26(6/0)	34(6/2)	34(6/2)	34(6/2)	42(8/2)	44(8/2)	42(8/2)	50(10/2)
#xxx	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	32(6/2)	34(6/2)	32(6/2)	40(8/2)

* The size of the index register (ix) does not affect execution time.

Table 7.6 : Move Long Instruction Execution Times.

Source	Destination								
	Dn	An	(An)	(An)+	– (An)	d(An)	d(An, ix)*	xxx.W	xxx.L
Dn	8(2/0)	8(2/0)	24(2/4)	24(2/4)	24(2/4)	32(4/4)	34(4/4)	32(4/4)	40(6/4)
An	8(2/0)	8(2/0)	24(2/4)	24(2/4)	24(2/4)	32(4/4)	34(4/4)	32(4/4)	40(6/4)
(An)	24(6/0)	24(6/0)	40(6/4)	40(6/4)	40(6/4)	48(8/4)	50(8/4)	48(8/4)	56(10/4)
(An) +	24(6/0)	24(6/0)	40(6/4)	40(6/4)	40(6/4)	48(8/4)	50(8/4)	48(8/4)	56(10/4)
– (An)	26(6/0)	26(6/0)	42(6/4)	42(6/4)	42(6/4)	50(8/4)	52(8/4)	50(8/4)	58(10/4)
d(An)	32(8/0)	32(8/0)	48(8/4)	48(8/4)	48(8/4)	56(10/4)	58(10/4)	56(10/4)	64(12/4)
d(An, ix)*	34(8/0)	34(8/0)	50(8/4)	50(8/4)	50(8/4)	58(10/4)	60(10/4)	58(10/4)	66(12/4)
xxx.W	32(8/0)	32(8/0)	48(8/4)	48(8/4)	48(8/4)	56(10/4)	58(10/4)	56(10/4)	64(12/4)
xxx.L	40(10/0)	40(10/0)	56(10/4)	56(10/4)	56(10/4)	64(12/4)	66(12/4)	64(12/4)	72(14/4)
d(PC)	32(8/0)	32(8/0)	48(8/4)	48(8/4)	48(8/4)	56(10/4)	58(10/4)	56(10/4)	64(12/4)
d(PC, ix)*	34(8/0)	34(8/0)	50(8/4)	50(8/4)	50(8/4)	58(10/4)	60(10/4)	58(10/4)	66(12/4)
#xxx	24(6/0)	24(6/0)	40(6/4)	40(6/4)	40(6/4)	48(8/4)	50(8/4)	48(8/4)	56(10/4)

* The size of the index register (ix) does not affect execution time.

7.2.2. MOVE INSTRUCTION EXECUTION TIMES. Tables 7.4, 7.5, and 7.6 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).

7.2.3. STANDARD INSTRUCTION EXECUTION TIMES. The number of clock periods shown in table 7.7 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 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 7.7 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.

7.2.4. IMMEDIATE INSTRUCTION EXECUTION TIMES. The number of clock periods shown in table 7.8 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 7.8, the headings have the following meanings : # = immediate operand, Dn = data register operand, An = address register operand, and M = memory operand.

7.2.5. SINGLE OPERAND INSTRUCTION EXECUTION TIMES. Table 7.9 indicates 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.

7.2.6. SHIFT/ROTATE INSTRUCTION EXECUTION TIMES. Table 7.10 indicates 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.

7.2.7. BIT MANIPULATION INSTRUCTION EXECUTION TIMES. Table 7.11 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 7.7 : Standard Instruction Execution Times.

Instruction	Size	op <ea>, An	op <ea>, Dn	op Dn, <M>
ADD	Byte Word Long	– **12**(2/0) + **10**(2/0) +**	**8**(2/0) + **8**(2/0) + **10**(2/0) +**	**12**(2/1) + **16**(2/2) + **24**(2/4) +
AND	Byte Word Long	– – –	**8**(2/0) + **8**(2/0) + **10**(2/0) +**	**12**(2/1) + **16**(2/2) + **24**(2/4) +
CMP	Byte Word Long	– **10**(2/0) + **10**(2/0) +	**8**(2/0) + **8**(2/0) + **10**(2/0) +	– – –
DIVS DIVU		– –	**162**(2/0) +* **144**(2/0) +*	– –
EOR	Byte Word Long	– – –	**8**(2/0) +*** **8**(2/0) +*** **12**(2/0) +***	**12**(2/1) + **16**(2/2) + **24**(2/4) +
MULS MULU		– –	**74**(2/0) +* **74**(2/0) +*	– –
OR	Byte Word Long	– – –	**8**(2/0) + **8**(2/0) + **10**(2/0) +**	**12**(2/1) + **16**(2/2) + **24**(2/4) +
SUB	Byte Word Long	– **12**(2/0) + **10**(2/0) +**	**8**(2/0) + **8**(2/0) + **10**(2/0) +**	**12**(2/1) + **16**(2/2) + **24**(2/4) +

Notes :
+ Add effective address calculation time
* Indicates maximum value
** The base time of 10 clock periods is increased to 12 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 TS68008 provides less than 10% difference between the best and worst case timings.
MULS, MULU - The multiply algorithm requires 42 + 2n clocks where n is defined as :
MULS : n = tag the <ea> with a zero as the MSB ; 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.
MULU : n = the number of ones in the <ea>

Table 7.8 : Immediate Instruction Clock Periods.

Instruction	Size	op#, Dn	op#, An	op#, M
ADDI	Byte Word Long	**16**(4/0) **16**(4/0) **28**(6/0)	– – –	**20**(4/1) + **24**(4/2) + **40**(6/4) +
ADDQ	Byte Word Long	**8**(2/0) **8**(2/0) **12**(2/0)	– **12**(2/0) **12**(2/0)	**12**(2/1) + **16**(2/2) + **24**(2/4) +
ANDI	Byte Word Long	**16**(4/0) **16**(4/0) **28**(6/0)	– – –	**20**(4/1) + **24**(4/2) + **40**(6/4) +
CMPI	Byte Word Long	**16**(4/0) **16**(4/0) **26**(6/0)	– – –	**16**(4/0) + **16**(4/0) + **24**(6/0) +
EORI	Byte Word Long	**16**(4/0) **16**(4/0) **28**(6/0)	– – –	**20**(4/1) + **24**(4/2) + **40**(6/4) +
MOVEQ	Long	**8**(2/0)	–	–
ORI	Byte Word Long	**16**(4/0) **16**(4/0) **28**(6/0)	– – –	**20**(4/1) + **24**(4/2) + **40**(6/4) +
SUBI	Byte Word Long	**16**(4/0) **16**(4/0) **28**(6/0)	– – –	**12**(2/1) + **16**(2/2) + **24**(2/4) +
SUBQ	Byte Word Long	**8**(2/0) **8**(2/0) **12**(2/0)	– **12**(2/0) **12**(2/0)	**20**(4/1) + **24**(4/2) + **40**(6/4) +

+ add effective address calculation time

Table 7.9 : Single Operand Instruction Execution Times.

Instruction	Size	Register	Memory
CLR	Byte Word Long	**8**(2/0) **8**(2/0) **10**(2/0)	**12**(2/1) + **16**(2/2) + **24**(2/4) +
NBCD	Byte	**10**(2/0)	**12**(2/1) +
NEG	Byte Word Long	**8**(2/0) **8**(2/0) **10**(2/0)	**12**(2/1) + **16**(2/2) + **24**(2/4) +
NEGX	Byte Word Long	**8**(2/0) **8**(2/0) **10**(2/0)	**12**(2/1) + **16**(2/2) + **24**(2/4) +
NOT	Byte Word Long	**8**(2/0) **8**(2/0) **10**(2/0)	**12**(2/1) + **16**(2/2) + **24**(2/4) +
Scc	Byte, False Byte, True	**8**(2/0) **10**(2/0)	**12**(2/1) + **12**(2/1) +
TAS	Byte	**8**(2/0)	**14**(2/1) +
TST	Byte Word Long	**8**(2/0) **8**(2/0) **8**(2/0)	**8**(2/0) + **8**(2/0) + **8**(2/0) +

+ add effective address calculation time

Table 7.10 : Shift/Rotate Instruction Clock Periods.

Instruction	Size	Register	Memory
ASR, ASL	Byte Word Long	10 + **2n**(2/0) 10 + **2n**(2/0) 12 + **2n**(2/0)	– **16**(2/2) + –
LSR, LSL	Byte Word Long	10 + **2n**(2/0) 10 + **2n**(2/0) 12 + **2n**(2/0)	– **16**(2/2) + –
ROR, ROL	Byte Word Long	10 + **2n**(2/0) 10 + **2n**(2/0) 12 + **2n**(2/0)	– **16**(2/2) + –
ROXR, ROXL	Byte Word Long	10 + **2n**(2/0) 10 + **2n**(2/0) 12 + **2n**(2/0)	– **16**(2/2) + –

+ add effective address calculation time
n is the shift count

Table 7.11 : Bit Manipulation Instruction Execution Times.

Instruction	Size	Dynamic		Static	
		Register	Memory	Register	Memory
BCHG	Byte Long	– **12**(2/0) *	**12**(2/1) + –	– **20**(4/0) *	**20**(4/1) + –
BCLR	Byte Long	– **14**(2/0) *	**12**(2/1) + –	– **22**(4/0) *	**20**(4/1) + –
BSET	Byte Long	– **12**(2/0) *	**12**(2/1) + –	– **20**(4/0)*	**20**(4/1) + –
BTST	Byte Long	– **10**(2/0)	**8**(2/0) + –	– **18**(4/0)	**16**(4/0) + –

+ add effective address calculation time
* Indicates maximum value

7.2.8. CONDITIONAL INSTRUCTION EXECUTION TIMES. Table 7.12 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.

Table 7.12 : Conditional Instruction Excecution Times.

Instruction	Displacement	Trap or Branch Taken	Trap or Branch Not Taken
Bcc	Byte Word	**18**(4/0) **18**(4/0)	**12**(2/0) **20**(4/0)
BRA	Byte Word	**18**(4/0) **18**(4/0)	– –
BSR	Byte Word	**34**(4/4) **34**(4/4)	– –
DBcc	CC True CC False	– **18**(4/0)	**20**(4/0) **26**(6/0)
CHK	–	**68**(8/6) +*	**14**(2/0) +
TRAP	–	**62**(8/6)	–
TRAPV	–	**66**(10/6)	**8**(2/0)

+ add effective address calculation time
* Indicates maximum value

7.2.9. JMP, JSR, LEA, PEA, AND MOVEM INSTRUCTION EXECUTION TIMES. Table 7.13 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).

7.2.10. MULTI-PRECISION INSTRUCTION EXECUTION TIMES. Table 7.14 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 7.14, the headings have the following meanings : Dn = data register operand and M = memory operand.

7.2.11. MISCELLANEOUS INSTRUCTION EXECUTION TIMES. Tables 7.15 and 7.16 indicate the number of clock periods for the following miscellaneous instructions. The number of bus read and write cycles 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.

Table 7.13 : JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times.

Instruct.	Size	(An)	(An) +	– (An)	d(An)	d(An, ix)*	xxx.W	xxx.L	d(PC)	d(PC, ix)*
JMP	–	**16**(4/0)	–	–	**18**(4/0)	**22**(4/0)	**18**(4/0)	**24**(6/0)	**18**(4/0)	**22**(4/0)
JSR	–	**32**(4/4)	–	–	**34**(4/4)	**38**(4/4)	**34**(4/4)	**40**(6/4)	**34**(4/4)	**38**(4/4)
LEA	–	**8**(2/0)	–	–	**16**(4/0)	**20**(4/0)	**16**(4/0)	**24**(6/0)	**16**(4/0)	**20**(4/0)
PEA	–	**24**(2/4)	–	–	**32**(4/4)	**36**(4/4)	**32**(4/4)	**40**(6/4)	**32**(4/4)	**36**(4/4)
MOVEM M → R	Word	**24 + 8n** (6 + 2n/0)	**24 + 8n** (6 + 2n/0)	– –	**32 + 8n** (8 + 2n/0)	**34 + 8n** (8 + 2n/0)	**32 + 8n** (10 + n/0)	**40 + 8n** (10 + 2n/0)	**32 + 8n** (8 + 2n/0)	**34 + 8n** (8 + 2n/0)
	Long	**24 + 16n** (6 + 4n/0)	**24 + 16n** (6 + 4n/0)	– –	**32 + 16n** (8 + 4n/0)	**34 + 16n** (8 + 4n/0)	**32 + 16n** (8 + 4n/0)	**40 + 16n** (8 + 4n/0)	**32 + 16n** (8 + 4n/0)	**34 + 16n** (8 + 4n/0)
MOVEM R → M	Word	**16 + 8n** (4/2n)	– –	**16 + 8n** (4/2n)	**24 + 8n** (6/2n)	**26 + 8n** (6/2n)	**24 + 8n** (6/2n)	**32 + 8n** (8/2n)	– –	– –
	Long	**16 + 16n** (4/4n)	– –	**16 + 16n** (4/4n)	**24 + 16n** (6/4n)	**26 + 16n** (6/4n)	**24 + 16n** (8/4n)	**32 + 16n** (6/4n)	– –	– –

n is the number of registers to move
* is the size of the index register (ix) does not affect the instruction's execution time

Table 7.14 : Multi-Precision Instruction Execution Times.

Instruction	Size	op Dn, Dn	op M, M
ADDX	Byte	**8**(2/0)	**22**(4/1)
	Word	**8**(2/0)	**50**(6/2)
	Long	**12**(2/0)	**58**(10/4)
CMPM	Byte	–	**16**(4/0)
	Word	–	**24**(6/0)
	Long	–	**40**(10/0)
SUBX	Byte	**8**(2/0)	**22**(4/1)
	Word	**8**(2/0)	**50**(6/2)
	Long	**12**(2/0)	**58**(10/4)
ABCD	Byte	**10**(2/0)	**20**(4/1)
SBCD	Byte	**10**(2/0)	**20**(4/1)

Table 7.15 : Miscellaneous Instruction Execution Times.

Instruction	Register	Memory
ANDI to CCR	**32**(6/0)	
ANDI to SR	**32**(6/0)	
EORI to CCR	**32**(6/0)	
EORI to SR	**32**(6/0)	
EXG	**10**(2/0)	
EXT	**8**(2/0)	
LINK	**32**(4/4)	
MOVE to CCR	**18**(4/0)	**18**(4/0) +
MOVE to SR	**18**(4/0)	**18**(4/0) +
MOVE from SR	**10**(2/0)	**16**(2/2) +
MOVE to USP	**8**(2/0)	
MOVE from USP	**8**(2/0)	
NOP	**8**(2/0)	
ORI to CCR	**32**(6/0)	
ORI to SR	**32**(6/0)	
RESET	**136**(2/0)	
RTE	**40**(10/0)	
RTR	**40**(10/0)	
RTS	**32**(8/0)	
STOP	**4**(0/0)	
SWAP	**8**(2/0)	
UNLK	**24**(6/0)	

+ add effective address calculation time

Table 7.16 : Move Peripheral Instruction Excecution Times.

Instruction	Size	Register → Memory	Memory → Register
MOVEP	Word	**24**(4/2)	**24**(6/0)
	Long	**32**(4/4)	**32**(8/0

+ add effective address calculation time

7.2.12. EXCEPTION PROCESSING EXECUTION TIMES. Table 7.17 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 instruction of the handler routine. The number of bus read and write cycles is shown in parenthesis as : (r/w).

Table 7.17 : Exception Processing Execution Times.

Exception	Periods
Address Error	**94**(8/14)
Bus Error	**94**(8/14)
CHK Instruction	**68**(8/6) +
Interrupt	**72**(9/16) *
Illegal Instruction	**62**(8/6)
Privileged Instruction	**62**(8/6)
Trace	**62**(8/6)
TRAP Instruction	**62**(8/6)
TRAPV Instruction	**66**(10/6)
Divide by Zero	**66**(8/6) +
$\overline{\text{RESET}}$**	**64**(12/0)

+ add effective address calculation time
* The interrupt acknowledge bus cycle is assumed to take four external clock periods
** Indicates the time from when $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ are first sampled as negated to when instruction execution starts.

SECTION 8

ELECTRICAL SPECIFICATIONS

This section contains the electrical specifications and associated timing information for the MC68008.

8.1. ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	– 0.3 to 7	V
V_{IN}	Input Voltage	– 0.3 to 7	V
T_A	Operating Temperature Range TS68008C TS68008V	0 to 70 0 to 70 – 40 to 65	°C
T_{stg}	Storage Temperature	– 55 to 150	°C

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 voltages 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 ground or V_{CC}).

8.2 THERMAL DATA

Parameter	Value		Unit
	θ_{JA}	θ_{JC}	
Thermal Resistance :			
Plastic DIL	40	20*	°C/W
PLCC	50	30*	

* Estimated

8.3. POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from :

$T_J = T_A + (P_D \cdot \theta_{JA})$ (1)

Where :

T_A = Ambient Temperature, °C

θJ_A = 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 :

$P_D = K \div (T_J + 273°C)$ (2)

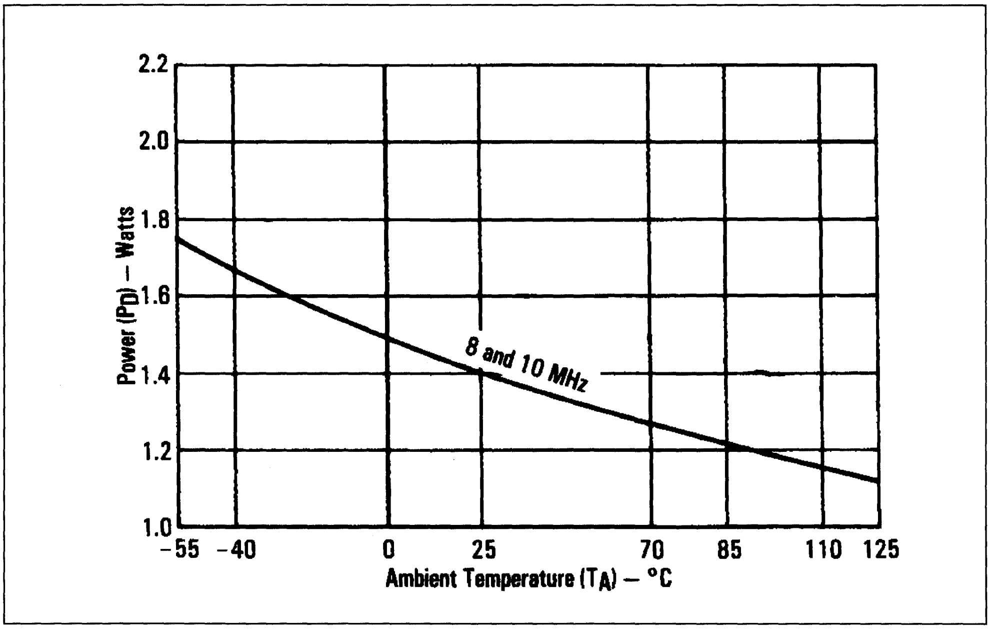
Solving equations 1 and 2 for K gives :

$K = P_D \cdot (T_A + 273°C) + \theta JA \cdot P_D^2$ (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 8.1 gives the graphic solution to these equations for the specification power dissipation of 1.50 watts over the ambient temperature range of –55°C to 125°C using a θ_{JA} of 45°C/W, a typical value for packages specified.

Figure 8.1 : TS68008 Power Dissipation (P_D) vs Ambient Temperature (T_A).



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 :

$$\theta_{JA} = \theta_{JC} + \theta_{CA}$$

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 $\theta_{JA} = \theta_{JC}$. Substitution of θ_{JC} for θ_{JA} in equation 1 will result in a lower semiconductor junction temperature.

Figure 8.2 : $\overline{\text{RESET}}$ Test Load.

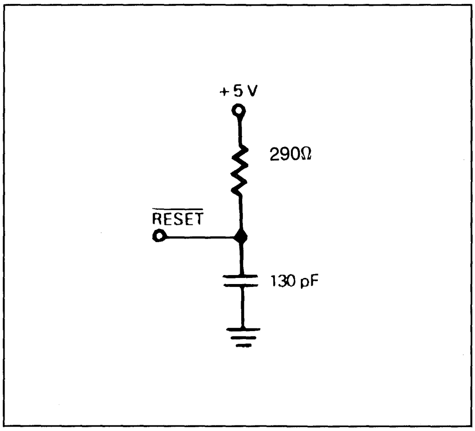


Figure 8.3 : $\overline{\text{HALT}}$ Test Load.

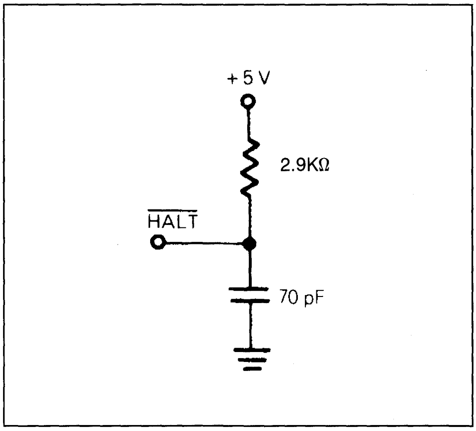
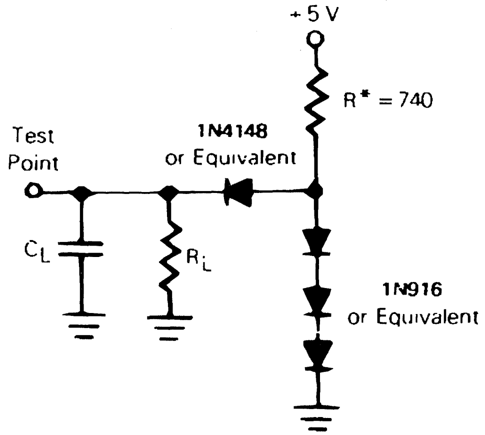


Figure 8.4 : Test Loads.



CL = 130pF (includes all parasitics)
RL = 60KΩ for $\overline{AS}$, A0–A19, $\overline{BG}$, D0–D7, E, FC0–FC2, $\overline{DS}$, R/$\overline{W}$
* R = 1.22KΩ for A0–A19, $\overline{BG}$, FC0–FC2

8.4. DC ELECTRICAL CHARACTERISTICS
(V_{CC} = 5.0 Vdc ± 5 % ; GND = 0 Vdc ; T_A = 0 °C to 70 °C; see figures 8.2, 8.3 and 8.4)

Symbol	Parameter	Min.	Max.	Unit
V_{IH}	Input High Voltage	2.0	V_{CC}	V
V_{IL}	Input Low Voltage	GND – 0.3	0.8	V
I_{in}	Input Leakage Current @ 5.25 V $\overline{BERR}$, $\overline{BR}$, $\overline{DTACK}$, CLK, $\overline{IPL0/2}$, $\overline{IPL1}$, $\overline{VPA}$, $\overline{HALT}$, $\overline{RESET}$ $\overline{BGACK}$	–	20	µA
I_{TSI}	Hi-Z (off state) Input Current @ 2.4 V/0.4 V A0-A19, $\overline{AS}$, D0-D7, FC0-FC2, $\overline{DS}$, R/$\overline{W}$	–	20	µA
V_{OH}	Output High Voltage (I_{OH} = – 400 µA) E, A0-A19, $\overline{AS}$, $\overline{BG}$, D0-D7, FC0-FC2, $\overline{DS}$, R/$\overline{W}$, $\overline{VMA}$	2.4	–	V
V_{OL}	Output Low Voltage (I_{OL} = 1.6 mA) $\overline{HALT}$ (I_{OL} = 3.2 mA) A0-A19, $\overline{BG}$, FC0-FC2 (I_{OL} = 5.0 mA) $\overline{RESET}$ (I_{OL} = 5.3 mA) E, $\overline{AS}$, D0-D7, $\overline{DS}$, R/$\overline{W}$	 – – – –	 0.5 0.5 0.5 0.5	V
P_D	Power Dissipation, *T_A = 0°C	–	1.5	W
C_{IN}	Capacitance (V_{in} = 0 V, T_A = 25 °C ; frequency = 1 MHz)**	–	20.0	pF

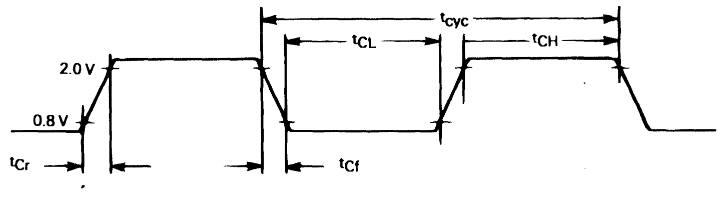
* During normal operation instantaneous VCC current requirements may be as high as 1.5A
* * Capacitance is periodically sampled rather than 100% tested.

SGS-THOMSON MICROELECTRONICS

8.5 CLOCK TIMING (see figure 8.5)

Symbol	Parameter	8 MHz		10 MHz		Unit
		Min.	Max.	Min.	Max.	
f	Frequency of Operation	2.0	8.0	2.0	10.0	MHz
t_{cyc}	Cycle Time	125	500	100	500	ns
t_{CL}	Clock Pulse Width	55	250	45	250	ns
t_{CH}		55	250	45	250	ns
t_{Cr}	Rise and Fall Times	–	10	–	10	ns
t_{Cf}		–	10	–	10	ns

Figure 8.5 : Input Clock Waveform.



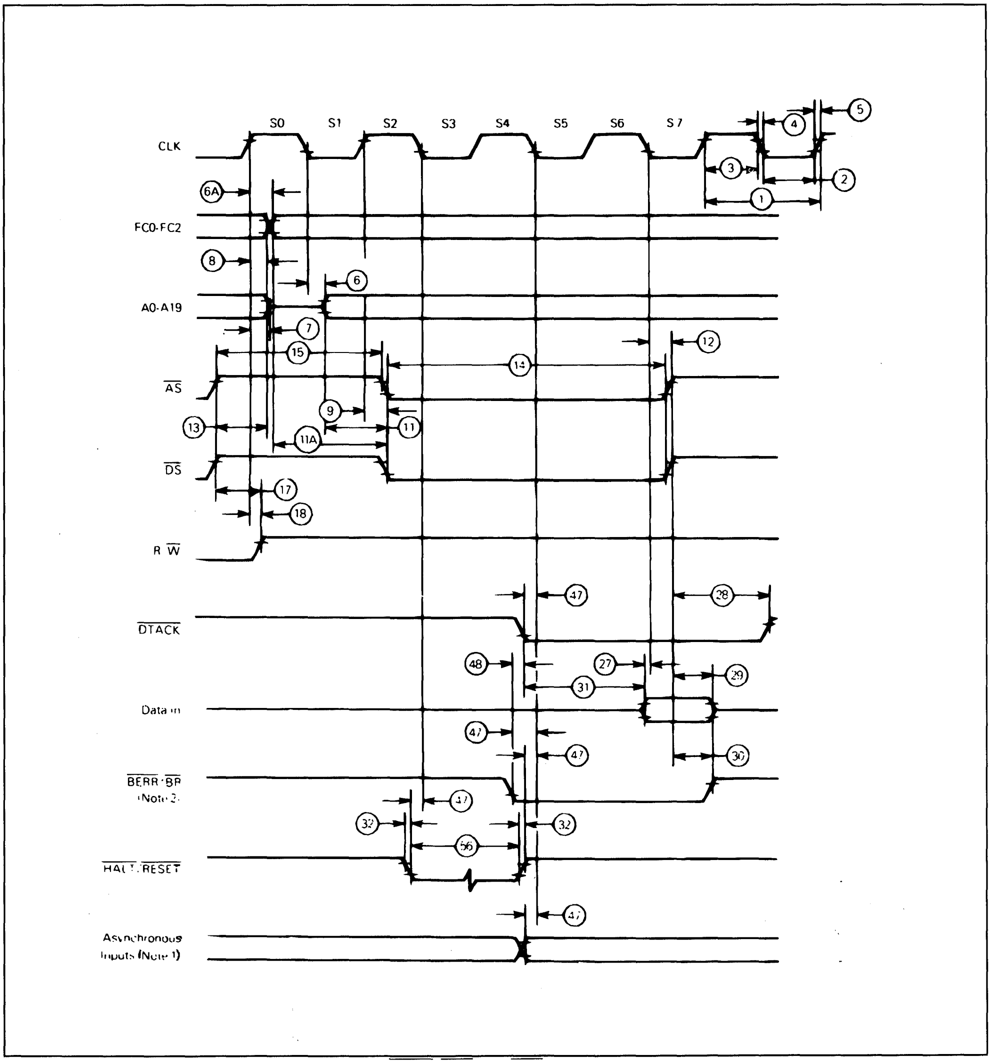
8.6. AC ELECTRICAL SPECIFICATIONS – READ CYCLES
(V_{CC} = 5.0V_{dC} ± 5% ; GND = 0Vdc ; T_A = T_L to T_H ; see figure 8.6)

N°	Symbol	Parameter	8MHz		10MHz		Unit
			Min.	Max.	Min.	Max.	
1	t_{CYC}	Clock Period	125	500	100	500	ns
2	t_{CL}	Clock Width Low	55	250	45	250	ns
3	t_{CH}	Clock Width High	55	250	45	250	ns
4	t_{Ct}	Clock Fall Time		10		10	ns
5	t_{Cr}	Clock Rise Time		10		10	ns
6	t_{CLAV}	Clock Low to Address Valid		70		60	ns
6A	t_{CHFCV}	Clock High to FC Valid		70		60	ns
7	t_{CHADZ}	Clock High to Address, Data Bus High Impedance (maximum)		80		70	ns
8	t_{CHAFI}	Clock High to Address, FC Invalid (minimum)	0		0		ns
9(1)	t_{CHSL}	Clock High to $\overline{AS}$, $\overline{DS}$ Low	0	60	0	55	ns
11(2)	t_{AVSL}	Address Valid to $\overline{AS}$, $\overline{DS}$ Low	30		20		ns
11A(2,6)	t_{FCVSL}	FC Valid to $\overline{AS}$, $\overline{DS}$ Low	60		50		ns
12(1)	t_{CLSH}	Clock Low to $\overline{AS}$, $\overline{DS}$ High		35		35	ns
13(2)	t_{SHARI}	$\overline{AS}$, $\overline{DS}$ High to Address/FC Invalid	30		20		ns
14(2,5)	t_{SL}	$\overline{AS}$, $\overline{DS}$ Width Low	270		195		ns
15(2)	t_{SH}	$\overline{AS}$, $\overline{DS}$ Width High	150		105		ns
17(2)	t_{SHRH}	$\overline{AS}$, $\overline{DS}$ High to R/$\overline{W}$ High	40		20		ns
18(1)	t_{CHRH}	Clock High to R/$\overline{W}$ High	0	40	0	40	ns
27(5)	t_{DICL}	Data In to Clock Low (setup time)	15		10		ns
28(2,5)	t_{SHDAH}	$\overline{AS}$, $\overline{DS}$ High to $\overline{DTACK}$ High	0	245	0	190	ns
29	t_{SHDII}	$\overline{AS}$, $\overline{DS}$ High to Data in Invalid (hold time)	0		0		ns
30	t_{SHBEH}	$\overline{AS}$, $\overline{DS}$ High to $\overline{BERR}$ High	0		0		ns
31(2,5)	t_{DALDI}	$\overline{DTACK}$ Low to Data Valid (asynchronous setup time on read)		90		65	ns
32	$t_{RHr,f}$	$\overline{HALT}$ and $\overline{RESET}$ Input Transition Time	0	200	0	200	ns
47(5)	t_{ASI}	Asynchronous Input Setup Time	10		10		ns
48(3)	t_{BELDAL}	$\overline{BERR}$ Low to $\overline{DTACK}$ Low	20		20		ns
56(4)	t_{HRPW}	$\overline{HALT}$/$\overline{RESET}$ Pulse Width	10		10		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 100 milliseconds 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 setup time (47) requirements are satisfied, the $\overline{DTACK}$ low-to-data setup time (31) requirement can be ignored. The data must only satisfy the data-in to clock-low setup time (27) for the following cycle.
6. Setup time to guarantee recognition on next falling edge of 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.

Figure 8.6 : Read Cycle Timing Diagram.



Notes :
1. Setup time for the asynchronous inputs $\overline{IPLO2}$, $\overline{IPL1}$, and $\overline{VPA}$ 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 volt and a high voltage of 2.0 volts, unless otherwise noted.

8.6. AC ELECTRICAL SPECIFICATIONS – $\overline{\text{WRITE CYCLES}}$

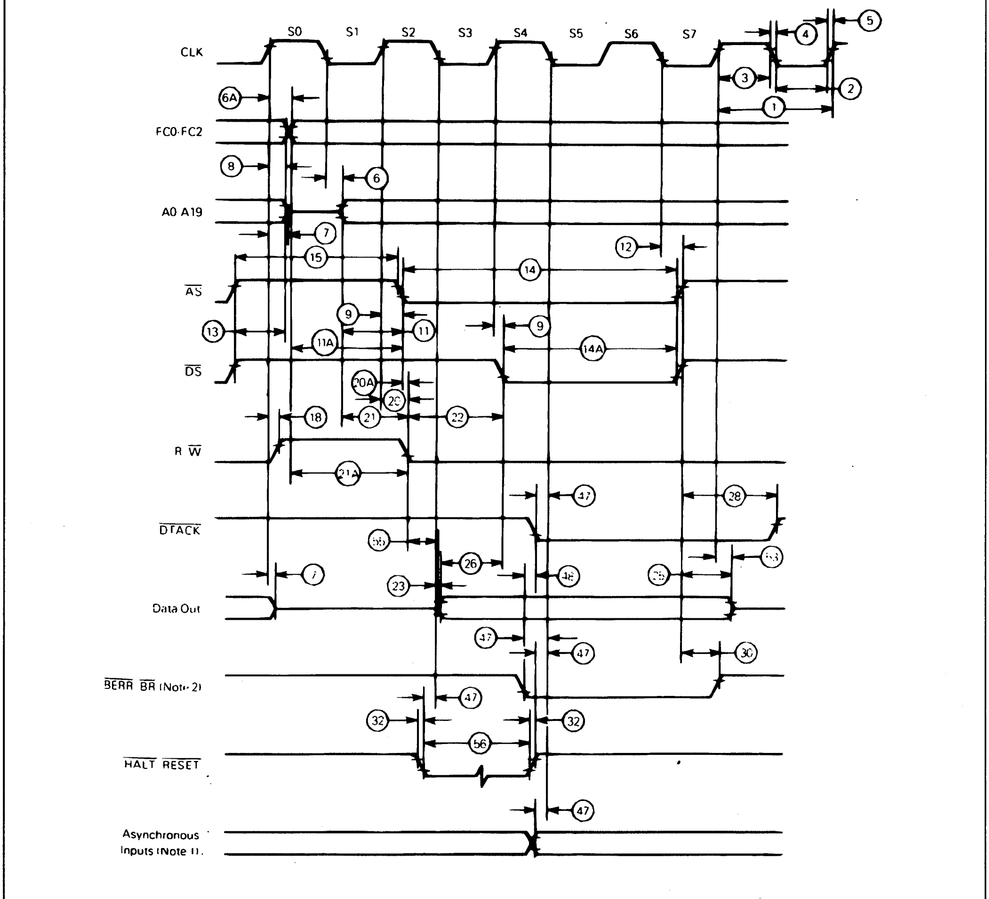
(V_{CC} = 5.0 V_{dC} ± 5% ; GND = 0Vdc ; T_A = T_L to T_H ; see figure 8.7)

N°	Symbol	Parameter	8MHz		10MHz		Unit
			Min.	Max.	Min.	Max.	
1	t_{CYC}	Clock Period	125	500	100	500	ns
2	t_{CL}	Clock Width Low	55	250	45	250	ns
3	t_{CH}	Clock Width High	55	250	45	250	ns
4	t_{Ct}	Clock Fall Time		10		10	ns
5	t_{Cr}	Clock Rise Time		10		10	ns
6	t_{CLAV}	Clock Low to Address Valid		70		60	ns
6A	t_{CHFCV}	Clock High to FC Valid		70		60	ns
7	t_{CHADZ}	Clock High to Address, Data Bus High Impedance (maximum)		80		70	ns
8	t_{CHAFI}	Clock High to Address, FC Invalid (minimum)	0		0		ns
9(1)	t_{CHSL}	Clock High to $\overline{\text{AS}}$, $\overline{\text{DS}}$ Low	0	60	0	55	ns
11(2)	t_{AVSL}	Address Valid to $\overline{\text{AS}}$ Low	30		20		ns
11A(2,7)	t_{FCVSL}	FC Valid to $\overline{\text{AS}}$ Low	60		50		ns
12(1)	t_{CLSH}	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$ High		35		35	ns
13(2)	t_{SHARI}	$\overline{\text{AS}}$, $\overline{\text{DS}}$ High to Address/FC Invalid	30		20		ns
14(2,5)	t_{SL}	$\overline{\text{AS}}$ Low	270		195		ns
14A(2)	t_{DSL}	$\overline{\text{DS}}$ Width Low	140		95		ns
15(2)	t_{SH}	$\overline{\text{AS}}$, $\overline{\text{DS}}$ Width High	150		105		ns
18(1)	t_{CHRH}	Clock High to R/$\overline{\text{W}}$ High	0	40	0	40	ns
20(1)	t_{CHRL}	Clock High to R/$\overline{\text{W}}$ Low		40		40	ns
20A(6)	t_{ASRV}	$\overline{\text{AS}}$, Low to R/$\overline{\text{W}}$ Valid		20		20	ns
21(2)	t_{AVRL}	Address Valid to R/$\overline{\text{W}}$ Low	20		0		ns
21A(2,7)	t_{FCVRL}	FC Valid to R/$\overline{\text{W}}$ Low	60		50		ns
22(2)	t_{RLSL}	R/$\overline{\text{W}}$ Low to $\overline{\text{DS}}$ Low	80		50		ns
23	t_{CLDO}	Clock Low to Data Out Valid		70		55	ns
25(2)	t_{SHDOI}	$\overline{\text{AS}}$, $\overline{\text{DS}}$ High to Data Out Invalid	50		20		ns
26(2)	t_{DOSL}	Data Out Valid to $\overline{\text{DS}}$ Low	35		20		ns
28(2,5)	t_{SHDAH}	$\overline{\text{AS}}$, $\overline{\text{DS}}$ High to $\overline{\text{DTACK}}$ High	0	245	0	190	ns
29	t_{SHDII}	$\overline{\text{AS}}$, $\overline{\text{DS}}$ High to Data in Invalid (hold time)	0		0		ns
30	t_{SHBEH}	$\overline{\text{AS}}$, $\overline{\text{DS}}$ High to $\overline{\text{BERR}}$ High	0		0		ns
32	$t_{RHr,f}$	$\overline{\text{HALT}}$ and $\overline{\text{RESET}}$ Input Transition Time	0	200	0	200	ns
47(5)	t_{ASI}	Asynchronous Input Setup Time	10		10		ns
48(3)	t_{BELDAL}	$\overline{\text{BERR}}$ Low to $\overline{\text{DTACK}}$ Low	20		20		ns
53	t_{CHDOI}	Clock High to Data Out Invalid	0		0		ns
55	t_{RLDBD}	R/$\overline{\text{W}}$ to Data Bus Impedance Driven	30		20		ns
56(4)	t_{HRPW}	$\overline{\text{HALT}}/\overline{\text{RESET}}$ Pulse Width	10		10		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 $\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$, 48 may be 0 nanoseconds.
4. For power up the MPU must be held in $\overline{\text{RESET}}$ state for 100 milliseconds 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 setup time (47) requirements are satisfied, the $\overline{\text{DTACK}}$ low-to-data setup time (31) requirement can be ignored. The data must only satisfy the data-in to clock-low setup time (27) for the following cycle.
6. When AS, and R/W are equally loaded (± 20%), subtract 10 nanoseconds from the values in these columns.
7. Setup time to guarantee recognition on next falling edge of 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.

Figure 8.7 : Write Cycle Timing Diagram.



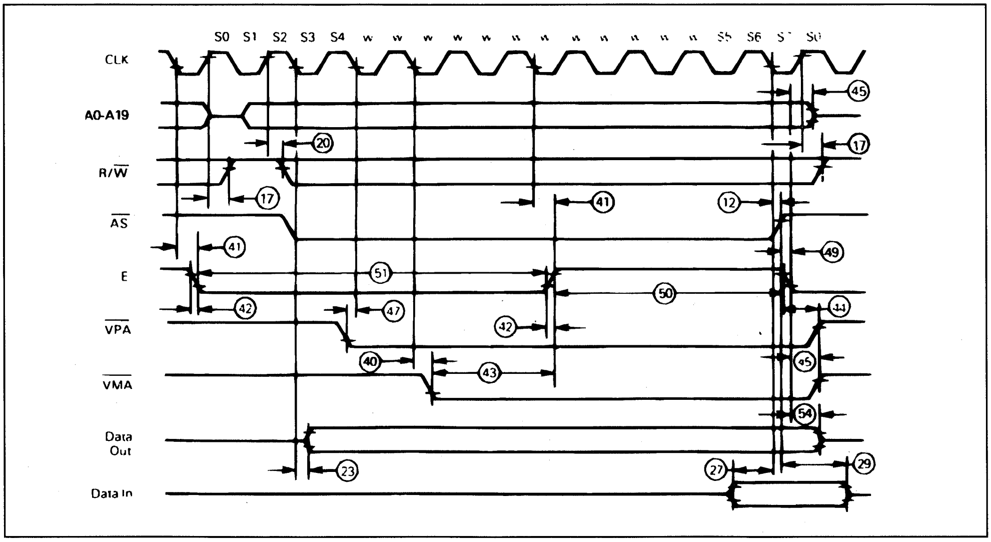
Notes :
1. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.
2. Because of loading variations, R/$\overline{\text{W}}$ may be valid after $\overline{\text{AS}}$ even though both are initiated by the rising edge of S2 (Specification 20A).

8.7. AC ELECTRICAL SPECIFICATIONS – TS 68008 to 6800 PERIPHERAL (V_{CC} = 5.0 V_{dC} ± 5% ; GND = 0Vdc ; T_A = 0° To 70°C ; see figures 8.8 and 8.9)

N°	Symbol	Parameter	8MHz		10MHz		Unit
			Min.	Max.	Min.	Max.	
12[(1)]	t_{CLSH}	Clock Low to $\overline{AS}$, $\overline{DS}$ High		35		35	ns
17[(2)]	t_{SHRH}	$\overline{AS}$, $\overline{DS}$ High to R/$\overline{W}$ High (read)	40		20		ns
18[(1)]	t_{CHRH}	Clock High to R/$\overline{W}$ High	0	40	0	40	ns
20[(1)]	t_{CHRL}	Clock High to R/$\overline{W}$ Low		40		40	ns
23	t_{CLDO}	Clock Low to Data Out Valid (write)		70		55	ns
27	t_{DICL}	Data In to Clock Low (setup time on read)	15		10		ns
29	t_{SHDII}	$\overline{AS}$, $\overline{DS}$ High to Data in Invalid (hold time on read)	0		0		ns
41	t_{CLET}	Clock Low to E Transition		50		50	ns
42	$t_{Er,f}$	E Output Rise and Fall Time		15		15	ns
44	t_{SHVPH}	$\overline{AS}$, $\overline{DS}$ High to $\overline{VPA}$ High	0	120	0	90	ns
45	t_{ELCAI}	E Low to Control, Address Bus Invalid (address hold time)	30		10		ns
47	t_{ASI}	Asynchronous Input Setup Time 10	10		10		ns
49[(3)]	t_{SHEL}	$\overline{AS}$, $\overline{DS}$ High to E Low	– 80	80	– 80	80	ns
50	t_{EH}	E Width High	450		350		ns
51	t_{EL}	E Width Low	700		550		ns
54	t_{ELDOI}	E Low to Data Out Invalid	30		20		ns

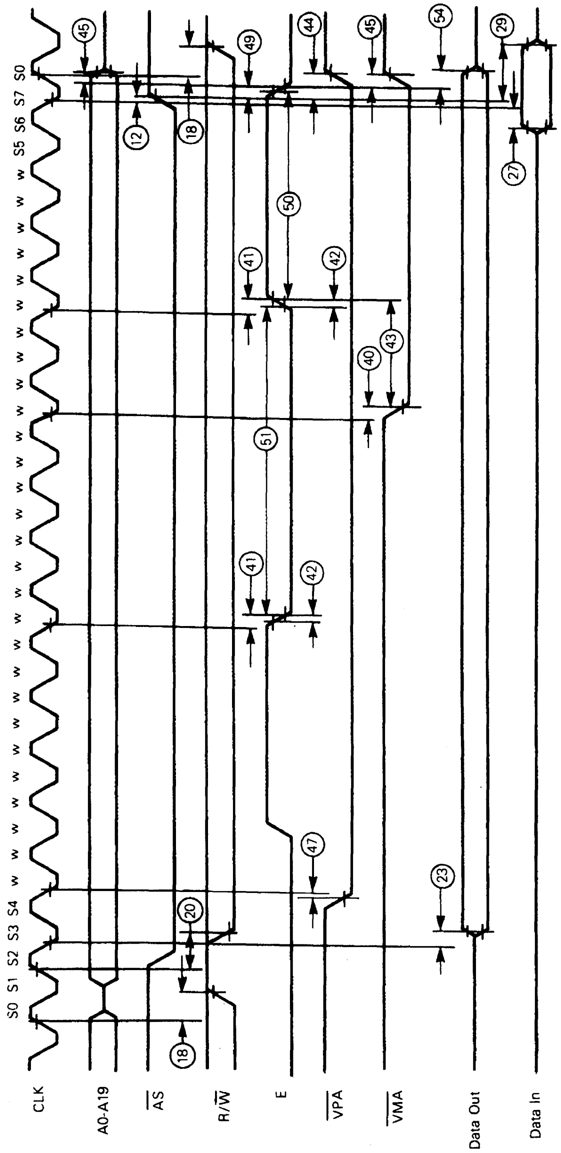
Notes :
1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the values given in these columns.
2. Actual value depends on clock period.
3. The falling edge of S6 triggers both the negation of the strobes ($\overline{AS}$, and x $\overline{DS}$) and the falling 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.

Figure 8.8 : TS68008 to 6800 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 best case possibly attainable.

Figure 8.9 : TS68008 to 6800 Peripheral Timing Diagram – Worst Case.


CLK
A0-A19
AS
R/W
E
VPA
VMA
Data Out
Data In
S0 S1 S2 S3 S4 w w w w w w w w w w w w w w w w w w w w w w w w w w w w w w S5 S6 S7 S0
12
18
20
23
27
29
40
41
42
43
44
45
47
49
50
51
54

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.

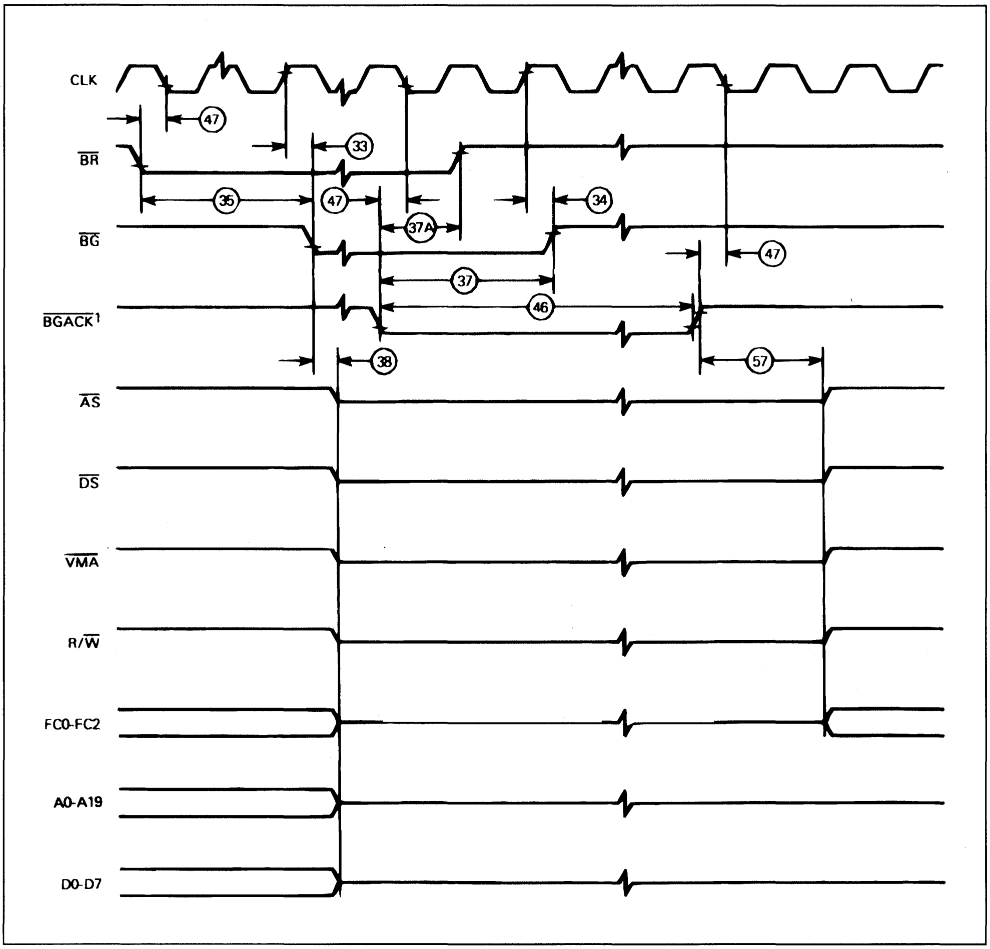
8.8. AC ELECTRICAL SPECIFICATIONS – BUS ARBITRATION
(V_{CC} = 5.0 V_{dC} ± 5% ; GND = 0Vdc ; T_A = T_L to T_H ; see figures 8.10, 8.11, and 8.12)

N°	Symbol	Parameter	8MHz		10MHz		Unit
			Min.	Max.	Min.	Max.	
7	t_{CHADZ}	Clock High to Address, Data Bus High Impedance		80		70	ns
16	t_{CHCZ}	Clock High to Control Bus High Impedance		80		70	ns
33	t_{CHGL}	Clock High to $\overline{BG}$ Low		40		40	ns
34	t_{CHGH}	Clock High to $\overline{BG}$ High		40		40	ns
35	t_{BRLGL}	$\overline{BR}$, Low to $\overline{BG}$ Low	1.5	90ns + 3.5	1.5	80ns + 3.5	Clk.Per.
36(1)	t_{BRHGH}	$\overline{BR}$ High to $\overline{BG}$ High	1.5	90ns + 3.5	1.5	80ns + 3.5	Clk.Per.
37	t_{GALGH}	$\overline{BGACK}$ Low to $\overline{BG}$ High (52-pin version only)	1.5	90ns + 3.5	1.5	80ns + 3.5	Clk.Per.
37A(2)	t_{GALBRH}	$\overline{BGACK}$ Low to $\overline{BR}$ High (52-pin version only)	20	1.5 Clocks	20	1.5 Clocks	ns
38	t_{GLZ}	$\overline{BG}$ Low to Control, Address, Data Bus High Impedance ($\overline{AS}$ high)		80		70	ns
39	t_{GH}	$\overline{BG}$ Width High	1.5		1.5		Clk.Per.
46	t_{GAL}	$\overline{BGACK}$ Width Low (52-pin version only)	1.5		1.5		Clk.Per.
47	t_{ASI}	Asynchronous Input Setup Time 10	10		10		ns
57	t_{GABD}	$\overline{BGACK}$ High to Control Bus Driven (52-pin version only)	1.5		1.5		Clk.Per.
58(1)	t_{GHBD}	BG High to Control Bus Driven	1.5		1.5		Clk.Per.

Notes : 1. For 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 garantee proper operation. If the maximum value 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.

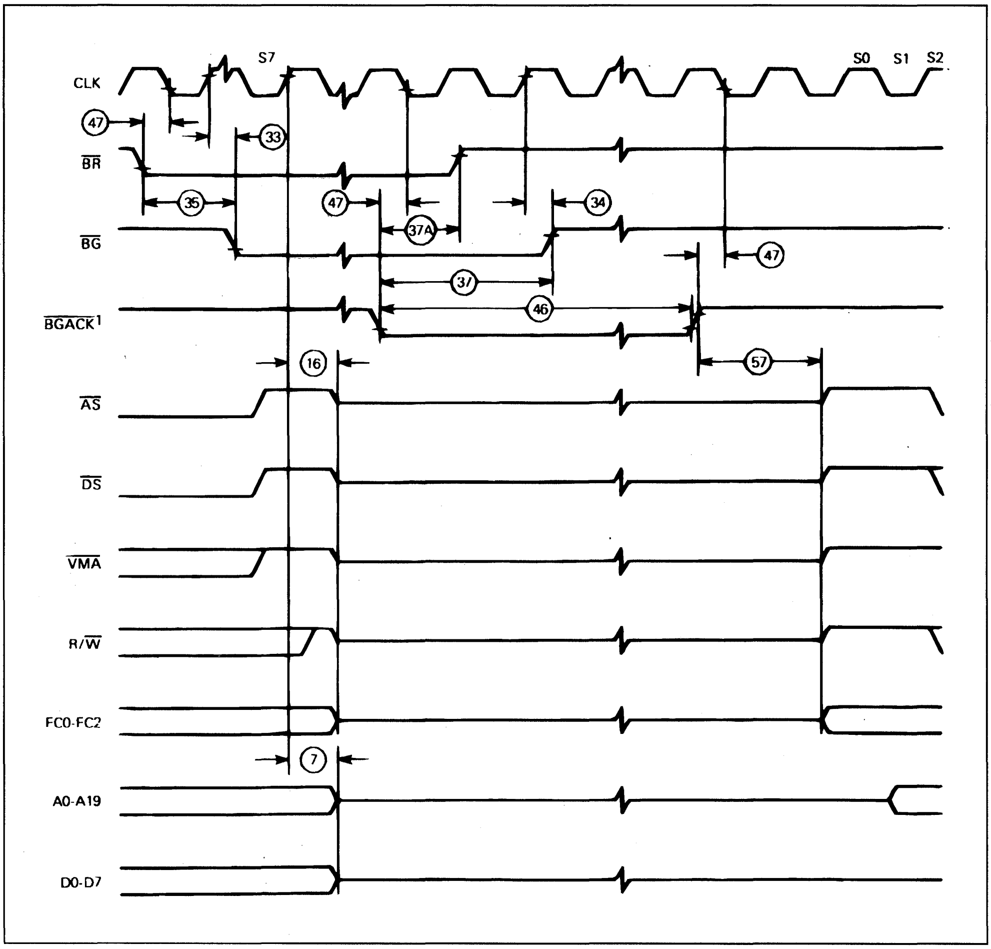
Figure 8.10 : Bus Arbitration Timing – Idle Bus Case.



Note 1: 52-Pin version only

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.

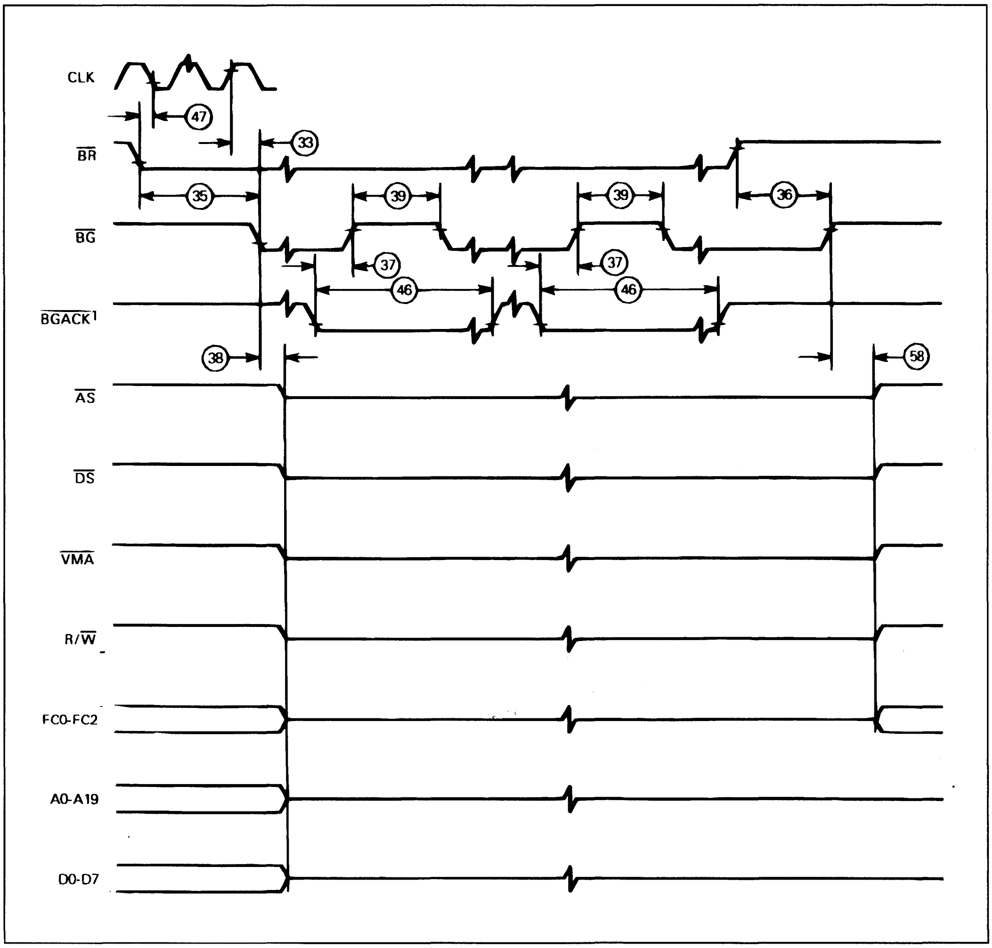
Figure 8.11 : Bus Arbitration Timing – Active Bus Case.



Note 1: 52-Pin version only

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 8.12 : Bus Arbitration Timing – Multiple Bus Requests (52 pin version only).



Note 1: 52-Pin version only

SECTION 9

ORDERING INFORMATION

This section contains detailed information to be used as a guide when ordering the TS68008

9.1. STANDARD VERSIONS

Part Number	Frequency(MHz)	Temperature Range	Package Type
TS68008 CP8	8.0	0 °C to + 70 °C	Plastic DIL
TS68008 VP8	8.0	– 40 °C to + 85 °C	P. Suffix
TS68008 CP10	10.0	0 °C to + 70 °C	
TS68008 VP10	10.0	– 40 °C to + 85 °C	
TS68008 CFN8	8.0	0 °C to + 70 °C	PLCC
TS68008 VFN8	8.0	– 40 °C to + 85 °C	FN Suffix
TS68008 CFN10	10.0	0 °C to + 70 °C	
TS68008 VFN10	10.0	– 40 °C to + 85 °C	

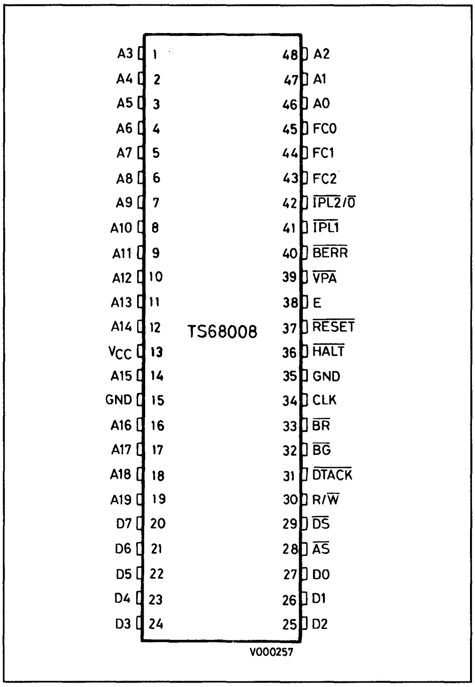
SECTION 10

MECHANICAL DATA

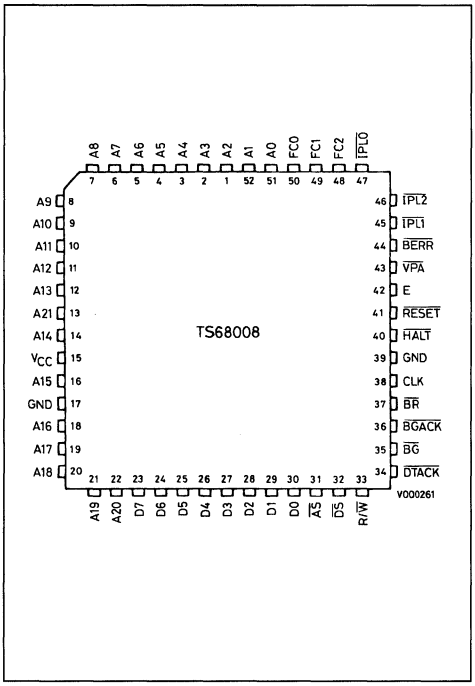
This section contains the pin assignments and package dimensions for the TS68008.

10.1. PIN ASSIGNMENTS

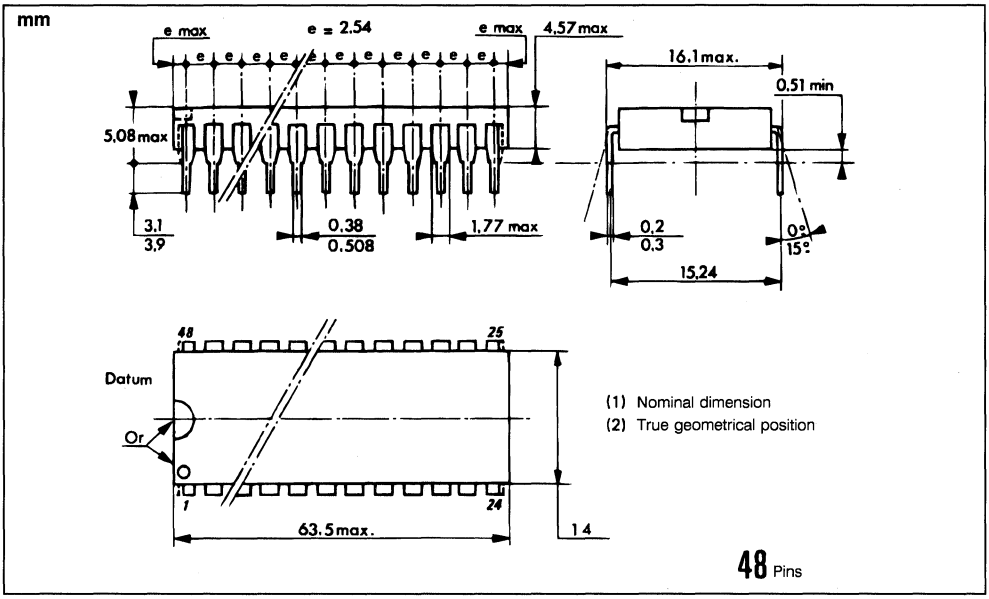
48 –Pin Dual–in–Line

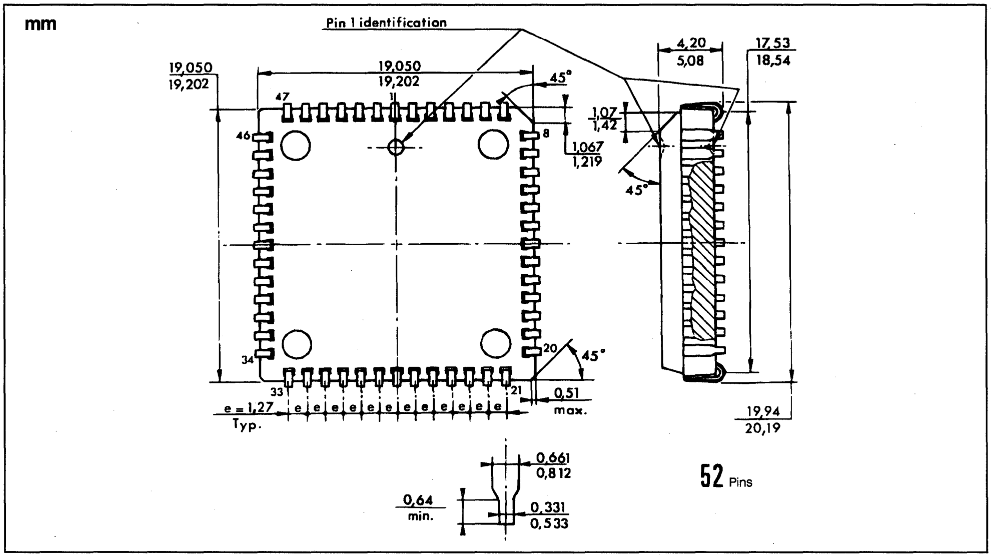


52 –Pin Quad Pack (PLCC)



10.2. PACKAGE DIMENSIONS



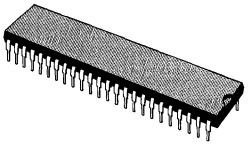


68000 PERIPHERALS

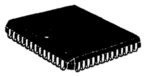
TS68230

HMOS PARALLEL INTERFACE/TIMER

- TS68000 BUS COMPATIBLE
- PORT MODES INCLUDE :
  BIT I/O
  UNIDIRECTIONAL 8 BIT AND 16 BIT
  BIDIRECTIONAL 8 BIT AND 16 BIT
- PROGRAMMABLE HANDSHAKING OPTIONS
- 24-BIT PROGRAMMABLE TIMER MODES
- FIVE SEPARATE INTERRUPT VECTORS
- SEPARATE PORT AND TIMER INTERRUPT SERVICE REQUESTS
- REGISTERS ARE READ/WRITE AND DIRECTLY ADDRESSABLE
- REGISTERS ARE ADDRESSED FOR MOVEP (Move Peripheral) AND DMAC COMPATIBILITY



P
(Plastic Package)

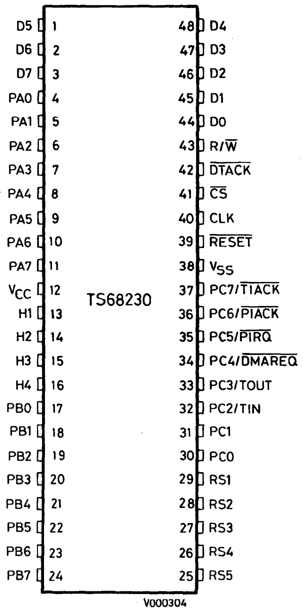


FN
(PLCC52)

DESCRIPTION

The TS68230 parallel interface/timer (PI/T) provides versatile double buffered parallel interfaces and a system oriented timer for TS68000 systems. The parallel interfaces operate in unidirectional or bidirectional modes, either 8 or 16 bits wide. In the unidirectional modes, an associated data direction register determines whether each port pin is an input or output. In the bidirectional modes the data direction registers are ignored and the direction is determined dynamically by the state of four handshake pins. These programmable handshake pins provide an interface flexible enough for connection to a wide variety of low, medium, or high speed peripherals or other computer systems. The PI/T ports allow use of vectored or auto-vectored interrupts, and also provide a DMA request pin for connection to the 68440 direct memory access controller (DMAC) or a similar circuit. The PI/T timer contains a 24-bit wide counter and a 5-bit prescaler. The timer may be clocked by the system clock (PI/T CLK pin) or by an external clock (TIN pin), and a 5-bit prescaler can be used. It can generate periodic interrupts, a square wave, or a single interrupt after a programmed time period. It can also be used for elapsed time measurement or as a device watchdog.

PIN CONNECTIONS



SECTION 1

INTRODUCTION

The TS68230 parallel interface/timer (PI/T) provides versatile double buffered parallel interfaces and a system oriented timer for TS68000 systems. The parallel interfaces operate in unidirectional or bidirectional modes, either 8 or 16 bits wide. In the unidirectional modes, an associated data direction register determines whether each port pin is an input or output. In the bidirectional modes the data direction registers are ignored and the direction is determined dynamically by the state of four handshake pins. These programmable handshake pins provide an interface flexible enough for connection to a wide variety of low, medium, or high speed peripherals or other computer systems. The PI/T ports allow use of vectored or autovectored interrupts, and also provide a DMA request pin for connection to the 68440 direct memory access controller (DMAC) or a similar circuit. The PI/T timer contains a 24-bit wide counter and a 5-bit prescaler. The timer may be clocked by the system clock (PI/T CLK pin) or by an external clock (TIN pin), and a 5-bit prescaler can be used. It can generate periodic interrupts, a square wave, or a single interrupt after a programmed time period. It can also be used for elapsed time measurement or as a device watchdog.

Features of the PI/T include :

- TS68000 Bus Compatible
- Port Modes Include :
  Bit I/O
  Unidirectional 8 Bit and 16 Bit
  Bidirectional 8 Bit and 16 Bit
- Programmable Handshaking Options
- 24-Bit Programmable Timer Modes
- Five Separate Interrupt Vectors
- Separate Port and Timer Interrupt Service Requests
- Registers are Read/Write and Directly Addressable
- Registers are Addressed for MOVEP (Move Peripheral) and DMAC Compatibility

The PI/T consists of two logically independent sections : the ports and the timer. The port section consists of port A (PA0-PA7), port B (PB0-PB7), four handshake pins (H1, H2, H3, and H4), two general input/output (I/O) pins, and six dual-function pins. The dual-function pins can individually operate as a third port (port C) or an alternate function related to either port A, port B, or the timer. The four programmable handshake pins, depending on the mode, can control data transfer to and from the ports, or can be used as interrupt generating inputs or I/O pins. Refer to figure 1.1.

The timer consists of a 24-bit counter, optionally clocked by a 5-bit prescaler. Three pins provide complete timer I/O : PC2/TIN, PC3/TOUT, and PC7/TIACK. Only the ones needed for the given configuration perform the timer function, while the others remain port C I/O.

The system bus interface provides for asynchronous transfer of data from the PI/T to a bus master over the data bus (D0-D7). Data transfer acknowledge (DTACK), register selects (RS1-RS5), timer interrupt acknowledge (TIACK), read/write line (R/W), chip select (CS), or port interrupt acknowledge (PIACK) control data transfer between the PI/T and an TS68000.

1.1. PORT MODE DESCRIPTION

The primary focus of most applications will be on port A, port B, the handshake pins, the port interrupt pins, and the DMA request pin. They are controlled in the following way : the port general control register contains a 2-bit field that specifies one of four operation modes. These govern the overall operation of the ports and determine their interrelationships. Some modes require additional information from each port's control register to further define its operation. In each port control register, there is a 2-bit submode field that serves this purpose. Each port mode/submode combination specifies a set of programmable characteristics that fully define the behavior of that port and two of the handshake pins. This structure is summarized in table 1.1 and figure 1.2.

Figure 1.1 : Block Diagram.

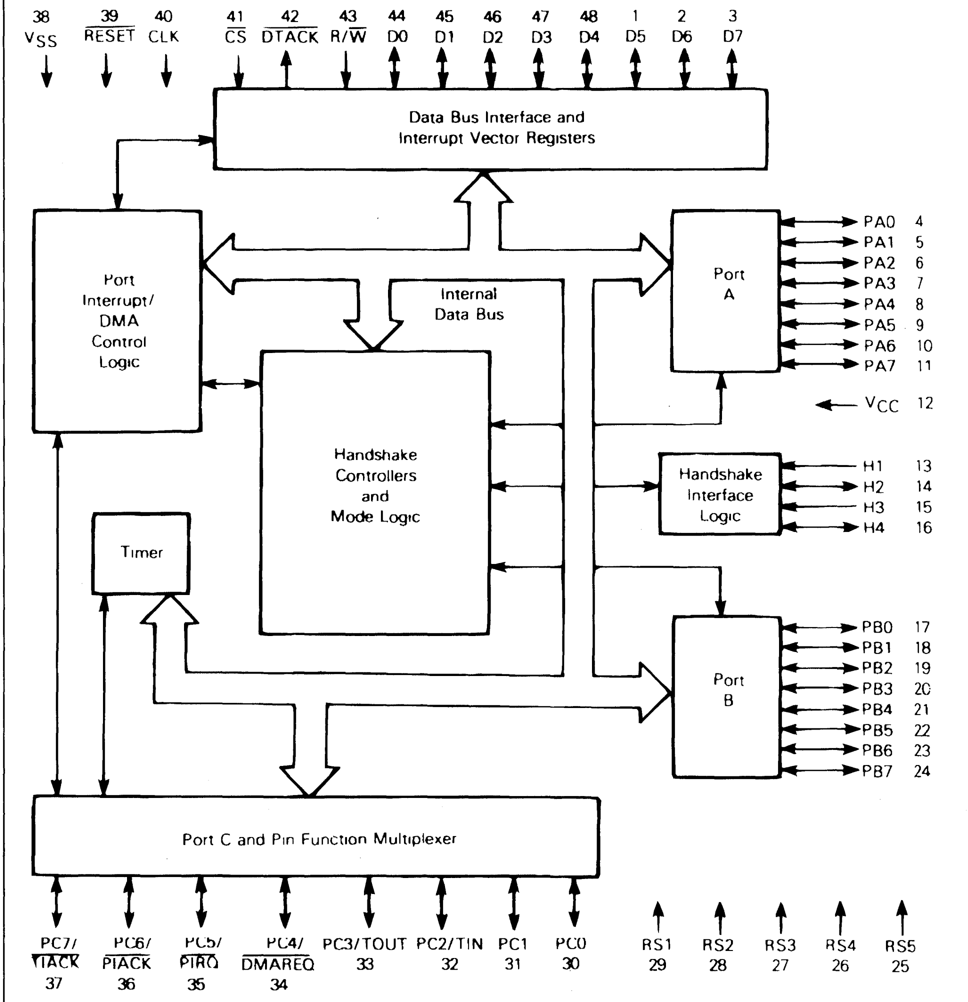


Table 1.1 : Port Mode Control Summary.

Mode 0 (unidirectional 8-bit mode) Port A Submode 00 - Pin-definable Double-buffered Input or Single-buffered Output H1 - Latches Input Data H2 - Status/interrupt Generating Input, General-purpose Output, or Operation with H1 in the Interlocked or Pulsed Handshake Protocols Submode 01 - Pin-definable Double-buffered Output or Non-latched Input H1 - Indicates Data Received by Peripheral H2 - Status/interrupt Generating Input, General-purpose Output, or Operation with H1 in the Interlocked or Pulsed Handshake Protocols Submode 1X - Pin-definable Single-buffered Output or non-latched Input H1 - Status/interrupt Generating Input H2 - Status/interrupt Generating Input or General-purpose Output Port B H3 and H4 - Identical to Port A, H1 and H2
Mode 1 (unidirectional 16-bit mode) Port A - Most-significant Data Byte or non-latched Input or Single-buffered Output Submode XX - (not used) H1 - Status/interrupt Generating Input H2 - Status/interrupt Generating Input or General-purpose Output Port B - Least-significant Data Byte Submode X0 - Pin-definable Double-buffered Input or Single-buffered Output H3 - Latches Input Data H4 - Status/interrupt Generating Input, General-purpose Output, or Operation with H3 in the Interlocked or pulsed handshake Protocols Submode X1 - Pin-definable Double-buffered Output or Non-latched Input H3 - Indicates Data Received by Peripheral H4 - Status/interrupt Generating Input, General-purpose Output, or Operation with H3 in the Interlocked or Pulsed Hanshake Protocols
Mode 2 (bidirectional 8-bit mode) Port A - Bit I/O Submode XX - (not used) Port B - Double-buffered Bidirectional Data Submode XX - (not used) H1 - Indicates Output Data Received by the Peripheral and Controls Output Drivers H2 - Operation with H1 in the Interlocked or Pulsed Output Handshake Protocols H3 - Latches Input Data H4 - Operation with H3 in the Interlocked or Pulsed Input Handshake Protocols
Mode 3 (bidirectional 16-bit mode) Port A - Double-buffered Bidirectional Data (most-signifiant data byte) Submode XX - (not used) Port B - Double-buffered Bidirectional Data (least-signifiant data byte) Submode XX - (not used) H1 - Indicates Output Data Received by the Peripheral and Controls Output Drivers H2 - Operation with H1 in the Interlocked or Pulsed Output Handshake Protocols H3 - Latches Input Data H4 - Operation with H3 in the Interlocked or Pulsed Input Handshake Protocols

Figure 1.2 : Port Mode Layout.

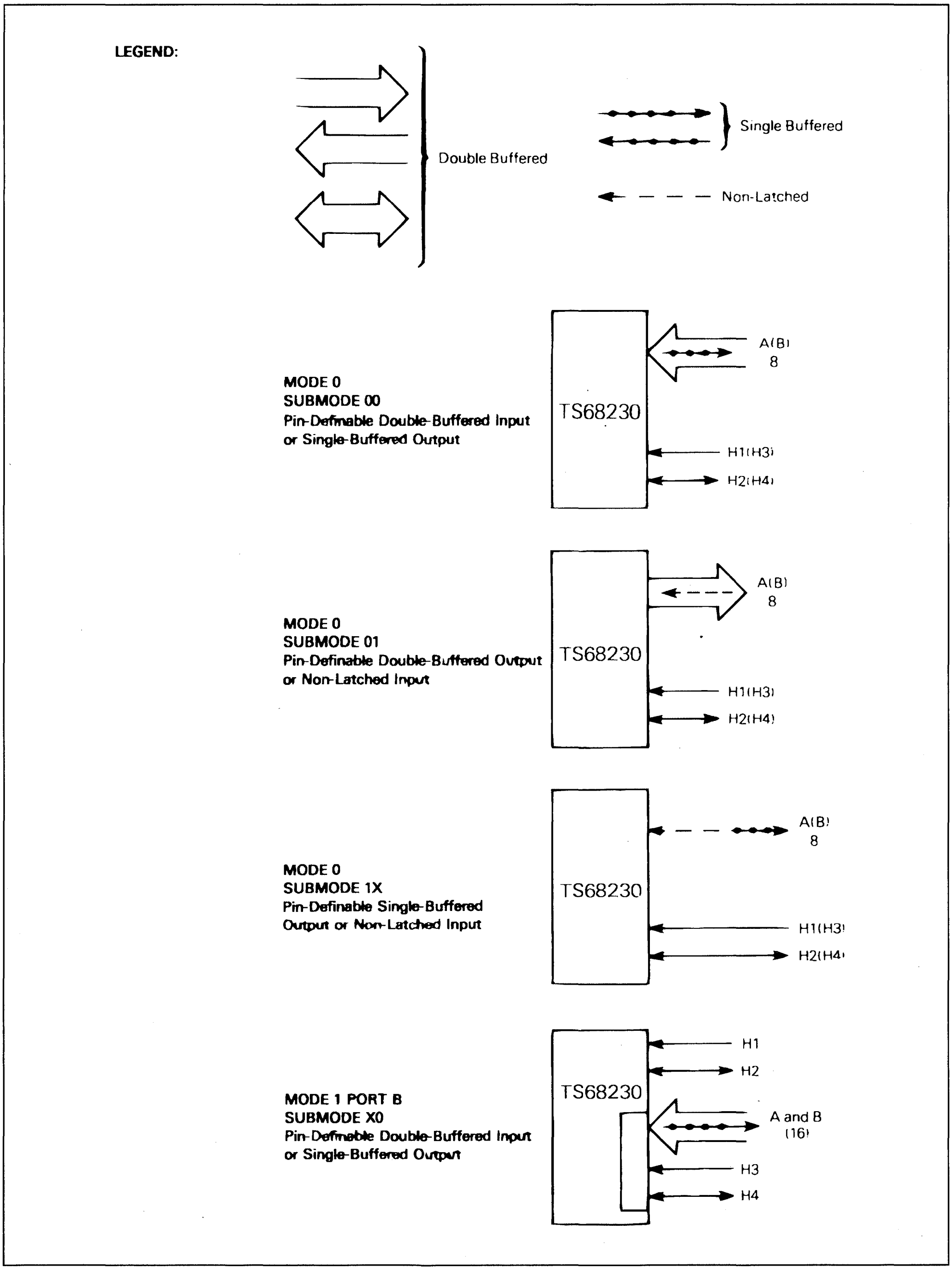
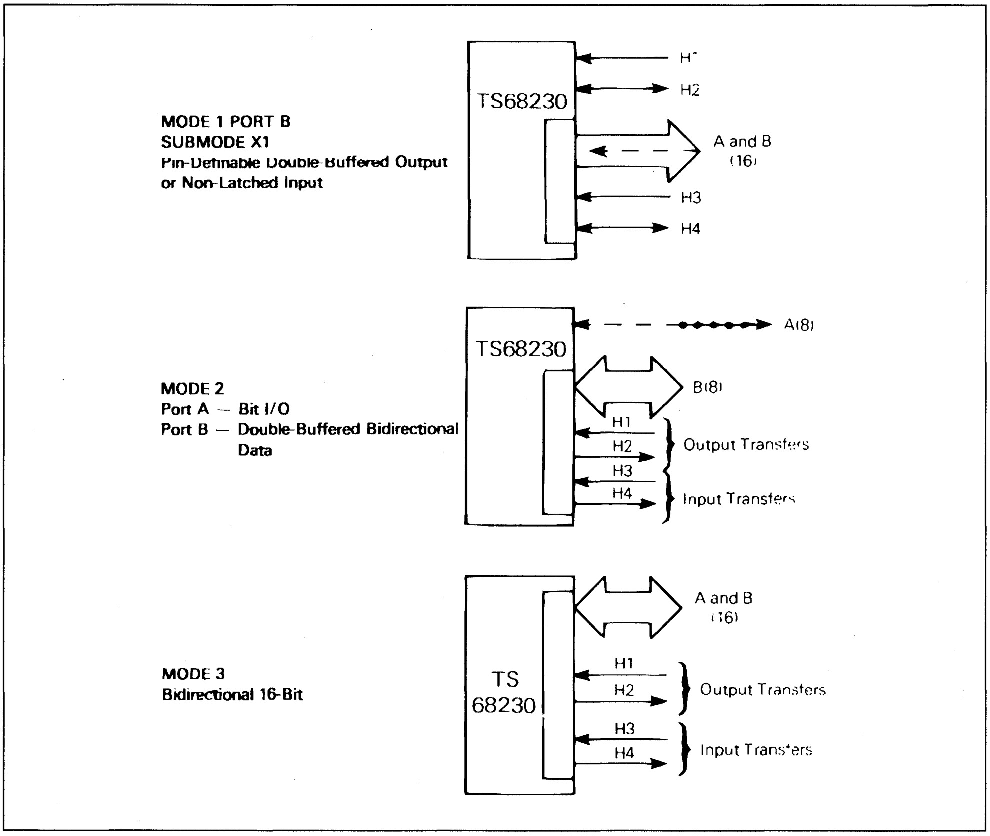

LEGEND:
Double Buffered
Single Buffered
Non-Latched
MODE 0
SUBMODE 00
Pin-Definable Double-Buffered Input or Single-Buffered Output
TS68230
A(B) 8
H1(H3)
H2(H4)
MODE 0
SUBMODE 01
Pin-Definable Double-Buffered Output or Non-Latched Input
TS68230
A(B) 8
H1(H3)
H2(H4)
MODE 0
SUBMODE 1X
Pin-Definable Single-Buffered Output or Non-Latched Input
TS68230
A(B) 8
H1(H3)
H2(H4)
MODE 1 PORT B
SUBMODE X0
Pin-Definable Double-Buffered Input or Single-Buffered Output
TS68230
H1
H2
A and B (16)
H3
H4

Figure 1.2 : Port Mode Layout (continued).



1.2. SIGNAL DESCRIPTION

Throughout this data sheet, signals are presented using the terms active and inactive or asserted and negated independent of whether the signal is active in the high-voltage state or low-voltage state. (The active state of each logic pin is given below). Active low signals are denoted by a superscript bar. R/$\overline{W}$ indicates a write is active low and a read active high. Table 1.2 further describes each pin and the logical pin assignments are given in figure 1.3.

1.2.1. BIDIRECTIONAL DATA BUS (D0-D7). The data bus pins D0-D7 form an 8-bit bidirectional data bus to/from an TS68000 bus master. These pins are active high.

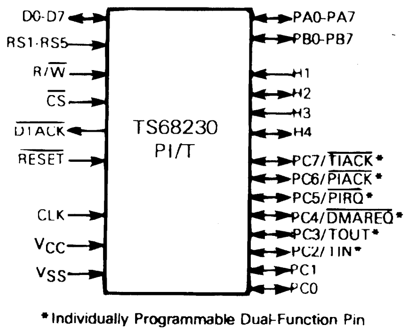
1.2.2. REGISTER SELECTS (RS1-RS5). The register select pins, RS1-RS5, are active high high-impedance inputs that determine which of the 23 internal registers is being selected. They are provided by the TS68000 bus master or other bus master.

Table 1.2 : Signal Summary.

Signal Name	Input/Output	Active State	Edge/Level Sensitive	Output States
CLK	Input		Falling and Rising Edge	
$\overline{\text{CS}}$	Input	Low	Level	
D0-D7	Input/output	High = 1, Low = 0	Level	High, Low, High Impedance
$\overline{\text{DMAREQ}}$	Output	Low		High, Low
$\overline{\text{DTACK}}$	Output	Low		High, Low, High Impedance*
H1(H3)***	Input	Low or High	Asserted Edge	
H2(H4)**	Input or Output	Low or High	Asserted Edge	High, Low, High Impedance
PA0-PA7**, PB0-PB7**, PC0-PC7	Input/output, Input or Output	High = 1, Low = 0	Level	High, Low, High Impedance
$\overline{\text{PIACK}}$	Input	Low	Level	
$\overline{\text{PIRQ}}$	Output	Low		Low, High Impedance*
RS1-RS5	Input	High = 1, Low = 0	Level	
R/$\overline{\text{W}}$	Input	High Read, Low Write	Level	
$\overline{\text{RESET}}$	Input	Low	Level	
$\overline{\text{TIACK}}$	Input	Low	Level	
TIN (external clock)	Input		Rising Edge	
TIN (run/halt)	Input	High	Level	
TOUT (square wave)	Output	Low		High, Low
TOUT ($\overline{\text{TIRQ}}$)	Output	Low		Low, High Impedance*

* Pullup resistors required.
** Note these pins have internal pullup resistors.
*** H1 is level sensitive for output buffer control in modes 2 and 3.

Figure 1.3 : Logical Pin Connection.



1.2.3. READ/WRITE (R/$\overline{\text{W}}$). R/$\overline{\text{W}}$ is a high impedance read/write input signal from the TS68000 bus master, indicating whether the current bus cycle is a read (high) or write (low) cycle.

1.2.4. CHIP SELECT ($\overline{\text{CS}}$). $\overline{\text{CS}}$ is a high-impedance input that selects the PI/T registers for the current bus cycle. Address strobe and the data strobe (upper or lower) of the bus master, along with the appropriate address bits, must be included in the chip-select equation. A low level corresponds to an asserted chip select.

1.2.5. DATA TRANSFER ACKNOWLEDGE ($\overline{\text{DTACK}}$). $\overline{\text{DTACK}}$ is an active low output that signals the completion of the bus cycle. During read or interrupt acknowledge cycles, $\overline{\text{DTACK}}$ is asserted after data has been provided on the data bus ; during write cycles it is asserted after data has been accepted at the data bus. Data transfer acknowledge is compatible with the TS68000 and with other TS68000 bus masters such as the 68440 direct memory access controller (DMAC). A pullup resistor is required to maintain $\overline{\text{DTACK}}$ high between bus cycles.

1.2.6. RESET ($\overline{\text{RESET}}$). $\overline{\text{RESET}}$ is a high-impedance input used to initialize all PI/T functions. All control and data direction registers are cleared and most internal operations are disabled by the assertion of $\overline{\text{RESET}}$ (low).

1.2.7. CLOCK (CLK). The clock pin is a high-impedance TTL-compatible signal with the same specifications as the TS68000. The PI/T contains dynamic logic throughout, and hence this clock must not be gated off at any time. It is not necessary that this clock maintain any particular phase relationship with the TS68000 system clock. It may be connected to an independent frequency source (faster or slower) as long as all bus specifications are met.

1.2.8. PORT A AND PORT B (PA0-PA7 AND PB0-PB7). Ports A and B are 8-bit ports that may be concatenated to form a 16-bit port in certain modes. The ports may be controlled in conjunction with the handshake pins H1-H4. For stabilization during system power up, ports A and B have internal pullup resistors to V_{CC}. All ports pins are active high.

1.2.9. HANDSHAKE PINS (H1-H4). Handshake pins H1-H4 are multi-purpose pins that (depending on the operational mode) may provide an interlocked handshake, a pulsed handshake, an interrupt input (independent of data transfers), or simple I/O pins. For stabilization during system power up, H2 and H4 have internal pullup resistors to V_{CC}. The sense of H1-H4 (active high or low) may be programmed in the port general control register bits 3-0. Independent of the mode, the instantaneous level of the handshake pins can be read from the port status register.

1.2.10. PORT C (PC0-PC7/ALTERNATE FUNCTION). This port can be used as eight general purpose I/O pins (PC0-PC7) or any combination of six special function pins and two general purpose I/O pins (PC0-PC1). Each dual-function pin can be a standard I/O or a special function independent of the other port C pins. When used as a port C pin, these pins are active high. They may be individually programmed as inputs or outputs by the port C data direction register. The dual-function pins are defined in the following paragraphs.

The alternate functions TIN, TOUT, and $\overline{\text{TIACK}}$ are timer I/O pins. TIN may be used as a rising-edge triggered external clock input or an external run/halt control pin (the timer is in the run state if run/halt is high and in the halt state if run/halt is low). TOUT may provide an active low timer interrupt request output or a general-purpose square-wave output, initially high. TIACK is an active low high-impedance input used for timer interrupt acknowledge.

Port A and B functions have an independent pair of active low interrupt request (PIRQ) and interrupt acknowledge (PIACK) pins.

The $\overline{\text{DMAREQ}}$ (direct memory access request) pin provides an active low direct memory access controller request pulse for three clock cycles, completely compatible with the 68440 DMAC.

1.3. REGISTER MODEL

A register model that includes the corresponding register selects is shown in table 1.3.

Table 1.3 : Register Model.

Register Select Bits									**Register Value after $\overline{\text{RESET}}$** (hex value)	
5 4 3 2 1	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**		
0 0 0 0 0	Port Mode Control		H34 Enable	H12 Enable	H4 Sense	H3 Sense	H2 Sense	H1 Sense	0 0	Port General Control Register
0 0 0 0 1	*	SVCRQ Select		IPF Select		Port Interrupt Priority Control			0 0	Port Service Request Register
0 0 0 1 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	0 0	Port A Data Direction Register
0 0 0 1 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	0 0	Port B Data Direction Register
0 0 1 0 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	0 0	Port C Data Direction Register
0 0 1 0 1	Interrupt Vector Number						*	*	0 F	Port Interrupt Vector Register
0 0 1 1 0	Port A Submode		H2 Control			H2 Int Enable	H1 SVCRQ Enable	H1 Stat Ctrl	0 0	Port A Control Register
0 0 1 1 1	Port B Submode		H4 Control			H4 Int Enable	H3 SVCRQ Enable	H3 Stat Ctrl	0 0	Port B Control Register
0 1 0 0 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	**	Port A Data Register
0 1 0 0 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	**	Port B Data Register
0 1 0 1 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	***	Port A Alternate Register
0 1 0 1 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	***	Port B Alternate Register
0 1 1 0 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	****	Port C Data Register
0 1 1 0 1	H4 Level	H3 Level	H2 Level	H1 Level	H4S	H3S	H2S	H1S	****	Port Status Register
0 1 1 1 0	*	*	*	*	*	*	*	*	0 0	(null)
0 1 1 1 1	*	*	*	*	*	*	*	*	0 0	(null)

* Unused, read as zero.
** Value before $\overline{\text{RESET}}$.
*** Current value on pins.
**** Undetermined value.

Table 1.3 : Register Model (continued).

Register Select Bits 5 4 3 2 1	7	6	5	4	3	2	1	0	Register Value after RESET (hex value)	
1 0 0 0 0	TOUT/$\overline{\text{TIACK}}$ Control			Z D Ctrl	*	Clock Control		Timer Enable	0 0	Timer Control Register
1 0 0 0 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	0 F	Timer Interrupt Vector Register
1 0 0 1 0	*	*	*	*	*	*	*	*	0 0	(null)
1 0 0 1 1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	**	Counter Preload Register (high)
1 0 1 0 0	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	**	Counter Preload Register (mid)
1 0 1 0 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	**	Counter Preload Register (low)
1 0 1 1 0	*	*	*	*	*	*	*	*	0 0	(null)
1 0 1 1 1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	**	Count Register (high)
1 1 0 0 0	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	**	Count Register (mid)
1 1 0 0 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	**	Count Register (low)
1 1 0 1 0	*	*	*	*	*	*	*	ZDS	0 0	Timer Status Register
1 1 0 1 1	*	*	*	*	*	*	*	*	0 0	(null)
1 1 1 0 0	*	*	*	*	*	*	*	*	0 0	(null)
1 1 1 0 1	*	*	*	*	*	*	*	*	0 0	(null)
1 1 1 1 0	*	*	*	*	*	*	*	*	0 0	(null)
1 1 1 1 1	*	*	*	*	*	*	*	*	0 0	(null)

* Unused, read as zero.
** Value before RESET.

1.4. BUS INTERFACE OPERATION

The PI/T has an asynchronous bus interface primarily designed for use with an TS68000 microprocessor. With care, however, it can be connected to synchronous microprocessor buses. This section completely describes the PI/T's bus interface, and is intended for the asynchronous bus designer unless otherwise mentioned.

In an asynchronous system the PI/T clock may operate at a significantly different frequency, either higher or lower, than the bus master and other system components, as long as all bus specifications are met. The TS68230 CLK pin has the same specifications as the TS68000 CLK pin, and must not be gated off at any time.

The following signals generate normal read and write cycles to the PI/T : $\overline{CS}$ (chip select), R/$\overline{W}$ (read/write), RS1-RS5(five register select bits), D0-D7 (the 8-bit bidirectional data bus), and $\overline{DTACK}$ (data transfer acknowledge). To generate interrupt acknowledge cycles, PC6/$\overline{PIACK}$ or PC7/$\overline{TIACK}$ is used instead of $\overline{CS}$, and the register select pins are ignored. No combination of the following pin functions may be asserted simultaneously : $\overline{CS}$, $\overline{PIACK}$, or $\overline{TIACK}$.

1.4.1. READ CYCLES. This category includes all register reads, except port or timer interrupt acknowledge cycles. When $\overline{CS}$ is asserted, the register select and R/$\overline{W}$ inputs are latched internally. They must meet small setup and hold time requirements with respect to the asserted edge of $\overline{CS}$. (Refer to **6.6 AC Electrical Specifications** for further information). The PI/T is **not** protected against aborted (shortened) bus cycles generated by an address error or bus error exception in which it is addressed.

Certain operations triggered by normal read (or write) bus cycles are not complete within the time allotted to the bus cycle. One example is transfers to/from the double-buffered latches that occur as a result of the bus cycle. If the bus master's clock is significan-tly faster than the PI/T's the possibility exists that, following the bus cycle, $\overline{CS}$ can be negated then re-asserted before completion of these internal operations. In this situation the PI/T does not recognize the re-assertion of $\overline{CS}$ until these operations are complete. Only at that time does it begin the internal sequencing necessary to react to the asserted $\overline{CS}$. Since $\overline{CS}$ also controls the $\overline{DTACK}$ response, this "bus cycle recovery time" can be related to the clock edge on which $\overline{DTACK}$ is asserted for that cycle. The PI/T will recognize the subsequent assertion of $\overline{CS}$ three clock periods after the clock edge on which $\overline{DTACK}$ was previously asserted.

The register select and R/$\overline{W}$ inputs pass through an internal latch that is transparent when the PI/T can recognize a new $\overline{CS}$ pulse (see above paragraph). Since the internal data bus of the PI/T is continuously engaged for read transfers, the read access time (to the data bus buffers) begins when the register selects are stabilized internally. Also, when the PI/T is ready to begin a new bus cycle, the assertion of $\overline{CS}$ enables the data bus buffers within a short propagation delay. This does not contribute to the overall read access time unless $\overline{CS}$ is asserted significantly after the register select and R/$\overline{W}$ inputs are stabilized (as may occur with synchronous bus microprocessors).

In addition to the chip select's previously mentioned duties, it controls the assertion of $\overline{DTACK}$ and latching of read data at the data bus interface. Except for controlling input latches and enabling the data bus buffers, all of these functions occur only after $\overline{CS}$ has been recognized internally and synchronized with the internal clock. Chip select is recognized on the falling edge of the clock if the setup time is met ; $\overline{DTACK}$ is asserted (low) on the next falling edge of the clock. Read data is latched at the PI/T's data bus interface at the same time $\overline{DTACK}$ is asserted. It is stable as long as chip select remains asserted independent of other external conditions.

From the above discussion it is clear that if the chip select setup time prior to the falling edge of the clock is met, the PI/T can consistently respond to a new read or write bus cycle every four clock cycles. This fact is especially useful in designing the PI/T's clock in synchronous bus systems not using $\overline{DTACK}$. (An extra clock period is required in interrupt acknowledge cycles, see **1.4.2 Interrupt Acknowledge Cycles**).

In asynchronous bus systems in which the PI/T's clock differs from that of the bus master, generally there is no way to guarantee that the chip select setup time with respect to the PI/T clock is met. Thus, the only way to determine that the PI/T recognized the assertion of $\overline{CS}$ is to wait for the assertion of $\overline{DTACK}$. In this situation, all latched bus inputs to the PI/T must be held stable until $\overline{DTACK}$ is asserted. These include register select, R/$\overline{W}$, and write data inputs (see below).

System specifications impose a maximum delay from the trailing (negated) edge of $\overline{CS}$ to the negated edge of $\overline{DTACK}$. As system speeds increase this becomes more difficult to meet with a simple pullup resistor tied to the $\overline{DTACK}$ line. Therefore, the PI/T provides an internal active pullup device to reduce the rise time, and a level-sensitive circuit that later turns this device off. $\overline{DTACK}$ is negated asynchronously as fast as possible following the rising edge of chip select, then three-stated to avoid interference with the next bus cycle.

The system designer must take care that $\overline{\text{DTACK}}$ is negated and three-stated quickly enough after each bus cycle to avoid interference with the next one. With an TS68000 this necessitates a relatively fast external path from the data strobe negation to $\overline{\text{CS}}$ bus master negation.

1.4.2. INTERRUPT ACKNOWLEDGE CYCLES. Special internal operations take place on PI/T interrupt acknowledge cycles. The port interrupt vector register or the timer vector register are implicitly addressed by the assertion of PC6/PIACK or PC7/TIACK, respectively. The signals are first synchronized with the falling edge of the clock. One clock period after they are recognized, the data bus buffers are enabled and the vector is driven onto the bus. $\overline{\text{DTACK}}$ is asserted after another clock period to allow the vector some setup time prior to $\overline{\text{DTACK}}$. DTACK is negated, then three-stated, as with normal read or write cycles, when $\overline{\text{PIACK}}$ or $\overline{\text{TIACK}}$ is negated.

1.4.3. WRITE CYCLES. In many ways, write cycles are similar to normal read cycles. On write cycles, data at the D0-D7 pins must meet the same setup specifications as the register select and R/W lines. Like these signals, write data is latched on the asserted edge of CS, and must meet small setup and hold time requirements with respect to that edge. The same bus cycle recovery conditions exist as for normal read cycles. No other differences exist.

SECTION 2

PORT GENERAL INFORMATION AND CONVENTIONS

This section introduces concepts that are generally applicable to the PI/T ports independent of the chosen mode and submode. For this reason, no particular port or handshake pins are mentioned ; the notation H1(H3) indicates that, depending on the chosen mode and submode, the statement given may be true for either the H1 or H3 handshake pin.

2.1. UNIDIRECTIONAL VS BIDIRECTIONAL

Figure 1.2 shows the configuration of ports A and B and each of the handshake pins in each port mode and submode. In modes 0 and 1, a data direction register is associated with each of the ports. These registers contain one bit for each port pin to determine whether that pin is an input or an output. Modes 0 and 1 are, thus, called unidirectional modes because each pin assumes a constant direction, changeable only by a reset condition or a programming change. These modes allow double-buffered data transfers in one direction. This direction, determined by the mode and submode definition, is known as the primary direction. Data transfers in the primary direction are controlled by the handshake pins. Data transfers not in the primary direction are generally unrelated, and single or unbuffered data paths exist.

In modes 2 and 3 there is no concept of primary direction as in modes 0 and 1. Except for port A in mode 2 (bit I/O), the data direction registers have no effect. These modes are bidirectional, in that the direction of each transfer (always 8 or 16 bits, double buffered) is determined dynamically by the state of the handshake pins. Thus, for example, data may be transferred out of the ports, followed very shortly by a transfer into the same port pins. Transfers to and from the ports are independent and may occur in any sequence. Since the instantaneous direction is always determined by the external system, a small amount of arbitration logic may be required.

2.1.1. CONTROL OF DOUBLE-BUFFERED DATA PORTS. Generally speaking, the PI/T is a double-buffered device. In the primary direction, double buffering allows orderly transfers by using the handshake pins in any of several programmable protocols. (When bit I/O is used, double buffering is not available and the handshake pins are used as outputs or status/interrupt inputs).

Use of double buffering is most beneficial in situations where a peripheral device and the computer system are capable of transferring data at roughly the same speed. Double buffering allows the fetch operation of the data transmitter to be overlapped with the store operation of the data receiver. Thus, throughput measured in bytes or words-per-second may be greatly enhanced. If there is a large mismatch in transfer capability between the computer and the peripheral, little or no benefit is obtained. In these cases there is no penalty in using double buffering.

2.1.2. DOUBLE-BUFFERED INPUT TRANSFERS. In all modes, the PI/T supports double-buffered input transfers. Data that meets the port setup and hold times is latched on the asserted edge of H1(H3). H1(H3) is edge sensitive, and may assume any duty cycle as long as both high and low minimum times are observed. The PI/T contains a port status register whose H1S(H3S) status bit is set anytime any input data that has not been read by the bus master is present in the double-buffered latches. The action of H2(H4) is programmable ; it may indicate whether there is room for more data in the PI/T latches or it may serve other purposes. The following options are available, depending on the mode.

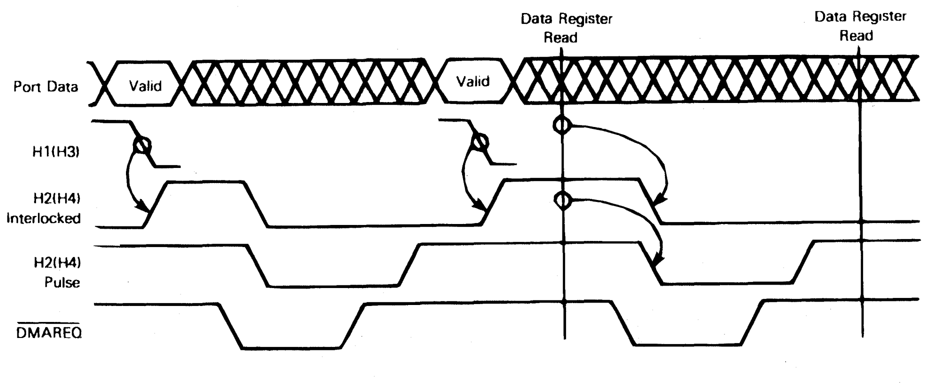
1. H2(H4) may be an edge-sensitive input that is independent of H1(H3) and the transfer of port data. On the asserted edge of H2(H4), the H2S(H4S) status bit is set. It is cleared by the direct method (refer to **2.3 Direct Method of Resetting Status**), the $\overline{\text{RESET}}$ pin being asserted, or when the H12 enable (H34 enable) bit of the port general control register is zero.
2. H2(H4) may be a general purpose output pin that is always negated. The H2S(H4S) status bit is always zero.
3. H2(H4) may be a general purpose output pin that is always asserted. The H2S(H4S) status bit is always zero.
4. H2(H4) may be an output pin in the interlocked input handshake protocol. It is asserted when the port input latches are ready to accept new data. It is negated asynchronously following the asserted edge of the H1(H3) input. As soon as the input latches become ready, H2(H4) is again asserted. When both double-buffered latches are full, H2(H4) remains negated until data is removed by a read of port A (port B) data register. Thus, anytime the H2(H4) output is asserted, new input data may be entered by asserting H1(H3). At other times transitions of H1(H3) are ignored. The H2S(H4S) status bit is always zero. When H12 enable (H34 enable) is zero, H2(H4) is held negated.

5. H2(H4) may be an output pin in the pulsed input handshake protocol. It is asserted exactly as in the interlocked input protocol, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case that a subsequent H1(H3) asserted edge occurs before termination of the pulse, H2(H4) is negated asynchronously. Thus, anytime after the leading edge of the H2(H4) pulse, new data may be entered in the PI/T double-buffered input latches. The H2S(H4S) status bit is always zero. When H12 enable (H34 enable) is zero, H2(H4) is held negated.

2.1.3. DOUBLE-BUFFERED OUTPUT TRANSFERS. The PI/T supports double-buffered output transfers in all modes. Data, written by the bus master to the PI/T, is stored in the port's output latch. The peripheral accepts the data by asserting H1(H3), which causes the next data to be moved to the port's output latch as soon as it is available. The function of H2(H4) is programmable ; it may indicate whether data has been moved to the output latch or it may serve other purposes. The H1S(H3S) status bit may be programmed for two interpretations. First, the status bit is a one when there is at least one latch in the double-buffered data path that can accept new data. After writing one byte/word of data to the ports, an interrupt service routine could check this bit to determine if it could store another byte/word, thus filling both latches. Second, when the bus master is finished, it is often useful to be able to check whether all of the data has been transferred to the peripheral. The H1S(H3S) status bit is set when both output latches are empty. The programmable options of the H2(H4) pin are given below, depending on the mode.

1. H2(H4) may be an edge-sensitive input pin independent of H1(H3) and the transfer of port data. On the asserted edge of H2(H4), the H2S(H4S) status bit is set. It is cleared by the direct method (refer to **2.3 Direct Method of Resetting Status**), the $\overline{\text{RESET}}$ pin being asserted, or when the H12 enable (H34 enable) bit of the port general control register is zero.
2. H2(H4) may be a general-purpose output pin that is always zero.
3. H2(H4) may be a general-purpose output pin that is always asserted. The H2S(H4s) status bit is always zero.
4. H2(H4) may be an output pin in the interlocked output handshake protocol. H2(H4) is asserted two clock cycles after data is transferred to the double-buffered output latches. The data remains stable and H2(H4) remains asserted until the next asserted edge of the H1(H3) input. At that time, H2(H4) is asynchronously negated. As soon as the next data is available, it is transferred to the output latches and H2(H4) is asserted. When H2(H4) is negated, asserted transitions on H1(H3) have no effect on the data paths. As is explained later, however, in modes 2 and 3 H1 does control the three-state output buffers of the bidirectional port(s). The H2S(H4S) status bit is always zero. When H12 enable (H34 enable) is zero, H2(H4) is held negated.

Figure 2.1 : Double-Buffered Input Transfers Timing Diagram.



5. H2(H4) may be an output pin in the pulsed output handshake protocol. It is asserted exactly as in the interlocked output protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock pulse is generated. But in the case that a subsequent H1(H3) asserted edge occurs before termination of the pulse, H2(H4) is negated asynchronously, thus shortening the pulse. The H2S(H4S) status bit is always zero. When H12 enable (H34 enable) is zero, H2(H4) is held negated.

A sample timing diagram is shown in figure 2.2. The H2(H4) interlocked and pulsed output handshake protocols are shown. The DMAREQ pin is also shown assuming it is enabled. All handshake pin sense bits are assumed to be zero ; thus, the pins are in the low state when asserted. Due to the great similarity between modes, this timing diagram is applicable to all double-buffered output transfers.

2.2. REQUESTING BUS MASTER SERVICE

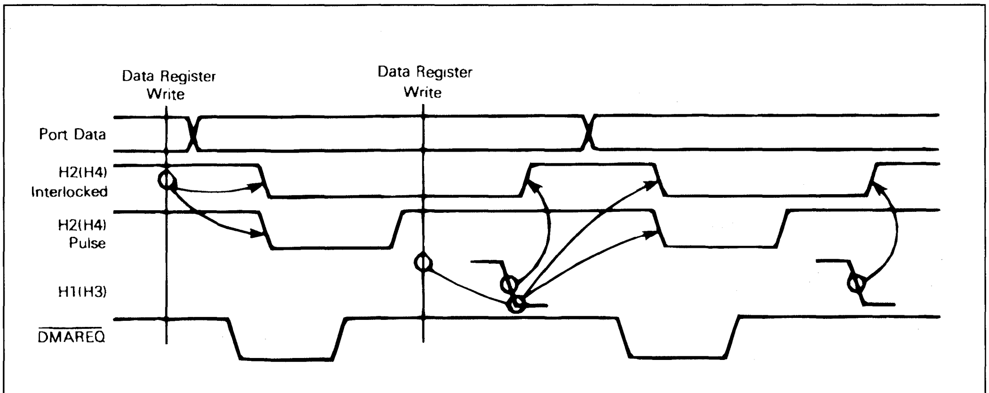
The PI/T has several means of indicating a need for service by a bus master. First, the processor may poll the port status register. It contains a status bit for each handshake pin, plus a level bit that always reflects the instantaneous state of that handshake pin. A status bit is one when the PI/T needs servicing (i.e., generally when the bus master needs to read or write data to the ports) or when a handshake pin used as a simple status input has been asserted. The interpretation of these bits is dependent on the chosen mode and submode.

Second, the PI/T may be placed in the processor's interrupt structure. As mentioned previously, the PI/T contains port A and B control registers that configure the handshake pins. Other bits in these registers enable an interrupt associated with each handshake pin. This interrupt is made available through the PC5/PIRQ pin, if the PIRQ function is selected. Three additional conditions are required for PIRQ to be asserted : 1) the handshake pin status bit is set, 2) the corresponding interrupt (service request) enable bit is set, and 3) DMA requests are not associated with that data transfer (H1 and H3 only). The conditions from each of the four handshake status bits and corresponding status bits are ORed to determine PIRQ. To clear the interrupt, the proper status bit must be cleared (see **2.3. Direct Method of Resetting Status**).

The third method of requesting service is via the PC4/DMAREQ pin. This pin can be associated with double-buffered transfers in each mode. If it is used as a DMA controller request, it can initiate requests to keep the PI/T's input/output double-buffering empty/full as much as possible. It will not overrun the DMA controller. The pin is compatible with the 68440 direct memory access controller (DMAC).

2.2.1. VECTORED, PRIORITIZED PORT INTERRUPTS. Use of TS68000 compatible vectored interrupts with the PI/T requires the PIRQ and PIACK pins. When PIACK is asserted while PIRQ is asserted, the PI/T places an 8-bit vector on the data pins D0-D7. Under normal conditions, this vector corresponds to the highest priority enabled active port interrupt source with which the DMAREQ pin is not currently associated. The most-significant six bits are provided by the port interrupt vector register (PIVR), with the lower two bits supplied by prioritization logic according to conditions present when PIACK is asserted. It is important to note that the

Figure 2.2 : Double-Buffered Output Transfers Timing Diagram.



only effect on the PI/T caused by interrupt acknowledge cycles is that the vector is placed on the data bus. Specifically, no registers, data, status, or other internal states of the PI/T are affected by the cycle.

Several conditions may be present when the PIACK input is asserted to the PI/T. These conditions affect the PI/T's response and the termination of the bus cycle. If the PI/T has no interrupt function selected, or is not asserting PIRQ, the PI/T will make no response to PIACK (DTACK will not be asserted). If the PI/T is asserting PIRQ when PIACK is received, the PI/T will output the contents of the port interrupt vector register and the prioritization bits. If the PIVR has not been initialized, $0F will be read from this register. These conditions are summarized in table 2.1.

The vector table entries for the PI/T appear as a contiguous block of four vector numbers whose common upper six bits are programmed in the PIVR. The following table pairs each interrupt source with the 2-bit value provided by the prioritization logic when interrupt acknowledge is asserted (see **4.2. Port Service Request Register** (PSRR)).

H1 source - 00	H2 source - 01
H3 source - 10	H4 source - 11

2.2.2. AUTOVECTORED PORT INTERRUPTS. Autovectored interrupts use only the PIRQ pin. The operation of the PI/T with vectored and autovectored interrupts is identical except that no vectors are supplied and the PC6/PIACK pin can be used as a port C pin.

2.2.3. DMA REQUEST OPERATION. The direct memory access request (DMAREQ) pulse (when enabled) is associated with output or input transfers to keep the initial and final output latches full or initial and final input latches empty, respectively. Figures 2.3 and 2.4 show all the possible paths in generating DMA requests. See **4.2. Port Service Request Register** (PSRR) for programming the operation of the DMA request bit.

DMAREQ is generated on the bus side of the TS68230 by the synchronized* chip select. If the conditions of figures 2.3 or 2.4 are met, an assertion of CS will cause DMAREQ to be asserted three PI/T clocks (plus the delay time from the clock edge) after CS is synchronized. DMAREQ remains asserted three clock cycles (plus the delay time from the clock edge) and is then negated.

DMAREQ pulses are associated with peripheral transfers or are generated by the synchronized* H1(H3) input. If the conditions of figures 2.3 or 2.4 are met, an assertion of the H1(H3) input will cause DMAREQ to be asserted 2.5 PI/T clock cycles (plus the delay time from clock edge) after H1(H3) is synchronized. DMAREQ remains asserted three clock cycles (plus the delay time from the clock edge) and is then negated.

Figure 2.3 : DMAREQ Associated with Output Transfers.

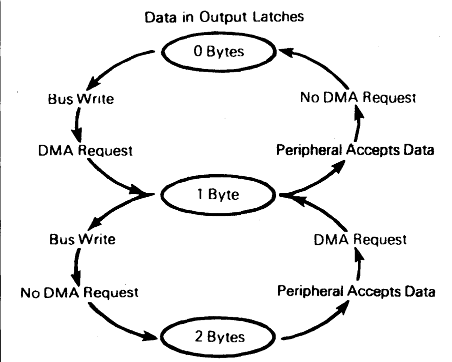
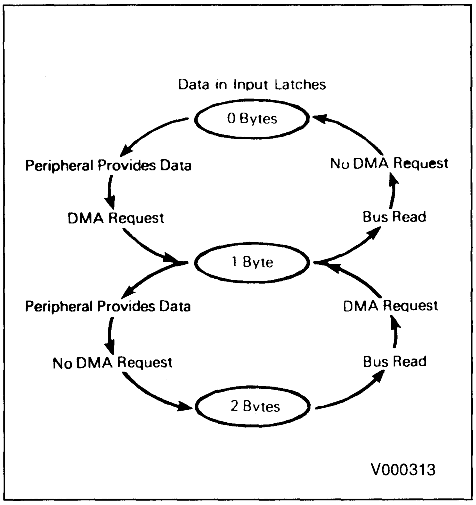


Table 2.1 : Response to Port Interrupt Acknowledge.

Conditions	PIRQ Negated OR Interrupt Request Function not Selected	PIRQ Asserted
PIVR has not been initialized since RESET.	No Response from PI/T. No DTACK.	PI/T provides $0F, the Uninitialized Vector*.
PIVR has been initialized since RESET.	No Response from PI/T. No DTACK.	PI/T provides PIVR contents with prioritization bits.

* The uninitialized vector is the value returned from an interrupt vector register before it has been initialized.
* Synchronized means that the appropriate input signal (H1, H3, or CS) has been sampled by the PI/T on the appropriatre edge of the clock (rising edge for H1(H3) and falling edge for CS). Refer to 1.4 BUS INTERFACE OPERATION for the exception concerning CS. If a bus access (assertion of CS) and a port access (assertion of H1(H3)) occur at the same time, CS will be recognized without any delay. H1(H3) will be recognized one clock cycle later.

Figure 2.4 : $\overline{\text{DMAREQ}}$ Associated with Input Transfers.



2.3. DIRECT METHOD OF RESETTING STATUS

In certain modes one or more handshake pins can be used as edge-sensitive inputs for the sole purpose of setting bits in the port status register. These bits consist of simple flip-flops. They are set (to one) by the occurrence of the asserted edge of the handshake pin input. Resetting a handshake status bit can be done by writing an 8-bit mask to the port status register. This is called the direct method of resetting. To reset a status bit that is resettable by the direct method, the mask must contain a one in the bit position of the port status register corresponding to the desired status bit. For status bits that are not resettable by the direct method in the chosen mode, the data written to the port status register has no effect. For status bits that are resettable by the direct method in the chosen mode, a zero in the mask has no effect.

2.4. HANDSHAKE PIN SENSE CONTROL

The PI/T contains exclusive-OR gates to control the sense of each of the handshake pins, whether used as inputs or outputs. Four bits in the port general control register may be programmed to determine whether the pins are asserted in the low- or high-voltage state. As with other control registers, these bits are reset to zero when the $\overline{\text{RESET}}$ pin is asserted, defaulting the asserted level to be low.

2.5. ENABLING PORTS A AND B

Certain functions involved with double-buffered data transfers, the handshake pins, and the status bits may be disabled by the external system or by the programmer during initialization. The port general control register contains two bits, H12 enable and H34 enable, which control these functions. These bits are cleared to the zero state when the $\overline{\text{RESET}}$ pin is asserted, and the functions are disabled. The functions are the following :

1. Independent of other actions by the bus master or peripheral (via the handshake pins), the PI/T's disabled handshake controller is held to the "empty" state ; i.e., no data is present in the double-buffered data path.
2. When any handshake pin is used to set a simple status flip-flop, unrelated to double-buffered transfers, these flip-flops are held reset to zero (see table 1.1).
3. When H2(H4) is used in an interlocked or pulsed handshake with H1(H3), H2(H4) is held negated, regardless of the chosen mode, submode, and primary direction. Thus, for double-buffered input transfers, the programmer may signal a peripheral when the PI/T is ready to begin transfers by setting the associated handshake enable bit to one.

2.6. PORT A AND B ALTERNATE REGISTERS

In addition to the port A and B data registers, the PI/T contains port A and B alternate registers. These registers are read only, and simply provide the instantaneous (non-latched) level of each port pin. They have no effect on the operation of the handshake pins, double-buffered transfers, status bits, or any other aspect of the PI/T, and they are mode/submode independent. Refer to **4.7. Port Alternate Registers** for further information.

SECTION 3

PORT MODES

This section contains information that distinguishes the various port modes and submodes. General characteristics common to all modes are defined in **Section 2 Port General Information and Conventions.** A description of the port A control register (PACR) and port B control register (PBCR) is given before each mode description. After each submode description, the programmable options are listed for that submode.

3.1. PORT A CONTROL REGISTER (PACR)

7 6	5 4 3	2	1	0
Port A Submode	H2 Control	H2 Interrupt Enable	H1 SVCRQ Enable	H1 Status Control

The port A control register, in conjunction with the programmed mode and the port B submode, controls the operation of port A and the handshake pins H1 and H2. The port A control register contains five fields : bits 7 and 6 specify the port A submode ; bits 5, 4, and 3 control the operation of the H2 handshake pin and the H2S status bit ; bit 2 determines whether an interrupt will be generated when the H2S status bit goes to one ; and bit 1 determines whether a service request (interrupt request or DMA request) will occur ; bit 0 controls the operation of the H1S status bit. The PACR is always readable and writable.

All bits are cleared to zero when the $\overline{\text{RESET}}$ pin is asserted. When the port A submode field is relevant in a mode/submode definition, it must not be altered unless the H12 enable bit in the port general control register is clear (see table 1.3 located at the end of this document). Altering these bits will give unpredictable results.

3.2. PORT B CONTROL REGISTER (PBCR)

7 6	5 4 3	2	1	0
Port B Submode	H4 Control	H4 Interrupt Enable	H3 SVCRQ Enable	H3 Status Control

The port B control register specifies the operation of port B and the handshake pins H3 and H4. The port B control register contains five fields : bits 7 and 6 specify the port B submode ; bits 5, 4, and 3 control the operation of the H4 handshake pin and H4S status bit ; bit 2 determines whether an interrupt will be generated when the H4S status bit goes to a one ; bit 1 determines whether a service request (interrupt request or DMA request) will occur ; and bit 0 controls the operation of the H3S status bit. The PBCR is always readable and writable. There is never a consequence to reading the register.

All bits are cleared to zero when the $\overline{\text{RESET}}$ pin is asserted. When the port B submode field is relevant in a mode/submode definition, it must not be altered unless the H34 enable bit in the port general control register is clear (see table 1.3 located at the end of this document).

3.3. MODE 0 - UNIDIRECTIONAL 8-BIT MODE

In mode 0, ports A and B operate independently. Each may be configured in any of its three possible submodes :

Submode 00 - Pin-Definable Double-Buffered Input or Single-Buffered Output

Submode 01 - Pin-Definable Double-Buffered Output or Non-Latched Input

Submode 1X - Bit I/O (Pin-Definable Single-Buffered Output or Non-Latched Input)

Handshake pins H1 and H2 are associated with port A and configured by programming the port A control register. (The H12 enable bit of the port general control register enables port A transfers). Handshake pins H3 and H4 are associated with port B and configured by programming the port B control register. (The H34 enable bit of the port general control register enables port B transfers). The port A and B data direction registers operate in all three submodes. Along with the submode, they affect the data read and write at the associated data register according to table 3.1. They also enable the output buffer associated with each port pin. The $\overline{\text{DMAREQ}}$ pin may be associated with either (not both) port A or port B, but does not function if the bit I/O submode (submode 1X) is programmed for the chosen port.

Table 3.1 : Mode 0 Port Data Paths.

Mode	Read Port A/B Data Register		Write Port A/B Data Register	
	DDR = 0	DDR = 1	DDR = X	
0 Submode 00	FIL, D. B.	FOL Note 3	FOL, S. B.	Note 1
0 Submode 01	Pin	FOL Note 3	IOL/FOL, D. B.	Note 2
0 Submode 1X	Pin	FOL Note 3	FOL, S. B.	Note 1

Abbreviations :

IOL - Initial Output Latch	S. B. - Single Buffered
FOL - Final Output Latch	D. B. - Double Buffered
FIL - Final Input Latch	DDR - Data Direction Register

Note 1 : Data is latched in the output data registers (final output latch) and will be single buffered at the pin if the DDR is 1. The output buffers will be turned off if the DDR is 0.
Note 2 : Data is latched in the double-buffered output data registers. The data in the final output latch will appear on the port pin if the DDR is a 1.
Note 3 : The output drivers that connect the final output latch to the pins are turned on

3.3.1. SUBMODE 00 - PIN-DEFINABLE DOUBLE-BUFFERED INPUT OR SINGLE-BUFFERED OUTPUT. In mode 0, double-buffered input transfers of up to eight bits are available by programming submode 00 in the desired port's control register. Data that meets the port setup and hold times is latched on the asserted edge of H1(H3) and is placed in the initial or final input latch. H1(H3) is edge sensitive and may assume any duty cycle as long as both high and low minimum times are observed. The PI/T contains a port status register whose H1S(H3S) status bit is set anytime any input data that has not been read by the bus master is present in the double-buffered latches. The action of H2(H4) is programmable. The following options are available :

1. H2(H4) may be an edge-sensitive status input that is independent of H1(H3) and the transfer of port data. On the asserted edge of H2(H4), the H2S(H4S) status bit is set. It is cleared by either the RESET pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H12 enable (H34 enable) bit of the port general register is clear.
2. H2(H4) may be a general-purpose output pin that is always negated. In this case the H2S(H4S) status bit is always clear.
3. H2(H4) may be a general-purpose output pin that is always asserted. In this case the H2S(H4S) status bit is always clear.
4. H2(H4) may be an output pin in the interlocked input handshake protocol. It is asserted when the port input latches are ready to accept new data. It is negated asynchronously following the asserted edge of the H1(H3) input. As soon as the input latches become ready, H2(H4) is again asserted. When the input double-buffered latches are full, H2(H4) remains negated until data is removed. Thus, anytime the H2(H4) output is asserted, new input data may be entered by asserting H1(H3). At other times, transitions on H1(H3) are ignored. The H2S(H4S) status bit is always clear. When H12 enable (H34 enable) in the port general control register is clear, H2(H4) is held negated.
5. H2(H4) may be an output pin in the pulsed input handshake protocol. It is asserted exactly as in the interlocked input protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case of a subsequent H1(H3) asserted edge occurring before termination of the pulse, H2(H4) is negated asynchronously. Thus, anytime after the leading edge of the H2(H4) pulse, new data may be entered in the double-buffered input latches. The H2S(H4S) status bit is always clear. When H12 enable (H34 enable) is clear, H2(H4) is held negated.

For pins used as outputs, the data path consists of a single latch driving the output buffer. Data written to the port's data register does not affect the operation of any handshake pin or status bit. Output pins may be used independently of the input transfers. However, read bus cycles to the data register do remove data from the port. Therefore, care should be taken to avoid processor instructions that perform unwanted read cycles.

Programmable Options Mode 0 - Port A Submode 00 and Port B Submode 00

PACR

7 6	**Port A Submode**
0 0	Submode 00

PACR

5 4 3	**H2 Control**
0 X X	Input pin - edge-sensitive status input, H2S is set on an asserted edge.
1 0 0	Output pin - negated, H2S is always clear.
1 0 1	Output pin - asserted, H2S is always clear.
1 1 0	Output pin - interlocked input handshake protocol, H2S is always clear.
1 1 1	Output pin - pulsed input handshake protocol, H2S is always clear.

PACR

2	**H2 Interrupt Enable**
0	The H2 interrupt is disabled.
1	The H2 interrupt is enabled.

PACR

1	**H1 SVCR Enable**
0	The H1 interrupt and DMA request are disabled.
1	The H1 interrupt and DMA request are enabled.

PACR

0	**H1 Status Control**
X	The H1S status bit is set anytime input data is present in the double-buffered input path.

PBCR

7 6	**Port B Submode**
0 0	Submode 00

PBCR

5 4 3	**H4 Control**
0 X X	Input pin - edge-sensitive status input, H4S is set on an asserted edge.
1 0 0	Output pin - negated, H4S is always cleared.
1 0 1	Output pin - asserted, H4S is always cleared.
1 1 0	Output pin - interlocked input handshake protocol, H4S is always cleared.
1 1 1	Output pin - pulsed input handshake protocol, H4S is always cleared.

Programmable Options Mode 0 - Port A Submode 00 and Port B Submode 00 (continued)

PBCR

2	**H4 Interrupt Enable**
0	The H4 interrupt is disabled.
1	The H4 interrupt is enabled.

PBCR

1	**H3 SVCRQ Enable**
0	The H3 interrupt and DMA request are disabled.
1	The H3 interrupt and DMA request are enabled.

PBCR

0	**H3 Status Control**
X	The H3S status bit is set anytime input data is present in the double-buffered input path.

3.3.2. SUBMODE 01 - PIN-DEFINABLE DOUBLE-BUFFERED OUTPUT OR NON-LATCHED INPUT. In mode 0, double-buffered output transfers of up to eight bits are available by programming submode 01 in the desired port's control register. The operation of H2 and H4 may be selected by programming the port A and B control registers, respectively. Data, written by the bus master to the PI/T, is stored in the port's output latches. The peripheral accepts the data by asserting H1(H3), which causes the next data to be moved to the port's output latch as soon as it is available.

The H1S(H3S) status bit may be programmed for two interpretations :

1. The H1S(H3S) status bit is set when either the port initial or final output latch can accept new data. It is cleared when both latches are full and cannot accept new data.
2. The H1S(H3S) status bit is set when both of the port output latches are empty. It is cleared when at least one latch is full.

The programmable options of the H2(H4) pin are :

1. H2(H4) may be an edge-sensitive input pin independent of H1(H3) and the transfer of port data. On the asserted edge of H2(H4), the H2S(H4S) status bit is set. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H1(H2) enable (H3(H4) enable) bit of the port general control register is clear.
2. H2(H4) may be a general-purpose output pin that is always negated. The H2S(H4S) status bit is always clear.
3. H2(H4) may be a general-purpose output pin that is always asserted. The H2S(H4S) status bit is always clear.
4. H2(H4) may be an output pin in the interlocked output handshake protocol. H2(H4) is asserted two clock cycles after data is transferred to the double-buffered output latches. The data remains stable at the port pins and H2(H4) remains asserted until the next asserted edge of the H1(H3) input. At that time, H2(H4) is asynchronously negated. As soon as the next data is available, it is transferred to the output latches. When H2(H4) is negated, asserted transitions of H1(H3) have no affect on data paths. The H2S(H4S) status bit is always clear. When H12 enable (H34 enable) is clear, H2(H4) is held negated.
5. H2(H4) may be an output pin in the pulsed output handshake protocol. It is asserted exactly as in the interlocked protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock pulse is generated. But in the case that a subsequent H1(H3) asserted edge occurs before termination of the pulse, H2(H4) is negated asynchronously shortening the pulse. The H3S(H4S) status bit is always clear. When H12 enable (H34 enable) is clear H2(H4) is held negated.

For pins used as inputs, data written to the associated data register is double-buffered and passed to the initial or final output latch, but, the output buffer is disabled.

Programmable Options Mode 0 - Port A Submode 01 and Port B Submode 01

PACR

7 6	**Port A Submode**
0 1	Submode 01

PACR

5 4 3	**H2 Control**
0 X X	Input pin - edge-sensitive status inputs, H2S is set on an asserted edge.
1 0 0	Output pin - negated, H2S is always clear.
1 0 1	Output pin - asserted, H2S is always clear.
1 1 0	Output pin - interlocked input handshake protocol, H2S is always clear.
1 1 1	Output pin - pulsed input handshake protocol, H2S is always clear.

PACR

2	**H2 Interrupt Enable**
0	The H2 interrupt is disabled.
1	The H2 interrupt is enabled.

PACR

1	**H1 SVCRQ Enable**
0	The H1 interrupt and DMA request are disabled.
1	The H1 interrupt and DMA request are enabled.

PACR

0	**H1 Status Control**
0	The H1S status bit is set when either the port A initial or final output latch can accept new data It is clear when both latches are full and cannot accept new data.
1	The H1S status bit is one when both of the port A output latches are empty. It is clear when at least one latch is full.

PBCR

7 6	**Port B Submode**
0 1	Submode 01

PBCR

5 4 3	**H4 Control**
0 X X	Input pin - edge-sensitive status input, H4S is set on an asserted edge.
1 0 0	Output pin - negated, H4S is always cleared.
1 0 1	Output pin - asserted, H4S is always cleared.
1 1 0	Output pin - interlocked input handshake protocol, H4S is always cleared.
1 1 1	Output pin - pulsed input handshake protocol, H4S is always cleared.

Programmable Options Mode 0 - Port A Submode 01 and Port B Submode 01 (continued)

PBCR	
2	**H4 Interrupt Enable**
0	The H4 interrupt is disabled.
1	The H4 interrupt is enabled.

PBCR	
1	**H3 SVCRQ Enable**
0	The H3 interrupt and DMA request are disabled.
1	The H3 interrupt and DMA request are enabled.

PBCR	
0	**H3 Status Control**
0	The H3S status bit is set when either the port B initial or final output latch can accept new data. It is clear when both latches are full and cannot accept new data.
1	The H3S status bit is one when both of the port B output latches are empty. It is clear when at least one latch is full.

3.3.3. SUBMODE 1X - BIT I/O (PIN-DEFINABLE SINGLE-BUFFERED OUTPUT OR NON-LATCHED INPUT). In mode 0, simple bit I/O is available by programming submode 1X in the desired port's control register. This submode is intended for applications in which several independent devices must be controlled or monitored. Data written to the associated (input/output) register is single buffered. If the data direction register bit for that pin is a one (output), the output buffer is enabled. If it is a zero (input) data written is still latched, but is not available at the pin. Data read from the data register is the instantaneous value of the pin or what was written to the data register, depending on the contents of the data direction register. H1(H3) is an edge-sensitive status input pin only and it controls no data related function. The H1S(H3S) status bit is set following the asserted edge of the input waveform. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the associated status bit in the port status register (PSR), or when the H12 enable (H34 enable) bit of the port general control register is clear. H2 may be programmed as :

1. H2(H4) may be an edge-sensitive status input that is independent of H1(H3) and the transfer of port data. On the asserted edge of H2(H4), the H2S(H4S) status bit is set. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H12 enable (H34 enable) bit of the port general control register is clear.
2. H2(H4) may be a general-purpose output pin that is always negated. In this case the H2S(H4S) status bit is always clear.
3. H2(H4) may be a general-purpose output pin that is always asserted. In this case the H2S(H4S) status bit is always clear.

Programmable Option Mode 0 - Port A Submode 1X and Port B Submode 1X

PACR

7 6	**Port A Submode**
1 X	Submode 1X

PACR

5 4 3	**H2 Control**
0 X X	Input pin - edge-sensitive status input, H2S is set on an asserted edge.
1 X 0	Output pin - negated, H2S is always cleared.
1 X 1	Output pin - asserted, H2S is always cleared.

PACR

2	**H2 Interrupt Enable**
0	The H2 interrupt is disabled.
1	The H2 interrupt is enabled.

PACR

1	**H1 SVCRQ Enable**
0	The H1 interrupt is disabled.
1	The H1 interrupt is enabled.

PACR

0	**H1 Status Control**
X	H1 is an edge-sensitive status input, H1S is set by an asserted edge of H1.

PBCR

7 6	**Port B Submode**
1 X	Submode 1X.

PBCR

5 4 3	**H4 Control**
0 X X	Input pin - edge-sensitive status input, H4S is set on an asserted edge.
1 X 0	Output pin - negated, H4S is always cleared.
1 X 1	Output pin - asserted, H4S is always cleared.

PBCR

2	**H4 Interrupt Enable**
0	The H4 interrupt is disabled.
1	The H4 interrupt is enabled.

PBCR

1	**H3 SVCRQ Enable**
0	The H3 interrupt is disabled.
1	The H3 interrupt is enabled.

Programmable Options Mode 0 - Port A Submode 1X and Port B Submode 1X (continued)

PBCR

0	**H3 Status Control**
X	H3 is an edge-sensitive status input, H3S is set by an asserted edge of H3.

3.4. MODE 1 - UNIDIRECTIONAL 16-BIT MODE

In mode 1, ports A and B are concatenated to form a single 16-bit port. The port B submode field controls the configuration of both ports. The possible submodes are :

Port B Submode X0 - Pin-Definable Double-Buffered Input or Single-Buffered Output

Port B Submode X1 - Pin-Definable Double-Buffered Output or Non-Latched Input

Handshake pins H3 and H4, configured by programming the port B control register, are associated with the 16-bit double-buffered transfer. These 16-bit transfers are enabled by setting the H34 enable bit in the port general control register (PGCR). Handshake pins H1 and H2 may be used as simple status inputs not related to the 16-bit data transfer or H2 may be an output. Enabling of the H1 and H2 handshake pins is done by setting the H12 enable bit of the port general control register. The port A and B data direction registers operate in each submode. Along with the submode, they affect the data read and written at the data register according to table 3.2. The data direction register also enables the output buffer associated with each port pin. The DMAREQ pin may be associated only with H3.

Table 3.2 : Mode 1 Port Data Paths.

Mode	Read Port A/B Register		Write Port A/B Register	
	DDR = 0	DDR = 1	DDR = 0	DDR = 1
1, Port B Submode X0	FIL, D. B.	FOL Note 3	FOL, S. B. Note 2	FOL, S. B. Note 2
1, Port B Submode X1	Pin	FOL Note 3	IOL/FOL, D. B. Note 1	IOL/FOL, D. B. Note 1

Note 1 : Data written to Port A goes to a temporary latch. When the Port B data register is later written, Port A data is transferred to IOL/FOL.
Note 2 : Data is latched in the output data registers (final output latch) and will be single buffered at the pin if the DDR is 1. The output buffers will be turned off if the DDR is 0.
Note 3 : The output drivers that connect the final output latch to the pins are turned on.

Abbreviations :

IOL - Initial Output Latch	S. B. - Single Buffered
FOL - Final Output Latch	D. B. - Double Buffered
FIL - Final Input Latch	DDR - Data Direction Register

Mode 1 can provide convenient high-speed 16-bit transfers. The port A and port B data registers are addressed for compatibility with the TS68000 move peripheral (MOVEP) instruction and with the 68440 direct memory access controller (DMAC). To take advantage of this, port A should contain the most-significant byte of data and always be read or written by the bus master first. The interlocked and pulsed handshake protocols, status bits, and DMAREQ are keyed to the access of port B data register in mode 1. Transfers proceed properly with interlocked or pulsed handshakes when the port B data register is accessed last.

3.4.1. PORT A CONTROL REGISTER (PACR).

7 6	5 4 3	2	1	0
Port A Submode	H2 Control	H2 Interrupt Enable	H1 SVCRQ Enable	H1 Status Control

The port A control register, in conjunction with the programmed mode and the port B submode, controls the operation of port A and the handshake pins H1 and H2. The port A control register contains five fields : bits 7 and 6 specify the port A submode ; bits 5, 4, and 3 control the operation of the H2 handshake pin and H2S status bit ; bit 2 determines whether an interrupt will be generated when the H2S status bit goes to one ; bit 1 determines whether a service request (interrupt request or DMA request) will occur ; and bit 0 controls the operation of the H1S status bit. The PACR is always readable and writable. There is never a consequence to reading the register.

All bits are cleared to zero when the $\overline{\text{RESET}}$ pin is asserted. When the port A submode field is relevant in a mode/submode definition, it must not be altered unless the H12 enable bit in the port general control register is clear (see table 1.3 located at the end of this document). Altering these bits may give unpredictable results if the H12 enable bit in the PGCR is set.

3.4.2. PORT B CONTROL REGISTER (PBCR).

7 6	5 4 3	2	1	0
Port B Submode	H4Control	H4 Interrupt Enable	H3 SVCRQ Enable	H3 Status Control

The port B control register specifies the operation of port B and the handshake pins H3 and H4. The port B control register contains five fields : bits 7 and 6 specify the port B submode ; bits 5, 4, and 3 control the operation of the H4 handshake pin and H4S status bit goes to a one ; bit 1 determines whether a service request (interrupt request or DMA request) will occur ; and bit 0 controls the operation of the H3S status bit. The PBCR is always readable and writable.

All bits are cleared to zero when the $\overline{\text{RESET}}$ pin is asserted. When the port B submode field is relevant in a mode/submode definition, it must not be altered unless the H34 enable bit in the port general control register is clear (see table 1.3 located at the end of this document). Altering these bits may give unpredictable results if the H12 enable bit in the PGCR is set.

3.4.3. SUBMODE X0 - PIN-DEFINABLE DOUBLE-BUFFERED INPUT OR SINGLE-BUFFERED OUTPUT.

In mode 1 submode X0, double-buffered input transfers of up to 16 bits may be obtained. The level of each pin is asynchronously latched with the asserted edge of H3 and placed in the initial input latch or the final input latch. The processor may check the H3S status bit to determine if new data is present. The $\overline{\text{DMAREQ}}$ pin may be used to signal a DMA controller to empty the input buffers. Regardless of the bus master, port A data should be read first and port B data should be read last. The operation of the internal handshake controller, the H3S bit, and the $\overline{\text{DMAREQ}}$ are keyed to the reading of the port B data register. (The 68440 DMAC can be programmed to perform the exact transfers needed for compatibility with the PI/T.) H4 may be programmed as :

1. H4 may be an edge-sensitive status input that is independent of H3 and the transfer of port data. On the asserted edge of H4, the H4S status bit is set. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H34 enable bit of the port general control register is clear.
2. H4 may be a general-purpose output pin that is always negated. In this case the H4S status bit is always clear.
3. H4 may be a general-purpose output pin that is always asserted. In this case the H4S status bit is always clear.
4. H4 may be an output pin in the interlocked input handshake protocol. It is asserted when the port input latches are ready to accept new data. It is negated asynchronously following the asserted edge of the H3 input. As soon as the input latches become ready, H4 is again asserted. When the input double-buffered latches are full, H4 remains negated until data is removed. Thus, anytime the H4 output is asserted, new input data may be entered by asserting H3. At other times transitions on H3 are ignored. The H4S status bit is always clear. When H34 enable in the port general control register is clear, H4 is held negated.

5. H4 may be an output pin in the pulsed input handshake protocol. It is asserted exactly as in the interlocked input protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case that a subsequent H3 asserted edge occurs before termination of the pulse, H4 is negated asynchronously. Thus, anytime after the leading edge of the H4 pulse, new data may be entered in the double-buffered input latches. The H4S status bit is always clear. When H34 enable is clear, H4 is held negated.

For pins used as outputs, the data path consists of a single latch driving the output buffer. Data written to the port's data register does not affect the operation of any handshake pin, status bit, or any other aspect of the PI/T. Thus, output pins may be used independently of the input transfer.

The programmable options of the H2 pin are :

1. H2 may be an edge-sensitive input pin independent of H1 and the transfer of port data. On the asserted edge of H2, the H2S status bit is set. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H12 enable bit of the port general control register is clear.
2. H2 may be a general-purpose output pin that is always negated. The H2S status bit is always clear.
3. H2 may be a general-purpose output pin that is always asserted. The H2S status bit is always clear.

Programmable Options Mode 1 - Port A Submode XX and Port B Submode X0

PACR

7 6	**Port A Submode**
0 0	Submode XX

PACR

5 4 3	**H2 Control**
0 X X	Input pin - edge-sensitive status input, H2S is set on an asserted edge.
1 X 0	Output pin - negated, H2S is always cleared.
1 X 1	Output pin - asserted, H2S is always cleared.

PACR

2	**H2 Interrupt Enable**
0	The H2 interrupt is disabled.
1	The H2 interrupt is enabled.

PACR

1	**H1 SVCRQ Enable**
0	The H1 interrupt is disabled.
1	The H1 interrupt is enabled.

PACR

0	**H1 Status Control.**
X	H1 is an edge-sensitive status input. H1S is set by an asserted edge of H1.

PBCR

7 6	**Port B Submode**
0 0	Submode X0.

Programmable Options Mode 1 - Port A Submode XX and Port B Submode X0 (continued)

PBCR

5 4 3	**H4 Control**
0 X X	Input pin - edge-sensitive status input, H4S is set on an asserted edge.
1 0 0	Output pin - negated, H4S is always cleared.
1 0 1	Output pin - asserted, H4S is always cleared.
1 1 0	Output pin - interlocked input handshake protocol.
1 1 1	Output pin - pulsed input handshake protocol.

PBCR

2	**H2 Interrupt Enable**
0	The H4 interrupt is disabled.
1	The H4 interrupt is enabled.

PBCR

1	**H3 SVCRQ Enable**
0	The H3 interrupt and DMA request are disabled.
1	The H3 interrupt and DMA request are enabled.

PBCR

0	**H3 Status Control**
X	The H3S status bit is set anytime input data is present in the double-buffered input path.

3.4.4. SUBMODE X1 - PIN-DEFINABLE DOUBLE-BUFFERED OUTPUT OR NON-LATCHED INPUT.
In mode 1 submode X1, double-buffered output transfers of up to 16 bits may be obtained. Data is written by the bus master (processor or DMA controller) in two bytes. The first byte (most significant) is written to the port A data register. It is stored in a temporary latch until the next byte is written to the port B data register. Then all 16 bits are transferred to one of the output latches of ports A and B. The DMAREQ pin may be used to signal a DMA controller to transfer another word to the port output latches. (The 68440 DMAC can be programmed to perform the exact transfers needed for compatibility with the PI/T.) H4 may be programmed as :

1. H4 may be an edge-sensitive status input that is independent of H3 and the transfer of port data. On the asserted edge of H4, the H4S status bit is set. It is cleared by either the RESET pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H34 enable bit of the port general control register is clear.
2. H4 may be a general-purpose output pin that is always negated. In this case the H4S status bit is always clear.
3. H4 may be a general-purpose output pin that is always asserted. In this case the H4S status bit is always clear.
4. H4 may be an output pin in the interlocked output handshake protocol. H4 is asserted two clock cycles after data is transferred to the double-buffered output latches. The data remains stable at the port pins and H4 remains asserted until the next asserted edge of the H3 input. At that time, H4 is asynchronously negated. As soon as the next data is available, it is transferred to the output latches. When H4 is negated, asserted transitions of H3 have no affect on data paths. The H4S status bit is always clear. When H34 enable is clear, H4 is held negated.
5. H4 may be an output pin in the pulsed output handshake protocol. It is asserted exactly as in the interlocked protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock pulse is generated. But in the case that a subsequent H3 asserted edge occurs before termination of the pulse, H4 is negated asynchronously shortening the pulse. The H4S status bit is always cleared. When H34 enable is clear, H4 is held negated.

The H3S status bit may be programmed for two interpretations :

1. The H3S status bit is set when either the port initial or final output latch can accept new data. It is clear when both latches are full and cannot accept new data.
2. The H3S status bit is set when both of the port output latches are empty. It is clear when at least one latch is full.

The programmable options of the H2 pin are :

1. H2 may be an edge-sensitive input pin independent of H1 and the transfer of port data. On the asserted edge of H2, the H2S status bit is set. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H12 enable bit of the port general control register is clear.
2. H2 may be a general-purpose output pin that is always negated. The H2S status bit is always clear.
3. H2 may be a general-purpose output pin that is always asserted. The H2S status bit is always clear.

For pins used as inputs, data written to either data register is double buffered and passed to the initial or final output latch, as usual, but the output buffer is disabled (refer to **3.3.2. Submode 01 - Pin-Definable Double-Buffered Output or Non-Latched Input**).

Programmable Options Mode 1 - Port A Submode XX and Port B Submode X1

PACR

7 6	**Port A Submode**
0 0	Submode XX.

PACR

5 4 3	**H2 Control**
0 X X	Input pin - edge-sensitive status input, H2S is set on an asserted edge.
1 X 0	Output pin - negated, H2S is always cleared.
1 X 1	Output pin - asserted, H2S is always cleared.

PACR

2	**H2 Interrupt Enable**
0	The H2 interrupt is disabled.
1	The H2 interrupt is enabled.

PACR

1	**H1 SVCRQ Enable**
0	The H1 interrupt is disabled.
1	The H1 interrupt is enabled.

PACR

0	**H1 Status Control**
X	H1 is an edge-sensitive status input. H1S is set by an asserted edge of H1.

PBCR

7 6	**Port B Submode**
0 0	Submode X1.

Programmable Options Mode 1 - Port A Submode XX and Port B Submode X1 (continued)

PBCR

5 4 3	**H4 Control**
0 X X	Input pin - edge-sensitive status input, H4S is set on an asserted edge.
1 0 0	Output pin - negated, H4S is always cleared.
1 0 1	Output pin - asserted, H4S is always cleared.
1 1 0	Output pin - interlocked input handshake protocol.
1 1 1	Output pin - pulsed input handshake protocol.

PBCR

2	**H4 Interrupt Enable**
0	The H4 interrupt is disabled.
1	The H4 interrupt is enabled.

PBCR

1	**H3 SVCRQ Enable**
0	The H3 interrupt and DMA request are disabled.
1	The H3 interrupt and DMA request are enabled.

PBCR

0	**H3 Status Control**
0	The H3S status bit is set when either the initial or final output latch of ports A and B can accept new data. It is clear when both latches are full and cannot accept new data.
1	The H3S status bit is set when both the initial and final output latches of ports A and B are empty. The H3S status bit is clear when at least one set of output latches is full.

3.5. MODE 2 - BIDIRECTIONAL 8-BIT MODE

In mode 2, port A is used for bit I/O with no associated handshake pins. Port B is used for bidirectional 8-bit double-buffered transfers. H1 and H2, enabled by the H12 enable bit in the port general control register, control output transfers, while H3 and H4, enabled by the port general control register bit H34 enable, control input transfers. The instantaneous direction of the data is determined by the H1 handshake pin. The port B data direction register is not used. The port A and port B submode fields do not affect PI/T operation in mode 2.

3.5.1. PORT A BIT I/O (PIN-DEFINABLE SINGLE-BUFFERED OUTPUT OR NON-LATCHED INPUT).

Mode 2, port A performs simple bit I/O with no associated handshake pins. This configuration is intended for applications in which several independent devices must be controlled or monitored. Data written to the port A data register is single buffered. If the port A data direction register bit for that pin is set (output), the output buffer is enabled. If it is zero (input), data written is still latched but not available at the pin. Data read from the data register is either the instantaneous value of the pin (if data is stable from $\overline{CS}$ asserted to $\overline{DTACK}$ asserted, data on these pins will be guaranteed valid in the data register) or what was written to the data register, depending on the contents of the port A data direction register. This is summarized in table 3.3.

Table 3.3 : Mode 2 Port A Data Paths.

Mode	Read Port A Data Register		Write Port A Data Register	
	DDR = 0	DDR = 1	DDR = 0	DDR = 1
2	Pin	FOL	FOL	FOL, S. B.

Abbreviations :
S. B. - Single Buffered
FOL - Final Output Latch
DDR - Data Direction Register

3.5.2. PORT B - DOUBLE-BUFFERED BIDIRECTIONAL DATA. The output buffers of port B are controlled by the level of H1. When H1 is negated, the port B output buffers (all eight) are enabled and the pins drive the bidirectional bus. Generally, H1 is negated by the peripheral in response to an asserted H2, which indicates that new output data is present in the double-buffered latches. Following acceptance of the data, the peripheral asserts H1, disabling the port B output buffers. Other than controlling the output buffers, H1 is edge-sensitive as in other modes.

3.5.2.1. Double-Buffered Input Transfers.

Port B input data that meets the port setup and hold times is latched on the asserted edge of H3 and placed in the initial input latch or the final input latch. H3 is edge-sensitive, and may assume any duty-cycle as long as both high and low minimum times are observed. The PI/T contains a port status register whose H3S status bit is set anytime any input data that has not been read by the bus master is present in the double-buffered latches. The action of H4 is programmable and can be programmed as :

1. H4 may be an output pin in the interlocked input handshake protocol. It is asserted when the port input latches are ready to accept new data. It is negated asynchronously following the asserted edge of the H3 input. As soon as the input latches become ready, H4 is again asserted. When the input double-buffered latches are full, H4 remains negated until data is removed. Thus, anytime the H4 output is asserted, new input data may be entered by asserting H3. At other times transitions on H3 are ignored. The H4S status bit is always clear. When H34 enable in the port general control register is clear, H4 is held negated.
2. H4 may be an output pin in the pulsed input handshake protocol. It is asserted exactly as in the interlocked input protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case that a subsequent H3 asserted edge occurs before termination of the pulse, H4 is negated asynchronously. Thus, anytime after the leading edge of the H4 pulse, new data may be entered in the double-buffered input latches. The H4S status bit is always clear. When H34 enable is clear, H4 is held negated.

3.5.2.2. Double-Buffered Output Transfers.

Data, written by the bus master to the PI/T, is stored in the port's output latch. The peripheral accepts the data by asserting H1, which causes the next data to be moved to the port's output latch as soon as it is available. The H1S status bit, in the port status register, may be programmed for two interpretations. Normally the status bit is a one when there is at least one latch in the double-buffered data path that can accept new data. After writing one byte of data to the ports, an interrupt service routine could check this bit to determine if it could store another byte ; thus filling both latches. When the bus master is finished, it is often useful to be able to check whether all of the data has been transferred to the peripheral.The H1S status control bit of the port A control register provides this flexibility. The H1S status bit is set when both output latches are empty. The programmable options for H2 are :

1. H2 may be an output pin in the interlocked output handshake protocol. It is asserted when the port output latches are ready to transfer new data. It is negated asynchronously following the asserted edge of the H1 input. As soon as the output latches become ready, H2 is again asserted. When the output double-buffered latches are full, H2 remains asserted until data is removed. Thus, anytime the H2 output is asserted, new output data may be transferred by asserting H1. At other times transitions on H1 are ignored. The H2S status bit is always clear. When H12 enable in the port general control register is clear, H2 is held negated.
2. H2 may be an output pin in the pulsed output handshake protocol. It is asserted exactly as in the interlocked output protocol above, but never remains asserted longer than four clock

cycles. Typically, a four clock cycle pulse is generated. But in the case that a subsequent H1 asserted edge occurs before termination of the pulse, H2 is negated asynchronously. Thus, anytime after the leading edge of the H2 pulse, new data may be transferred to the double-buffered output latches. The H2S status bit is always clear. When H12 enable is clear, H2 is held negated.

The $\overline{\text{DMAREQ}}$ pin may be associated with either input transfers (H3) or output transfers (H1), but not both. Refer to table 3.4 for a summary of the port B data register responses in mode 2.

Table 3.4 : Mode 2 Port B Data Paths.

Mode	Read Port B Data Register	Write Port B Data Register
2	FIL, D. B.	IOL/FOL, D. B.
Abbreviations : IOL - Initial Output Latch FOL - Final Output Latch FIL - Final Input Latch	D. B. - Double Buffered	

Programmable Options Mode 2 - Port A Submode XX and Port B Submode XX

PACR

7 6	**Port A Submode**
X X	Submode XX.

PACR

5 4 3	**H2 Control**
X X 0	Output pin - interlocked output handshake protocol, H2S is always cleared.
X X 1	Output pin - pulsed output handshake protocol, H2S is always cleared.

PACR

2	**H2 Interrupt Enable**
0	The H2 interrupt is disabled.
1	The H2 interrupt is enabled.

PACR

1	**H1 SVCRQ Enable**
0	The H1 interrupt and DMA request are disabled.
1	The H1 interrupt and DMA request are enabled.

PACR

0	**H1 Status Control**
0	The H1 status bit is set when either the port B initial or final output latch can accept new data. It is clear when both latches are full and cannot accept new data.
1	The H1S status bit is set when both of the port B output latches are empty. It is clear when at least one latch is full.

PBCR

7 6	**Port B Submode**
X X	Submode XX.

Programmable Options Mode 2 - Port A Submode XX and Port B Submode XX (continued)

PBCR

5 4 3	**H4 Control**
X X 0	Output pin - interlocked input handshake protocol, H4S is always cleared.
X X 1	Output pin - pulsed input handshake protocol, H4S is always cleared.

PBCR

2	**H4 Interrupt Enable**
0	The H4 interrupt is disabled.
1	The H4 interrupt is enabled.

PBCR

1	**H3 SVCRQ Enable**
0	The H3 interrupt and DMA request are disabled.
1	The H3 interrupt and DMA request are enabled.

PBCR

0	**H3 Status Control**
X	The H3S status bit is set anytime input data is present in the double-buffered input path.

3.6. MODE 3 - BIDIRECTIONAL 16-BIT MODE

In mode 3, ports A and B are used for bidirectional 16-bit double-buffered transfers. H1 and H2 control output transfers, while H3 and H4 control input transfers. H1 and H2 are enabled by the H12 enable bit while H3 and H4 are enabled by the H34 enable bit of the port general control register. The instantaneous direction of data is determined by the H1 handshake pin, thus, the data direction registers are not used and have no affect. The port A and port B submode fields do not affect PI/T operation in mode 3. Port A and port B output buffers are controlled by the level of H1. When H1 is negated, the output buffers (all 16) are enabled and the pins drive the bidirectional port bus. Generally a peripheral will negate H1 in response to an asserted H2, which indicates that new output data is present in the double-buffered latches. Following acceptance of the data, the peripheral asserts H1, disabling the output buffers. Other than controlling the output buffers, H1 is edge-sensitive as in other modes. The port A and port B data direction registers are not used.

3.6.1. DOUBLE-BUFFERED INPUT TRANSFERS.

Port A and B input data that meets the port setup and hold times is latched on the asserted edge of H3 and placed in the initial input latch or the final input latch. H3 is edge-sensitive, and may assume any duty-cycle as long as both high and low minimum times are observed. The PI/T contains a port status register whose H3S status bit is set anytime any input data is present in the double-buffered latches that has not been read by the bus master. The action of H4 is programmable and can be programmed as :

1. H4 may be an output pin in the interlocked input handshake protocol. It is asserted when the port input latches are ready to accept new data. It is negated asynchronously following the asserted edge of the H3 input. As soon as the input latches become ready, H4 is again asserted. When the input double-buffered latches are full, H4 remains negated until data is removed. Thus, anytime the H4 output is asserted, new input data may be entered by asserting H3. At other times transitions on H3 are ignored. The H4S status bit is always clear. When H34 enable in the port general control register is clear, H4 is held negated.
2. H4 may be an output pin in the pulsed input handshake protocol. It is asserted exactly as in the interlocked input protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case that a subsequent H3 asserted edge occurs before termination of the pulse, H4 is negated asynchronously. Thus, anytime after the leading edge of the H4 pulse, new data may be entered in the double-buffered input latches. The H4 status bit is always clear. When H34 enable is clear, H4 is held negated.

3.6.2. DOUBLE-BUFFERED OUTPUT TRANSFERS. Data, written by the bus master to the PI/T, is stored in the port's output latch. The peripheral accepts the data by asserting H1, which causes the next data to be moved to the port's output latch as soon as it is available. The H1S status bit, in the port status register, may be programmed for two interpretations. Normally the status bit is a one when there is at least one latch in the double-buffered data path that can accept new data. After writing one byte of data to the ports, an interrupt service routine could check this bit to determine if it could store another byte ; thus filling both latches. When the bus master is finished, it is often useful to be able to check whether all of the data has been transferred to the peripheral. The H1S status control bit of the port A control register provides this flexibility. The H1S status bit is set when both output latches are empty. The programmable options for H2 are :

1. H2 may be an output pin in the interlocked output handshake protocol. It is asserted when the port output latches are ready to transfer new data. It is negated asynchronously following the asserted edge of the H1 input. As soon as the output latches become ready, H2 is again asserted. When the output double-buffered latches are full, H2 remains asserted until data is removed. Thus, anytime the H2 output is asserted, new output data may be transferred by asserting H1. At other times transitions on H1 are ignored. The H2S status bit is always clear. When H12 enable in the port general control register is clear, H2 is held negated.
2. H2 may be an output pin in the pulsed output handshake protocol. It is asserted exactly as in the interlocked output protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock pulse is generated. But in the case that a subsequent H1 asserted edge occurs before termination of the pulse, H2 is negated asynchronously shortening the pulse. The H2S status bit is always zero. When H12 enable is zero, H2 is held negated.

Mode 3 can provide convenient high-speed 16-bit transfers. The port A and B data registers are addressed for compatibility with the TS68000's move peripheral (MOVEP) instruction and with the 68440 DMAC. To take advantage of this port A should contain the most significant data and always be read or written by the bus master first. The interlocked and pulsed handshake protocols, status bits, and DMAREQ are keyed to the access of port B data register in mode 3. If it is accessed last, the 16-bit double-buffered transfer proceeds smoothly.

The $\overline{\text{DMAREQ}}$ pin may be associatedwith either input transfers (H3) or output transfers (H1), but not both. Refer to table 3.5 for a summary of the port A and B data paths in mode 3.

Table 3.5 : Mode 3 Port A and B Data Paths.

Mode	Read Port A and B Data Register	Write Port A and B Data Register
3	FIL, D. B.	IOL/FOL, D. B., Note 1
Note 1 : Data written to Port A goes to a temporary latch. When the Port B data register is later written, Port A data is transferred to IOL/FOL.		
Abbreviations : IOL - Initial Output Latch FOL - Final Output Latch FIL - Final Input Latch	S. B. - Single Buffered D. B. - Double Buffered	

Programmable Options Mode 3 - Port A Submode XX and Port B Submode XX

PACR

7 6	**Port A Submode**
X X	Submode XX.

PACR

5 4 3	**H2 Control**
X X 0	Output pin - interlocked output handshake protocol, H2S status always cleared.
X X 1	Output pin - pulsed output handshake protocol, H2S status always cleared.

PACR

2	**H2 Interrupt Enable**
0	The H2 interrupt is disabled.
1	The H2 interrupt is enabled.

PACR

1	**H1 SVCRQ Enable**
0	The H1 interrupt and DMA request are disabled.
1	The H1 interrupt and DMA request are enabled.

PACR

0	**H1 Status Control**
0	The H1 status bit is set when either the port B initial or final output latch can accept new data. It is clear when both latches are full and cannot accept new data.
1	The H1S status bit is set when both of the port B output latches are empty. It is clear when at least one latch is full.

PBCR

7 6	**Port B Submode**
X X	Submode XX.

PBCR

5 4 3	**H4 Control**
X X 0	Output pin - interlocked input handshake protocol, H4S is always clear.
X X 1	Output pin - pulsed input handshake, H4S is always clear.

PBCR

2	**H4 Interrupt Enable**
0	The H4 interrupt is disabled.
1	The H4 interrupt is enabled.

PBCR

1	**H3 SVCRQ Enable**
0	The H3 interrupt and DMA request are disabled.
1	The H3 interrupt and DMA request are enabled.

PBCR

0	**H3 Status Control**
X	The H3S status bit is set anytime input data is present in the double-buffered input path.

SECTION 4

PROGRAMMER'S MODEL

This section describes the internal accessible register organization as represented in table 1.3 located at the end of this document and in table 4.1. Address space within the address map is reserved for future expansion.

Table 4.1 : PI/T Register Addressing Assignments.

Register		Register Select Bits					Accessible	Affected by Reset	Affected by Read Cycle
		5	4	3	2	1			
Port General Control Register	(PGCR)	0	0	0	0	0	R W	Yes	No
Port Service Request Register	(PSRR)	0	0	0	0	1	R W	Yes	No
Port A Data Direction Register	(PADDR)	0	0	0	1	0	R W	Yes	No
Port B Data Direction Register	(PBDDR)	0	0	0	1	1	R W	Yes	No
Port C Data Direction Register	(PCDDR)	0	0	1	0	0	R W	Yes	No
Port Interrupt Vector Register	(PIVR)	0	0	1	0	1	R W	Yes	No
Port A Control Register	(PACR)	0	0	1	1	0	R W	Yes	No
Port B Control Register	(PBCR)	0	0	1	1	1	R W	Yes	No
Port A Data Register	(PADR)	0	1	0	0	0	R W	No	**
Port B Data Register	(PBDR)	0	1	0	0	1	R W	No	**
Port A Alternate Register	(PAAR)	0	1	0	1	0	R	No	No
Port B Alternate Register	(PBAR)	0	1	0	1	1	R	No	No
Port C Data Register	(PCDR)	0	1	1	0	0	R W	No	No
Port Status Register	(PSR)	0	1	1	0	1	R W*	Yes	No
Timer Control Register	(TCR)	1	0	0	0	0	R W	Yes	No
Timer Interrupt Vector Register	(TIVR)	1	0	0	0	1	R W	Yes	No
Counter Preload Register High	(CPRH)	1	0	0	1	1	R W	No	No
Counter Preload Register Middle	(CPRM)	1	0	1	0	0	R W	No	No
Counter Preload Register Low	(CPRL)	1	0	1	0	1	R W	No	No
Count Register High	(CNTRH)	1	0	1	1	1	R	No	No
Count Register Middle	(CNTRM)	1	1	0	0	0	R	No	No
Count Register Low	(CNTRL)	1	1	0	0	1	R	No	No
Timer Status Register	(TSR)	1	1	0	1	0	R W*	Yes	No

* A write to this register may perform a special resetting operation.
** Mode dependant.

R = Read.
W = Write.

Throughout this section the following conventions are maintained :

1. A read from a reserved location in the map results in a read from the "null register". The null register returns all zeros for data and results in a normal bus cycle. A write to one of these locations results in a normal bus cycle, but written data is ignored.
2. Unused bits of a defined register are denoted by "*" and are read as zeros ; written data is ignored.
3. Bits that are unused in the chosen mode/submode but are used in others are denoted by "X", and are readable and writable. Their content, however, is ignored in the chosen mode/submode.
4. All registers are addressable as 8-bit quantities. To facilitate operation with the MOVEP instruction and the DMAC, addresses are ordered such that certain sets of registers may also be accessed as words (two bytes) or long words (four bytes).

4.1. PORT GENERAL CONTROL REGISTER (PGCR)

7	6	5	4	3	2	1	0
Port Mode Control		H34 Enable	H12 Enable	H4 Sense	H3 Sense	H2 Sense	H1 Sense

The port general control register controls many of the functions that are common to the overall operation of the ports. The PGCR is composed of three major fields : bits 7 and 6 define the operational mode of ports A and B and affect operation of the handshake pins and status bits ; bits 5 and 4 allow a software-controlled disabling of particular hardware associated with the handshake pins of each port ; and bits 3-0 define the sense of the handshake pins. The PGCR is always readable and writable.

All bits are reset to zero when the $\overline{\text{RESET}}$ pin is asserted.

The port mode control field should be altered only when the H12 enable and H34 enable bits are zero. Except when mode is desired (submode 1X), the port general control register should be written once to establish the mode with the H12 and H34 bits clear. Any other necessary control registers can then be programmed, after which H12 or H34 is set. In order to enable the respective operation(s), the port general control register should be written again.

PGCR

7 6	**Port Mode Control**
0 0	Mode 0 (Unidirectional 8-Bit Mode).
0 1	Mode 1 (Unidirectional 16-Bit Mode).
1 0	Mode 2 (Bidirectional 8-Bit Mode).
1 1	Mode 3 (Bidirectional 16-Bit Mode).

PGCR

5	**H34 Enable**
0	Disabled.
1	Enabled.

PGCR

4	**H12 Enable**
0	Disabled.
1	Enabled.

PGCR

0-	**0 Handshake Pin Sense**
0	The associated pin is at the high-voltage level when negated and at the low-voltage level when asserted.
1	The associated pin is at the low-voltage level when negated and at the high-voltage level when asserted.

4.2. PORT SERVICE REQUEST REGISTER (PSRR)

7	6	5	4	3	2	1	0
*	SVCRQ Select		Operation Select		Port Interrupt Priority Control		

The port service request register controls other functions that are common to the overall operation to the ports. It is composed of four major fields : bit 7 is unused and is always read as zero ; bits 6 and 5 define whether interrupt or DMA requests are generated from activity on the H1 and H3 handshake pins ; bits 4 and 3 determine whether two dual-function pins operate as port C or port interrupt request/acknowledge pins ; and bits 2, 1, and 0 control the priority among all port interrupt sources. Since bits 2, 1, and 0 affect interrupt operation, it is recommended that they be changed only when the affected interrupt(s) is (are) disabled or known to remain inactive. The PSRR is always readable and writable.

All bits are reset to zero when the $\overline{\text{RESET}}$ pin is asserted.

PSRR

6 5	**SVCRQ Select**
0 X	The PC4/$\overline{\text{DMAREQ}}$ pin carries the PC4 function ; DMA is not used.

PSRR **SVCRQ Select**

1 0 The PC4/$\overline{\text{DMAREQ}}$ pin carries the $\overline{\text{DMAREQ}}$ function and is associatedwith double-buffered transfers controlled by H1. H1 is removed from PI/T's interrupt structure, and thus, does not cause interrupt requests to be generated. To obtain DMAREQ pulses, port A control register bit 1 (H1 SVCRQ enable) must be a one.

1 1 The PC4/$\overline{\text{DMAREQ}}$ pin carries the $\overline{\text{DMAREQ}}$ function and is associated with double-buffered transfers controlled by H3. H3 is removed from the PI/T's interrupt structure, and thus, does not cause interrupts requests to be generated. To obtain $\overline{\text{DMAREQ}}$ pulses, port B control register bit 1 (H3 SVCRQ enable) must be one.

PSRR

4 3 **Interrupt Pin Function Select**

0 0 The PC5/$\overline{\text{PIRQ}}$ pin carries the PC5 function, no interrupt support.
The PC6/$\overline{\text{PIACK}}$ pin carries the PC6 function, no interrupt support.

0 1 The PC5/$\overline{\text{PIRQ}}$ pin carries the $\overline{\text{PIRQ}}$ function, supports autovectored interrupts.
The PC6/$\overline{\text{PIACK}}$ pin carries the PC6 function, supports autovectored interrupts.

1 0 The PC5/$\overline{\text{PIRQ}}$ pin carries the PC5 function.
The PC6/$\overline{\text{PIACK}}$ pin carries the $\overline{\text{PIACK}}$ function.

1 1 The PC5/$\overline{\text{PIRQ}}$ pin carries the $\overline{\text{PIRQ}}$ function, supports vectored interrupts.
The PC6/$\overline{\text{PIACK}}$ pin carries the $\overline{\text{PIACK}}$ function, supports vectored interrupts.

Bits 2, 1, and 0 determine port interrupt priority. The priority as shown in table 4.2 is in descending order left to right.

Table 4.2 : PSRR Port Interrupt Priority Control.

210	**Highest**			**Lowest**
000	H1S	H2S	H3S	H4S
001	H2S	H1S	H3S	H4S
010	H1S	H2S	H4S	H3S
011	H2S	H1S	H4S	H3S

210	**Highest**			**Lowest**
100	H3S	H4S	H1S	H2S
101	H3S	H4S	H2S	H1S
110	H4S	H3S	H1S	H2S
111	H4S	H3S	H2S	H1S

4.3. PORT DATA DIRECTION REGISTERS

The following paragraphs describe the port data direction registers.

4.3.1. PORT A DATA DIRECTION REGISTER (PADDR). The port A data direction register determines the direction and buffering characteristics of each of the port A pins. One bit in the PADDR is assigned to each pin. A zero indicates that the pin is used as a input, while a one indicates it is used as an output. The PADDR is always readable and writable. This register is ignored in mode 3.

All bits are reset to the zero (input) state when the RESET pin is asserted.

4.3.2. PORT B DATA DIRECTION REGISTER (PBDDR). The PBDDR is identical to the PADDR for the port B pins and the port B data register, except that this register is ignored in modes 2 and 3.

4.3.3. PORT C DATA DIRECTION REGISTER (PCDDR). The port C data direction register specifies whether each dual-function pin that is chosen for port C operation is an input (zero) or an output (one) pin. The PCDDR, along with bits that determine the respective pin's function, also specify the exact hardware to be accessed at the port C data register address (see **4.6.3. Port C Data Register** (PCDR) for more details). The PCDDR is an 8-bit register that is readable and writable at all times. Its operation is independent of the chosen PI/T mode.

These bits are cleared to zero when the RESET pin is asserted.

4.4. PORT INTERRUPT VECTOR REGISTER (PIVR)

7	6	5	4	3	2	1	0
Interrupt Vector Number						*	*

The port interrupt vector register contains the upper order six bits of the four port interrupt vectors. The contents of this register may be read two ways : by an ordinary read cycle, or by a port interrupt acknowledge bus cycle. The exact data read depends on how the cycle was initiated and other factors. Behavior during a port interrupt acknowledge cycle is summarized in table 2.1.

From a normal read cycle, there is never a consequence to reading this register. Following negation of the RESET pin, but prior to writing to the PIVR, a $0F will be read. After writing to the register, the upper six bits may be read and the lower two bits are forced to zero. No prioritization computation is performed.

4.5. PORT CONTROL REGISTERS (PACR, PBCR)

The port A and B control registers (PACR and PBCR) are described in **Section 3 Port Modes.** The description is organized such that for each mode/submode all programmable options of each pin and status bit are given.

4.6. PORT DATA REGISTERS

The following paragraphs describe the port data registers.

4.6.1. PORT A DATA REGISTER (PADR). The port A data register is a holding register for moving data to and from the port A pins. The port A data direction register determines whether each pin is an input (zero) or an output (one), and is used in configuring the actual data paths. The data paths are described in **Section 3 Port Modes.**

This register is readable and writable at all times. Depending on the chosen mode/submode, reading or writing may affect the double-buffered handshake mechanism. The port A data register is not affected by the assertion of the RESET pin.

4.6.2. PORT B DATA REGISTER (PBDR). The port B data register is a holding register for moving data to and from port B pins. The port B data direction register determines whether each pin is an input (zero) or an output (one), and is used in configuring the actual data paths. The data paths are described in **Section 3 Port Modes**.

This register is readable and writable at all times. Depending on the chosen mode/submode, reading or writing may affect the double-buffered handshake mechanism. The port B data register is not affected by the assertion of the RESET pin.

4.6.3. PORT C DATA REGISTER (PCDR). The port C data register is a holding register for moving data to and from each of the eight port C/ alternate-function pins. The exact hardware accessed is determined by the type of bus cycle (read or write) and individual conditions affecting each pin. These conditions are : 1) whether the pin is used for the port C or alternate function, and 2) whether the port C data direction register indicates the input or output direction. The port C data register is single buffered for output pins and non-latched for input pins. These conditions are summarized in table 4.3.

Table 4.3 : PCDR Hardware Accesses.

Operation	Port C Function		Alternate Function	
	PCDDR = 0	PCDDR = 1	PCDDR = 0	PCDDR = 1
Read Port C Data Register	Pin	Output Register	Pin	Output Register
Write Port C Data Register	Output Register, Buffer Disabled	Output Register, Buffer Enabled	Output Register	Output Register

Note that two additional useful benefits result from this structure. First, it is possible to directly read the state of a dual-function pin while used for the non-port C function. Second, it is possible to generate program controlled transitions on alternate-function pins by switching back to the port C function and writing to the PCDR.

This register is readable and writable at all times and operation is independent of the chosen PI/T mode. The port C data register is not affected by the assertion of the RESET pin.

4.7. PORT ALTERNATE REGISTERS

The following paragraphs describe the port alternate registers.

4.7.1. PORT A ALTERNATE REGISTER (PAAR). The port A alternate register is an alternate register for reading the port A pins. It is a read-only address and no other PI/T condition is affected. In all modes, the instantaneous pin level is read and no input latching is performed except at the data bus interface. Writes to this address are answered with DTACK, but the data is ignored.

4.7.2. PORT B ALTERNATE REGISTER (PBAR). The port B alternate register is an alternate register for reading the port B pins. It is a read-only address and no other PI/T condition is affected. In all modes, the instantaneous pin level is read and no input latching is performed except at the data bus interface. Writes to this address are answered with DTACK, but the data is ignored.

4.8. PORT STATUS REGISTER (PSR)

7	6	5	4	3	2	1	0
H4 Level	H3 Level	H2 Level	H1 Level	H4S	H3S	H2S	H1S

The port status register contains information about handshake pin activity. Bits 7-4 show the instantaneous level of the respective handshake pin, and are independent of the handshake pin sense bits in the port general control register. Bits 3-0 are the respective status bits referred to throughout this document. Their interpretation depends on the programmed mode/submode of the PI/T. For bits 3-0 a one is the active or asserted state.

4.9. TIMER CONTROL REGISTER (TCR)

7	6	5	4	3	2	1	0
TOUT/TIACK Control			Z.D Control	*	Clock Control		Timer Enable

The timer control register (TCR) determines all operations of the timer. Bits 7-5 configure the PC3/TOUT and PC7/TIACK pins for port C, square wave, vectored interrupt, or autovectored interrupt operation ; bit 4 specifies whether the counter receives data from the counter preload register or continues counting when zero detect is reached ; bit 3 is unused and is read as zero ; bits 2 and 1 configure the path from the CLK and TIN pins to the counter controller ; and bit 0 enables the timer. This register is readable and writable at all times. All bits are cleared to zero when the RESET pin is asserted.

TCR 7 6 5	**TOUT/$\overline{\text{TIACK}}$ Control**
0 0 X	The dual-function pins PC3/TOUT and PC7/$\overline{\text{TIACK}}$ carry the port C function.
0 1 X	The dual-function pin PC3/TOUT carries the TOUT function. In the run state it is used as a square-wave output and is toggled on zero detect. The TOUT pin is high while in the halt state. The dual-function pin PC7/$\overline{\text{TIACK}}$ carries the PC7 function.
1 0 0	The dual-function pin PC3/TOUT carries the TOUT function. In the run or halt state it is used as a timer interrupt request output. The timer interrupt is disabled, thus, the pin is always three stated. The dual-function pin PC7/$\overline{\text{TIACK}}$ carries the $\overline{\text{TIACK}}$ function ; however, since interrupt request is negated, the PI/T produces no response (i.e., no data or DTACK) to an asserted $\overline{\text{TIACK}}$. Refer to **5.1.3. Timer Interrupt Acknowledge Cycles** for details.
1 0 1	The dual-function pin PC3/TOUT carries the TOUT function and is used as a timer interrupt request output. The timer interrupt is enabled ; thus, the pin is low when the timer ZDS status bit is one. The dual-function pin PC7/$\overline{\text{TIACK}}$ carries the $\overline{\text{TIACK}}$ function and is used as a timer interrupt acknowledge input. Refer to the **5.1.3. Timer Interrupt Acknowledge Cycles** for details. This combination supports vectored timer interrupts.
1 1 0	The dual-function pin PC3/TOUT function. In the run or halt state it is used as a timer interrupt request output. The timer interrupt is disabled ; thus, the pin is always three-stated. The dual-function pin PC7/$\overline{\text{TIACK}}$ carries the PC7 function.
1 1 1	The dual-function pin PC3/TOUT carries the TOUT function and is used as a timer interrupt request output. The timer interrupt is enabled ; thus, the pin is low when the timer ZDS status bit is one. The dual-function pin PC7/$\overline{\text{TIACK}}$ carries the PC7 function and autovectored interrupts are supported.

TCR 4	**Zero Detect Control**
0	The counter is loaded from the counter preload register on the first clock to the 24-bit counter after zero detect, then resumes counting.
1	The counter rolls over on zero detect, then continues counting.

TCR	
3	Unused and is always read as zero.

TCR 2 1	**Clock Control**
0 0	The PC2/TIN input pin carries the port C function, and the CLK pin and prescaler are used. The prescaler is decremented on the falling transition of the CLK pin ; the 24-bit counter is decremented, rolls over, or is loaded from the counter preload registers when the prescaler rolls over from \$OO to \$1F. The timer enable bit determines whether the timer is in the run or halt state.
0 1	The PC2/TIN pin serves as a timer input, and the CLK pin and prescaler are used. The prescaler is decremented on the falling transition of the CLK pin ; the 24-bit counter is decremented, rolls over, or is loaded from the counter preload registers when the prescaler rolls over from \$00 to \$1F. The timer is in the run state when the timer enable bit is one and the TIN pin is high ; otherwise, the timer is in the halt state.
1 0	The PC2/TIN pin serves as a timer input and the prescaler is used. The prescaler is decremented following the rising transition of the TIN pin after being synchronized with the internal clock. The 24-bit counter is decremented, rolls over, or is loaded from the counter preload registers when the prescaler rolls over from \$00 to \$1F. The timer enable bit determines whether the timer is in the run or halt state.

1 1	The PC2/TIN pin serves as a timer input and the prescaler is not used. The 24-bit counter is decremented, rolls over, or is loaded from the counter preload registers following the rising edge of the TIN pin after being synchronized with the internal clock. The timer enable bit determines whether the timer is in the run or halt state.

TCR

0	**Timer Enable**
0	Disabled
1	Enabled

4.10. TIMER INTERRUPT VECTOR REGISTER (TIVR)

The timer interrupt vector register contains the 8-bit vector supplied when the timer interrupt acknowledge pin TIACK is asserted. The register is readable and writable at all times, and the same value is always obtained from a normal read cycle or a timer interrupt acknowledge bus cycle (TIACK). When the RESET pin is asserted the value of $0F is loaded into the register. Refer to **5.1.3. Timer Interrupt Acknowledge Cycles** for more details.

4.11. COUNTER PRELOAD REGISTER H, M, L (CPRH-L)

7	6	5	4	3	2	1	0	
Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	CPRH
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	CPRM
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	CPRL

The counter preload registers are a group of three 8-bit registers used for storing data to be transferred to the counter. Each of the registers is individually addressable, or the group may be accessed with the MOVEP.L or the MOVEP.W instructions. The address $12 (one less than the address of CPRH) is the null register and is reserved so that zeros are read in the upper eight bits of the destination data register when a MOVEP.L is used. Data written to this address is ignored.

These registers are readable and writable at all times. A read cycle proceeds independently of any transfer to the counter, which may be occurring simultaneously. To insure proper operation of the PI/T timer, a value of $000000 may not be stored in the counter preload registers for use with the counter. The RESET pin does not affect the contents of these registers.

4.12. COUNT REGISTER H, M, L (CNTRH-L)

7	6	5	4	3	2	1	0	
Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	CNTRH
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	CPRM
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	CPRL

The count registers are a group of three 8-bit addresses at which the counter can be read. The contents of the counter are not latched during a read bus cycle ; thus, the data read at these addresses is not guaranteed if the timer is in the run state. Write operations to these addresses result in a normal bus cycle but the data is ignored.

Each of the registers is individually addressable, or the group may be accessed with the MOVEP.L or the MOVEP.W instructions. The address, one less than the address CNTRH, is the null register and is reserved so that zeros are read in the upper eight bits of the destination data register when a MOVEP.L is used. Data written to this address is ignored.

4.13. TIMER STATUS REGISTER (TSR)

7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	ZDS

The timer status register contains one bit from which the zero detect status can be determined. The ZDS status bit (bit 0) is an edge-sensitive flip-flop that is set to one when the 24-bit counter decrements from $000001 to $000000. The ZDS status bit is cleared to zero following the direct reset operation or when the timer is halted. Note that when the RESET pin is asserted the timer is disabled, and thus enters the halt state.

This register is always readable without consequence. A write access performs a direct reset operation if bit 0 in the written data is one. Following that, the ZDS bit is zero.

This register is constructed with a reset dominant S-R flip-flop so that all clearing conditions prevail over the possible zero detect condition.

Bits 7-1 are unused and are read as zero.

4.14. REGISTER VALUE AFTER RESET

Table 1.3, located at the end of this document, shows the values that remain or are changed after a reset. Note that interrupt vector registers are initialized to $0F. For the port interrupt vector register, the only time that bits 0 and 1 are set is after reset.

SECTION 5

TIMER OPERATION AND APPLICATIONS SUMMARY

This section describes the programmable options available, capabilities, and restrictions that apply to the timer. Programming of the timer control register is outlined with several examples given.

5.1. TIMER OPERATION

The TS68230 timer can provide several facilities needed by TS68000 operating systems. It can generate periodic interrupts, a square wave, or a single interrupt after a programmed time period. Also, it can be used for elapsed time measurement or as a device watchdog.

The PI/T timer contains a 24-bit synchronous down counter that is loaded from three 8-bit counter preload registers. The 24-bit counter may be clocked by the output of a 5-bit (divide-by-32) prescaler or by an external timer input (TIN). If the prescaler is used, it may be clocked by the system clock (CLK pin) or by the TIN external input. The counter signals the occurrence of an event primarily through zero detection. (A zero is when the counter of the 24-bit timer is equal to zero). This sets the zero detect status (ZDS) bit in the timer status register. It may be checked by the processor or may be used to generate a timer interrupt. The ZDS bit can be reset by writing a one to the timer status register in that bit position independent of timer operation.

The general operation of the timer is flexible and easily programmable. The timer is fully configured and controlled by programming the 8-bit timer control register (refer to **4.9 Timer Control Register** (TCR) for additional information). It controls : 1) the choice between the port C operation and the timer operation of three timer pins, 2) whether the counter is loaded from the counter preload register or rolls over when zero detect is reached, 3) the clock input, 4) whether the prescaler is used, and 5) whether the timer is enabled.

5.1.1. RUN/HALT DEFINITION. The overall operation of the timer is described in terms of the run or halt states. The control of the current state is determined by programming the timer control register. When in the halt state, all of the following occur :

1. The prior content of the counter is not altered and is reliably readable via the count registers.
2. The prescaler is forced to \$1F whether or not it is used.
3. The ZDS status bit is forced to zero, regardless of the possible zero contents of the 24-bit counter.

The run state is characterized by :

1. The counter is clocked by the source programmed in the timer control register.
2. The counter is not reliably readable.
3. The prescaler is allowed to decrement if programmed for use.
4. The ZDS status bit is set when the 24-bit counter transitions from \$000001 to \$000000.

5.1.2. TIMER RULES. The following is a set of rules that allow easy application of the timer.

1. Refer to 5.1.1. Run/Halt Definition.
2. When the $\overline{\text{RESET}}$ pin is asserted, all bits of the timer control register are cleared, configuring the dual function pins as port C inputs.
3. The contents of the counter preload registers and counter are not affected by the $\overline{\text{RESET}}$ pin.
4. The count registers provide a direct read data path from each portion of the 24-bit counter, but data written to their addresses is ignored. (This results in a normal bus cycle). These registers are readable at any time, but their contents are never latched. Unreliable data may be read when the timer is in the run state.
5. The counter preload registers are readable and writable at any time and this occurs independently of any timer operation. No protection mechanisms are provided against ill-timed writes.
6. The input frequency to the 24-bit counter from the TIN pin or prescaler output must be between zero and the input frequency at the CLK pin divided by eight, regardless of the configuration chosen.
7. For configurations in which the prescaler is used (with the CLK pin or TIN pin as an input), the contents of the counter preload register (CPR) is transferred to the counter the first time that the prescaler passes from \$00 to \$1F (rolls over) after entering the run state. Thereafter, the counter decrements, rolls over, or is loaded from the counter preload register each time the prescaler rolls over.

8. For configurations in which the prescaler is not used, the contents of the counter preload registers are transferred to the counter on the first asserted edge of the TIN input after entering the run state. On subsequent asserted edges the counter decrements, rolls over, or is loaded from the counter preload registers.
9. The smallest value allowed in the counter preload register for use with the counter is $000001.

5.1.3. TIMER INTERRUPT ACKNOWLEDGE CYCLES. Several conditions may be present when the timer interrupt acknowledge pin (TIACK) is asserted. These conditions affect the PI/T's response and the termination of the bus cycle (see table 5.1).

5.2. TIMER APPLICATIONS SUMMARY

The following paragraphs outline programming of the timer control register for several typical examples.

5.2.1.PERIODIC INTERRUPT GENERATOR EXAMPLE.

7	6	5	4	3	2	1	0
TOUT/$\overline{\text{TIACK}}$ Control			Z.D Control	*	Clock Control		Timer Enable
1	x	1	0	0	00 or 1X		Changed

In this configuration the timer generates a periodic interrupt. The TOUT pin is connected to the system's interrupt request circuitry and the TIACK pin may be used as an interrupt acknowledge input to the timer. The TIN pin may be used as a clock input.

The processor loads the counter preload registers (CPR) and timer control register (TCR), and then enables the timer. When the 24-bit counter passes from $000001 to $000000, the ZDS status bit is set and the TOUT (interrupt request) pin is asserted. At the next clock to the 24-bit counter, it is again loaded with the contents of the CPRs and thereafter decrements. In normal operation, the processor must direct clear the status bit to negate the interrupt request (see figure 5.1).

5.2.2. SQUARE WAVE GENERATOR.

7	6	5	4	3	2	1	0
TOUT/$\overline{\text{TIACK}}$ Control			Z.D. Control	*	Clock Control		Timer Enable
1	x	1	1	0	00 or 1X		Changed

In this configuration the timer produces a square wave at the TOUT pin. The TOUT pin is connected to the user's circuitry and the TIACK pin is not used. The TIN pin may be used as a clock input.

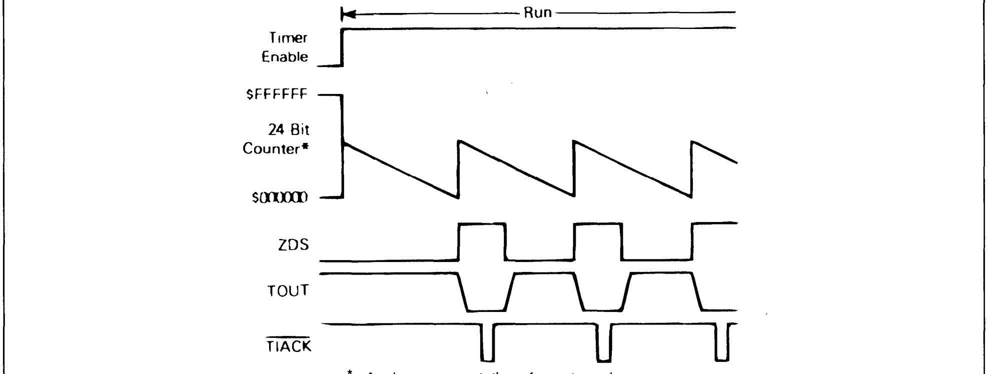
The processor loads the counter preload registers and timer control register, and then enables the timer. When the 24-bit counter passes form $000001 to $000000 the ZDS status bit is set and the TOUT (square wave output) pin is toggled. At the next clock to the 24-bit counter it is again loaded with the contents of the CPRs, and thereafter decrements. In this application there is no need for the processor to direct clear the ZDS status bit ; however, it is possible for the processor to sync itself with the square wave by clearing the ZDS status bit, then polling it. The processor may also read the TOUT level at the port C address.

Note that the PC3/TOUT pin functions as PC3 following the negation of $\overline{\text{RESET}}$. If used in the square wave configuration, a pullup resistor may be required to keep a known level prior to programming. Prior to enabling the timer, TOUT is high (see figure 5.2).

Table 5.1 : Response to Timer Interrupt Acknowledge

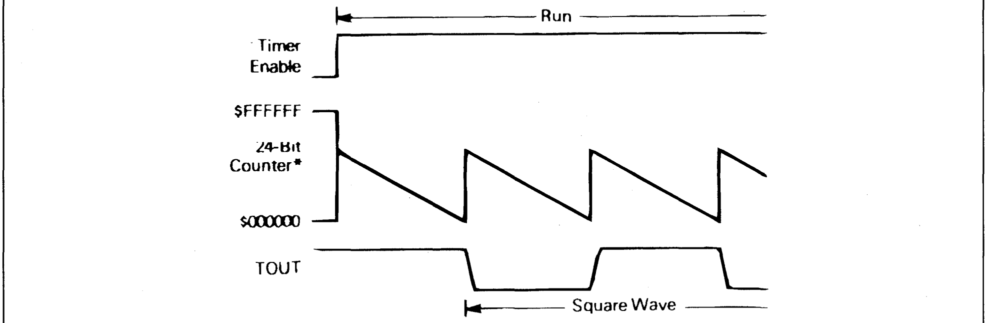
PC3/TOUT Function	Response to Asserted $\overline{\text{TIACK}}$
PC3 - Port C Pin	No Response No $\overline{\text{DTACK}}$
TOUT - Square Wave	No Response No $\overline{\text{DTACK}}$
TOUT - Negated Timer Interrupt Request	No Response No $\overline{\text{DTACK}}$
TOUT - Asserted Timer Interrupt Request	Timer Interrupt Vector Contents $\overline{\text{DTACK}}$ Asserted

Figure 5.1 :Periodic Interrupt Generator Example.



* Analog representation of counter value.

Figure 5.2 : Square Wave Generator Example.



* Analog representation of counter value.

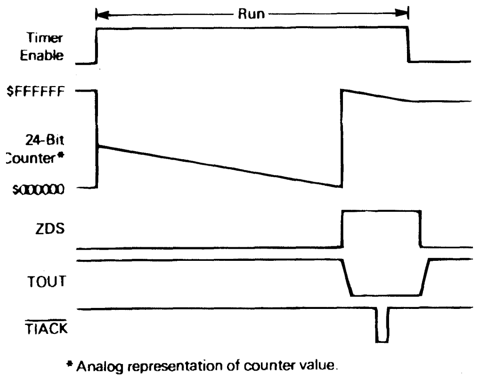
5.2.3. INTERRUPT AFTER TIMEOUT.

7	6	5	4	3	2	1	0
TOUT/$\overline{\text{TIACK}}$ Control			Z.D Control	*	Clock Control		Timer Enable
0	1	x	0	0	00 or 1X		Changed

In this configuration the timer generates an interrupt after a programmed time period has expired. The TOUT pin is connected to the system's interrupt request circuitry and the $\overline{\text{TIACK}}$ pin may be an interrupt acknowledge input to the timer. The TIN pin may be used as a clock input.

This configuration is similar to the periodic interrupt generator except that the zero detect control bit is set. This forces the counter to roll over after zero detect is reached, rather than reloading from the CPRs. When the processor takes the interrupt it can halt the timer, read the counter and calculate the time from the interrupt request to entering the service routine. Accurate knowledge of the interrupt latency may be useful in some applications (see figure 5.3).

Figure 5.3 : Single Interrupt after Timeout Example.



5.2.4. ELAPSED TIME MEASUREMENT EXAMPLES. Elapsed time measurement takes several forms ; two forms are described in the following paragraphs.

5.2.4.1. System Clock Example.

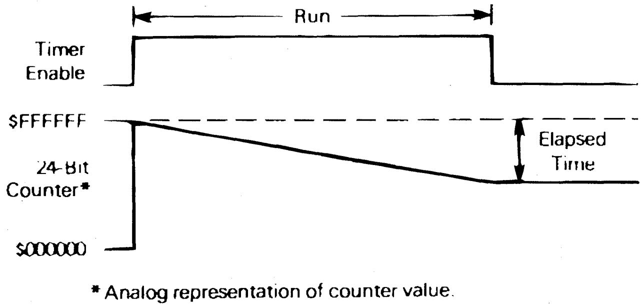
7 6 5	4	3	2 1	0
TOUT/$\overline{\text{TIACK}}$ Control	Z.D Control	*	Clock Control	Timer Enable
0 0 X	1	0	0 0	Changed

This configuration allows time interval measurement by software. The TIN pin may be used as an external timer enable if desired.

The processor loads the counter preload registers (generally with all ones), loads the timer control register, and then enables the timer. The counter is allowed to decrement until the ending event takes place. When it is desired to read the time interval, the processor must halt the timer and then read the counter. If TIN is used as an enable, the start and stop counter functions are controlled externally.

For applications in which the interval may exceed the programmed time interval, zero detection can be counted by polling the status register or through interrupts to simulate additional timer bits. Note that the ZDS bit is latched and should be cleared after each detection of zero. At the end, the timer can be halted and read (see figure 5.4).

Figure 5.4 : Elapsed Time Measurement Example.



5.2.4.2. External Clock.

7 6 5	4	3	2 1	0
TOUT/$\overline{\text{TIACK}}$ Control	Z.D Control	*	Clock Control	Timer Enable
0 0 X	1	0	1 X	Changed

This configuration allows measurement (counting) of the number of input pulses occurring in an interval in which the counter is enabled. The TIN input pin provides the input pulses. Generally the TOUT and TIACK pins are not used.

This configuration is similar to the elapsed time measurement/system clock configuration except that the TIN pin is used to provide the input frequency. It can be connected to a simple oscillator and the same methods could be used. Alternately, it could be gated off and on externally and the number of cycles occurring while in the run state can be counted. However, minimum pulse width high and low specifications must be met.

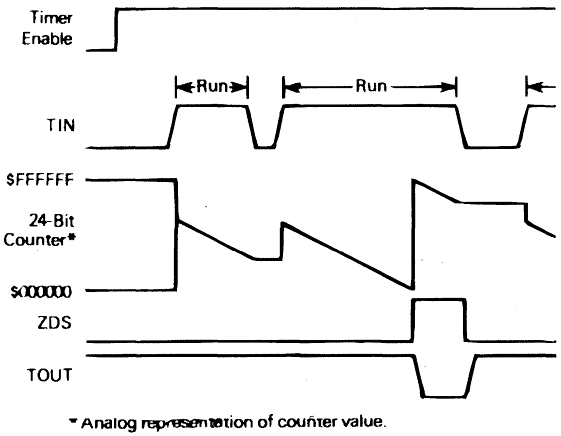
5.2.5. DEVICE WATCHDOG.

7 6 5	4	3	2 1	0
TOUT/$\overline{\text{TIACK}}$ Control	Z.D Control	*	Clock Control	Timer Enable
1 X 1	1	0	0 1	Changed

This configuration provides the watchdog function needed in many systems. The TIN pin is the timer input whose period at the high (one) level is to be checked. Once allowed by the processor, the TIN input pin controls the run/halt mode. The TOUT pin is connected to external circuitry requiring notification when the TIN pin has been asserted longer than the programmed time. The TIACK pin (timer interrupt acknowledge) is only needed if the TOUT pin is connected to the interrupt circuitry.

The processor loads the counter preload register and timer control register, and then enables the timer. When the TIN input is asserted (one, high) the timer transfers the contents of the counter preload register to the counter and begins counting. If the TIN input is negated before zero detect is reached, the TOUT output and the ZDS status bit remain negated. If zero detect is reached while the TIN input is still asserted, the ZDS status bit is set and the TOUT output is asserted. (The counter rolls over and keeps counting). In either case, when the TIN input is negated the ZDS status bit is zero, the TOUT output is negated, the counting stops, and the prescaler is forced to all ones (see figure 5.5).

Figure 5.5 : Device Watchdog Example.



SECTION 6

ELECTRICAL SPECIFICATIONS

This section contains electrical specifications and associated timing information for the TS68230.

6.1 ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	– 0.3 to + 7.0	V
V_{IN}	Input Voltage	– 0.3 to + 7.0	V
T_A	Operating Temperature Range TS68230C TS68230V	T_L to T_H 0 to + 70 – 40 to + 85	°C
T_{stg}	Storage Temperature	– 55 to + 150	°C

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}).

6.2 THERMAL DATA

θ_{JA}	Thermal Resistance Plastic	50	°C/W

6.3. POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from :

$T_J = T_A + (P_D \bullet \theta_{JA})$

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 :

$P_D = K \ (T_J + 273°C)$

Solving equations 1 and 2 for K gives :

$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2$

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.

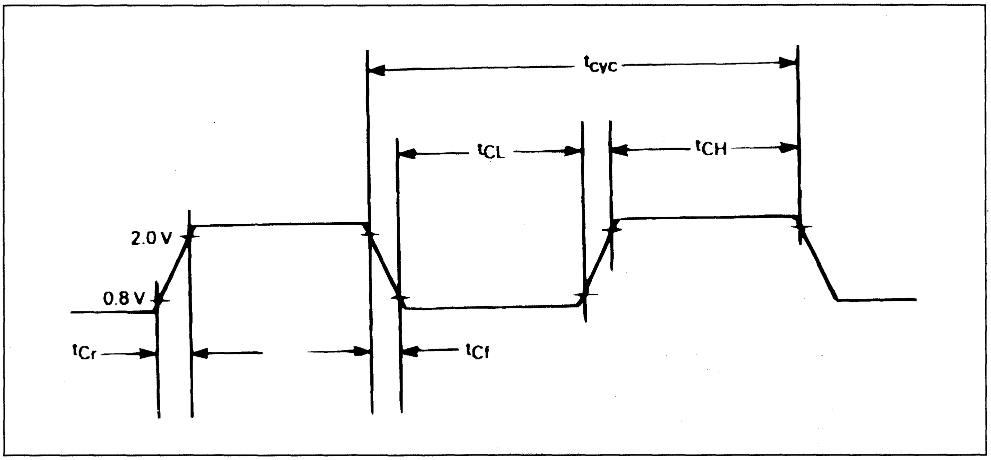
6.4 DC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0Vdc ± 5%, T_A = T_L to T_H unless otherwise noted)

Symbol	Parameter		Min.	Max.	Unit
V_{IH}	Input High Voltage	All Inputs	V_{SS} + 2.0	V_{CC}	V
V_{IL}	Input Low Voltage	All Inputs	V_{SS} – 0.3	V_{SS} + 0.8	V
I_{IN}	Input Leakage Current (V_{IN} = 0 to 5.25V)	H1, H3, R/$\overline{W}$, $\overline{RESET}$, CLK, RS1-RS5, $\overline{CS}$		10.0	µA
I_{TSI}	Hi-Z Input Current (V_{IN} = 0.4 to 2.4)	D0-D7 $\overline{DTACK}$, PC0-PC7, H2, H4, PA0-PA7, PB0-PB7	 – 0.1	20 – 1.0	µA mA
V_{OH}	Output High Voltage (I_{Load} = – 400µA, V_{CC} = min) (I_{Load} = – 150µA, V_{CC} = min) (I_{Load} = – 100µA, V_{CC} = min)	 $\overline{DTACK}$, D0-D7 H2, H4, PB0-PB7, PA0-PA7 PC0-PC7	V_{SS} + 2.4		V
V_{OL}	Output Low Voltage (I_{Load} = 8.8mA, V_{CC} = min) (I_{Load} = 5.3mA, V_{CC} = min) (I_{Load} = 2.4mA, V_{CC} = min)	 PC3/TOUT, PC5/$\overline{PIRQ}$ D0-D7, $\overline{DTACK}$ PA0-PA7, PB0-PB7, H2, H4, PC0-PC2, PC4, PC6, PC7		0.5	V
P_{INT}	Internal Power Dissipation (measured at T_A = 0°C)			750	mW
C_{IN}	Input Capacitance (V_{in} = 0, T_A = 25°C, f = 1MHz)			15	pF

6.5 AC ELECTRICAL SPECIFICATIONS – CLOCK TIMING (see figure 6.1)

Symbol	Parameter	8 MHz		10 MHz		Unit
		Min.	Max.	Min.	Max.	
f	Frequency of Operation	2.0	8.0	2.0	10.0	MHz
t_{cyc}	Cycle Time	125	500	100	500	ns
t_{CL} t_{CH}	Clock Pulse Width	55 55	250 250	45 45	250 250	ns
t_{Cr} t_{Cf}	Rise and Fall Times		10 10		10 10	ns

Figure 6.1 : Clock Input Timing Diagram.



6.6. AC ELECTRICAL SPECIFICATIONS

(V_{CC} = 5.0Vdc ± 5%, V_{SS} = 0Vdc, T_A = T_L to T_H unless otherwise specified)

Read and Write Cycle Timings (figures 6.2 and 6.3)

Number	Parameter	8MHz		10MHz		Unit
		Min.	Max.	Min.	Max.	
1	R/$\overline{W}$, RS1-RS5 Valid to CS Low (setup time)	0		0		ns
2(1)	$\overline{CS}$ Low to R/$\overline{W}$ and RS1-RS5 Invalid (hold time)	100		65		ns
3(2)	$\overline{CS}$ Low to CLK Low (setup time)	30		20		ns
4(3)	$\overline{CS}$ Low to Data Out Valid		75		60	ns
5	RS1-RS5 Valid to Data Out Valid		140		100	ns
6	CLK Low to $\overline{DTACK}$ Low (read/write cycle)	0	70	0	60	ns
7(4)	$\overline{DTACK}$ Low to $\overline{CS}$ High (hold time)	0		0		ns
8	$\overline{CS}$ or $\overline{PIACK}$ or $\overline{TIACK}$ High to Data Out Invalid (hold time)	0		0		ns
9	$\overline{CS}$ or $\overline{PIACK}$ or $\overline{TIACK}$ High to D0-D7 High Impedance		50		45	ns
10	$\overline{CS}$ or $\overline{PIACK}$ or $\overline{TIACK}$ High to $\overline{DTACK}$ High		50		45	ns
11	$\overline{CS}$ or $\overline{PIACK}$ or $\overline{TIACK}$ High to $\overline{DTACK}$ High Impedance		100		55	ns
12	Data In Valid to $\overline{CS}$ Low (setup time)	0		0		ns
13	$\overline{CS}$ Low to Data in Invalid (hold time)	100		65		ns
23	CLK low on which $\overline{DMAREQ}$ is asserted to CLK low on which $\overline{DMAREQ}$ is negated	2.5	3	2.5	3	CLK Per
28	Read Data Valid to $\overline{DTACK}$ Low (setup time)	0		0		ns
32(5)	Synchronized $\overline{CS}$ to CLK low on which $\overline{DMAREQ}$ is asserted	3	3	3	3	CLK Per
35	CLK Low to $\overline{DMAREQ}$ Low (delay time)	0	120	0	100	ns
36	CLK Low to $\overline{DMAREQ}$ High (delay time)	0	120	0	100	ns
37(5)	Synchronized H1(H3) to CLK low on which $\overline{PIRQ}$ is asserted	3	3	3	3	CLK Per
38(5)	Synchronized $\overline{CS}$ to CLK low on which $\overline{PIRQ}$ is high impedance	3	3	3	3	CLK Per
39	CLK Low to PIRQ Low or High Impedance	0	250	0	225	ns
40(6)	TIN Frequency (external clock) - Prescaler used.	0	1	0	1	f_{clk} (Hz)(7)
41	TIN Frequency (external clock) - Prescaler not used.	0	1/8	0	1/8	f_{clk} (Hz)(7)
42	TIN Pulse Width High or Low (external clock)	55		45		ns
43	TIN Pulse Width Low (run/halt control)	1		1		CLK Per
44	CLK Low to TOUT High, Low, or High Impedance	0	250	0	225	ns
45	$\overline{CS}$, $\overline{PIACK}$, or $\overline{TIACK}$ High to $\overline{CS}$, $\overline{PIACK}$, or $\overline{TIACK}$ Low	50		30		ns

Notes :
1. See **1.4. Bus Interface Operation** for exception.
2. This specification only applies if the PI/T had completed all operations initiated by the previous bus cycle when $\overline{CS}$ was asserted. Following a normal read or write bus cycle, all operations are complete within three clocks after the falling edge of the CLK pin on which DTACK was asserted. If CS is asserted prior to completion of these operations, the new bus cycle, and hence, DTACK is postponed.
   If all operations of the previous bus cycle were complete when $\overline{CS}$ was asserted, this specification is made only to insure that DTACK is asserted with respect to the falling edge of the CLK pin as shown in the timing diagram, not to guarantee operation of the part. If the CS setup time is violated, DTACK may be asserted as shown, or may be asserted one clock cycle later.
3. Assuming the RS1-RS5 to data valid time has also expired.
4. This specification imposes a lower bound on $\overline{CS}$ low time, guaranteeing that $\overline{CS}$ will be low for at least 1 CLK period.
5. Synchronized means that the input signal has seen seen by the PI/T on the appropriate edge of the clock (rising edge for H1(H3) and falling edge for $\overline{CS}$). (Refer to the 1.4. **Bus Interface Operation** for the exception concerning $\overline{CS}$).
6. This limit applies to the frequency of the signal at TIN compared to the frequency of the CLK signal during each clock cycle. If any period of the waveform at TIN is smaller than the period of the CLK signal at that instant, then it is likely that the timer circuit will completely ignore one cycle of the TIN signal.
   If these two signals are derived from different sources they will have different instantaneous frequency variations. In this case the frequency applied to the TIN pin must be distinctly less than the frequency at the CLK pin to avoid lost cycles of the TIN signal. With signals derived from different crystal oscillators applied to the TIN and CLK pins with fast rise and fall times, the TIN frequency can approach 80 to 90% of the frequency of the CLK signal without a loss of a cycle of the TIN signal.
   If these signals are derived from the same frequency source then the frequency of the signal applied to TIN can be 100% of the frequency at the CLK pin. They may be generated by different buffers from the same signal or one may be an inverted version of the other. The TIN signal may be generated by an 'AND' function of the clock and a control signal.
7. CLK refers to the actual frequency of the CLK pin, not the maximum allowable CLK frequency.

Figure 6.2 : Read Cycle Timing Diagram.

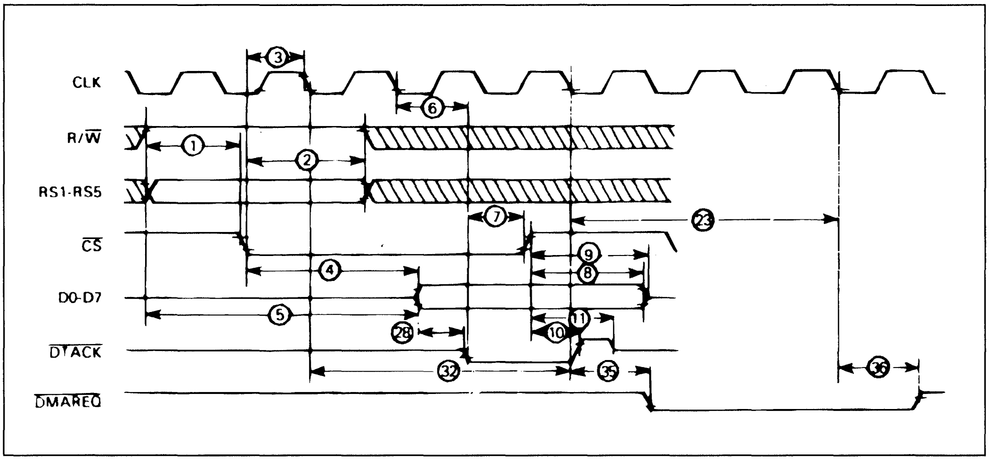
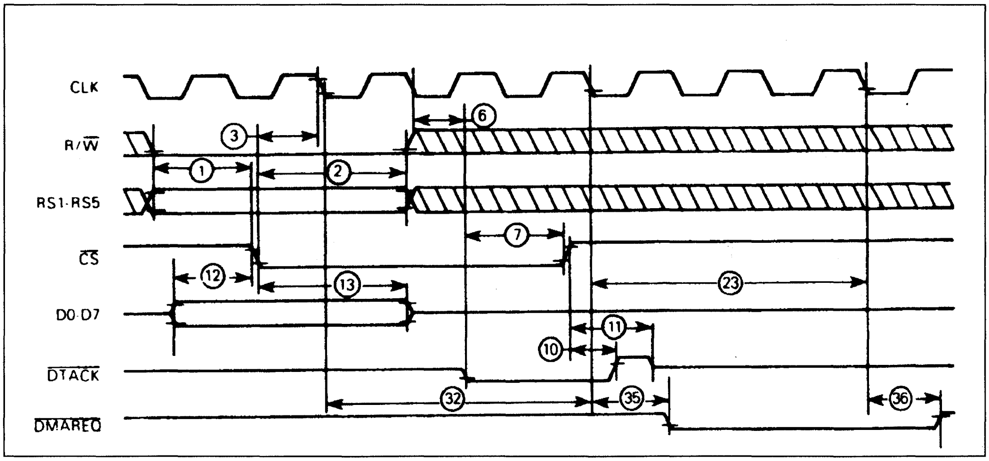


Figure 6.3 : Write Cycle Timing Diagram.



Note : Timing measurements are referenced to and from a low voltage of 0.8volt and a high voltage of 2.0volts, unless otherwise noted.

6.6. AC ELECTRICAL SPECIFICATIONS

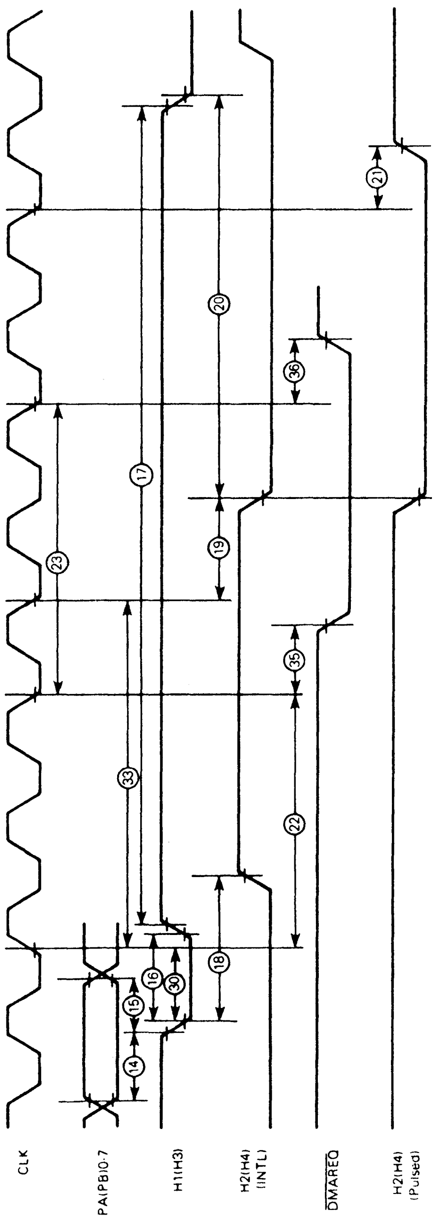
(V_{CC} = 5.0Vdc ± 5%, V_{SS} = 0Vdc, T_A = T_L to T_H unless otherwise noted)

Peripheral Input Timings (figures 6.4)

Number	Parameter	8MHz		10MHz		Unit
		Min.	Max.	Min.	Max.	
14	Port Input Data Valid to H1(H3) Asserted (setup time)	100		60		ns
15	H1(H3) Asserted to Port Input Data Invalid (hold time)	20		20		ns
16	Handshake Input H1(H4) Pulse Width Asserted	40		40		ns
17	Handshake Input H1(H4) Pulse Width Negated	40		40		ns
18	H1(H3) Asserted to H2(H4) Negated (delay time)		150		120	ns
19	CLK Low to H2(H4) Asserted (delay time)		100		100	ns
20(1)	H2(H4) Asserted to H1(H3) Asserted	0		0		ns
21(2)	CLK Low to H2(H4) Pulse Negated (delay time)		125		125	ns
22(3.4)	Synchronized H1(H3) to CLK low on which $\overline{\text{DMAREQ}}$ is asserted	2.5	3.5	2.5	3.5	CLK Per
23	CLK low on which $\overline{\text{DMAREQ}}$ is asserted to CLK low on which $\overline{\text{DMAREQ}}$ is negated	2.5	3	2.5	3	CLK Per
30(5)	H1(H3) Asserted to CLK High (setup time)	50		40		ns
33(3.4)	Synchronized H1(H3) to CLK low on which H2(H4) is asserted	3.5	4.5	3.5	4.5	CLK Per
35	CLK Low to $\overline{\text{DMAREQ}}$ Low (delay time)	0	120	0	100	ns
36	CLK Low to $\overline{\text{DMAREQ}}$ High (delay time)	0	120	0	100	ns

Notes :
1. This specification assures recognition of the asserted edge of H1(H3).
2. This specification applies only when a pulsed handshake option is chosen and the pulse is not shortened due to an early asserted edge of H1(H3).
3. The maximum value is caused by a peripheral access (H1(H3) asserted) and bus access ($\overline{\text{CS}}$ asserted) occurring at the same time.
4. Syncrhonized means that the input signal has been seen by the PI/T on the appropriate edge of the clock (rising edge for H1(H3) and falling edge for $\overline{\text{CS}}$). (Refer to the **1.4 Bus Interface Operation** for the exception concerning $\overline{\text{CS}}$).
5. If the setup time on the rising edge of the clock is not met, H1(H3) may not be recognized until the next rising of the clock.

Figure 6.4 : Peripheral Input Timing Diagram.



Notes : 1. Timing diagram shows H1, H2, H3, and H4 asserted low.
2. Timing measurements are referenced to and from a low voltage of 0.8volt and a high voltage of 2.0volts, unless otherwise noted.

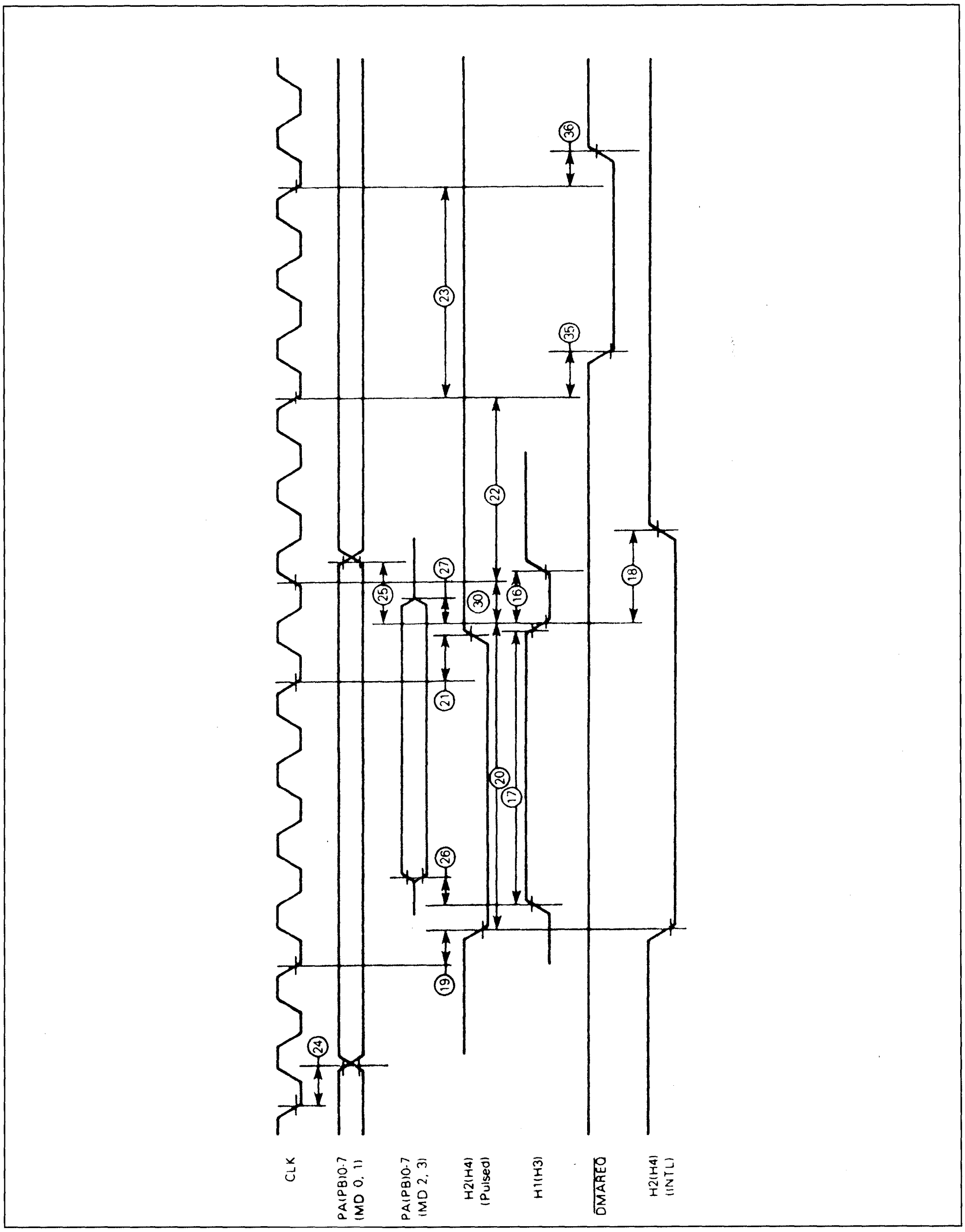
6.6. AC ELECTRICAL SPECIFICATIONS
(V_{CC} = 5.0Vdc ± 5%, V_{SS} = 0Vdc, T_A = T_L to T_H unless otherwise noted)

Peripheral Output Timings (figures 6.5)

Number	Parameter	8MHz		10MHz		Unit
		Min.	Max.	Min.	Max.	
16	Handshake Input H1(H4) Pulse Width Asserted	40		40		ns
17	Handshake Input H1(H4) Pulse Width Negated	40		40		ns
18	H1(H3) Asserted to H2(H4) Negated (delay time)		150		120	ns
19	CLK Low to H2(H4) Asserted (delay time)		100		100	ns
20(1)	H2(H4) Asserted to H1(H3) Asserted	0		0		ns
21(2)	CLK Low to H2(H4) Pulse Negated (delay time)		125		125	ns
22(3.4)	Synchronized H1(H3) to CLK low on which $\overline{\text{DMAREQ}}$ is asserted	2.5	3.5	2.5	3.5	CLK Per
23	CLK low on which $\overline{\text{DMAREQ}}$ is asserted to CLK low on which DMAREQ is negated	2.5	3	2.5	3	CLK Per
24	CLK Low to Port Output Data Valid (delay time) (modes 0 and 1)		150		120	ns
25(3.4)	Synchronized H1(H3) to Port Output Data Invalid (modes 0 and 1)	1.5	2.5	1.5	2.5	CLK Per
26	H1 Negated to Port Output Data Valid (modes 2 and 3)		70		50	ns
27	H1 Asserted to Port Output Data High Impedance (modes 2 and 3)	0	70	0	70	ns
30(5)	H1(H3) Asserted to CLK High (setup time)	50		40		ns
35	CLK Low to $\overline{\text{DMAREQ}}$ Low (delay time)	0	120	0	100	ns
36	CLK Low to $\overline{\text{DMAREQ}}$ High (delay time)	0	120	0	100	ns

Notes :
1. This specification assures recognition of the asserted edge of H1(H3).
2. This specification applies only when a pulsed handshake option is chosen and the pulse is not shortened due to an early asserted edge of H1(H3).
3. The maximum value is caused by a peripheral access (H1(H3) asserted) and bus access ($\overline{\text{CS}}$ asserted) occurring at the same time.
4. Synchronized means that the input signal has been seen by the PI/T on the appropriate edge of the clock (rising edge for H1(H3) and falling edge for $\overline{\text{CS}}$). (Refer to the **1.4. Bus Interface Operation** for the exception concerning $\overline{\text{CS}}$).
5. If the setup time on the rising edge of the clock is not met, H1(H3) may not be recognized until the next rising of the clock.

Figure 6.5 : Peripheral Ouput Timing Diagram.



Notes : 1. Timing diagram shows H1, H2, H3, and H4 asserted low.
2. Timing measurements are referenced to and from a low voltage of 0.8volts and a high voltage of 2.0volts, unless otherwise noted.

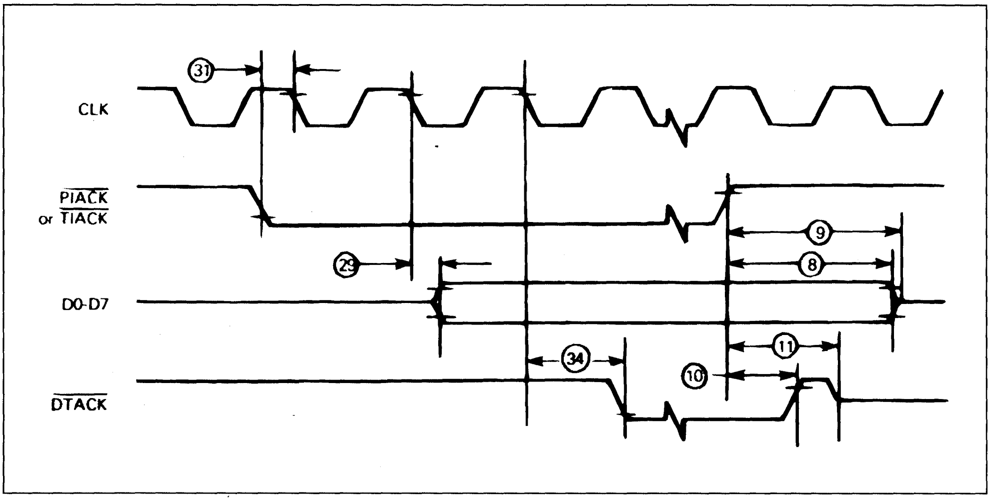
6.6. AC ELECTRICAL SPECIFICATIONS

(V_{CC} = 5.0Vdc ± 5%, V_{SS} = 0Vdc, T_A = T_L to T_H unless otherwise noted)

Iack Timings (figure 6.6)

Number	Parameter	8MHz		10MHz		Unit
		Min.	Max.	Min.	Max.	
8	$\overline{\text{CS}}$ or $\overline{\text{PIACK}}$ or $\overline{\text{TIACK}}$ High to Data Out Invalid (hold time)	0		0		ns
9	$\overline{\text{CS}}$ or $\overline{\text{PIACK}}$ or $\overline{\text{TIACK}}$ High to D0-D7 High Impedance		50		45	ns
10	$\overline{\text{CS}}$ or $\overline{\text{PIACK}}$ or $\overline{\text{TIACK}}$ High to $\overline{\text{DTACK}}$ High		50		45	ns
11	$\overline{\text{CS}}$ or $\overline{\text{PIACK}}$ or $\overline{\text{TIACK}}$ High to $\overline{\text{DTACK}}$ High Impedance		100		55	ns
29	CLK Low to Data Output Valid, Interrupt Acknowledge Cycle		120		100	ns
31	$\overline{\text{PIACK}}$ or $\overline{\text{TIACK}}$ Low to CLK Low (setup time)	50		40		ns
34	CLK Low to $\overline{\text{DTACK}}$ Low Interrupt Acknowledge Cycle (delay time)		100		100	ns

Figure 6.6 : IACK Timing Diagram.



Note : Timing measurements are referenced to and from a low voltage of 0.8volt and a high voltage of 2.0volts, unless otherwise noted.

SECTION 7

ORDERING INFORMATION

7.1. STANDARD VERSIONS

Part Number	Frequency (MHz)	Temperature Range	Package Type
TS68230CP8 TS68230CP10	8.0 10.0	0°C to + 70°C 0°C to + 70°C	Plastic DIL P Suffix
TS68230CFN8 TS68230CFN10	8.0 10.0	0°C to + 70°C 0°C to + 70°C	PLCC FN Suffix

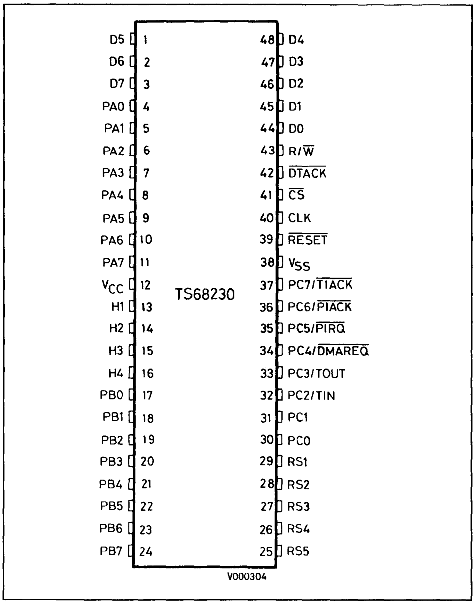
SECTION 8

MECHANICAL DATA

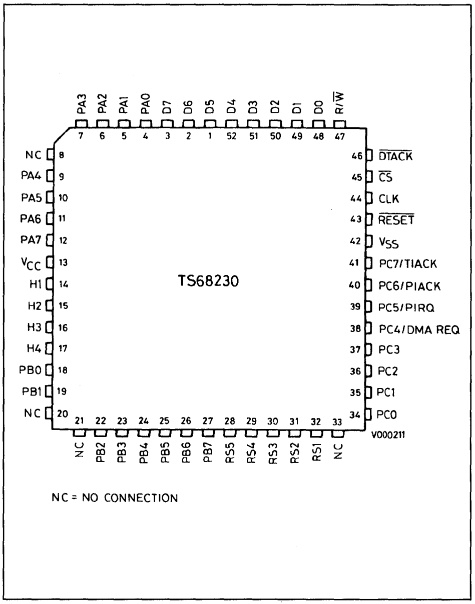
This section contains the pin assignments and package dimensions of the TS68230. In addition, detailed information is provided to be used as a guide when ordering.

8.1. PIN ASSIGNMENTS

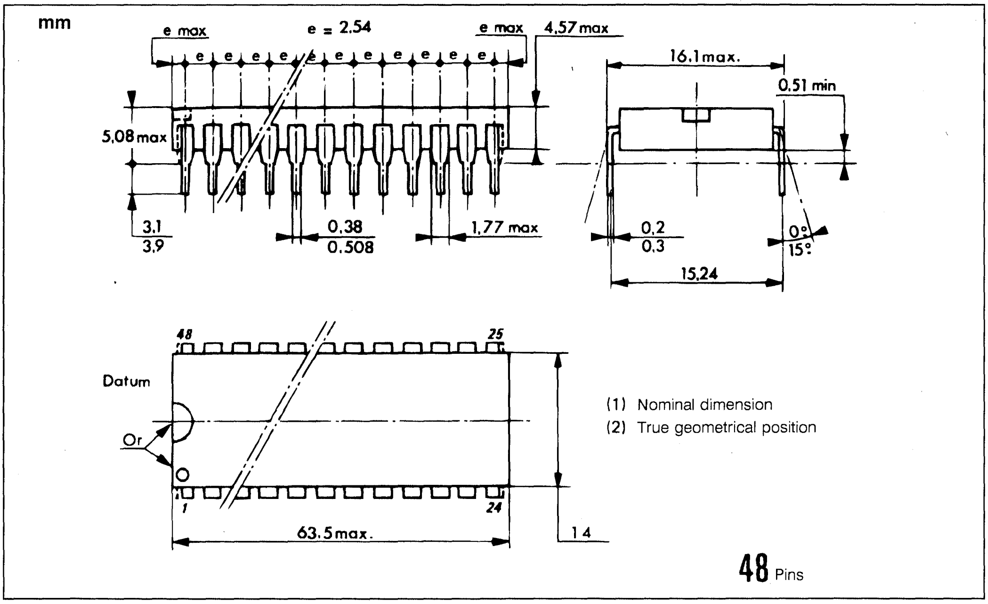
48-Pin Dual-in-Line

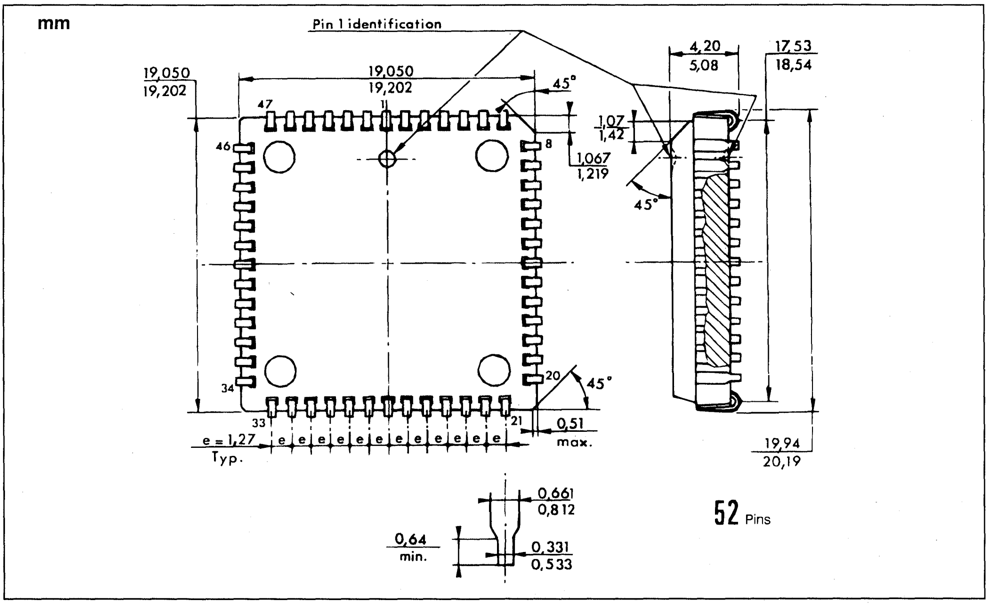


52-Pin Quad Pack (PLCC)



8.2. PACKAGE MECHANICAL DATA





APPENDIX

Table 1.3 : Register Model (sheet 1 of 2).

Register Select Bits									Register Value after $\overline{\text{RESET}}$ (hex value)	
5 4 3 2 1	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**		
0 0 0 0 0	Port Mode Control		H34 Enable	H12 Enable	H4 Sense	H3 Sense	H2 Sense	H1 Sense	0 0	Port General Control Register
0 0 0 0 1	*	SVCRQ Select		IPF Select		Port Interrupt Priority Control			0 0	Port Service Request Register
0 0 0 1 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	0 0	Port A Data Direction Register
0 0 0 1 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	0 0	Port B Data Direction Register
0 0 1 0 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	0 0	Port C Data Direction Register
0 0 1 0 1	Interrupt Vector Number						*	*	0 F	Port Interrupt Vector Register
0 0 1 1 0	Port A Submode		H2 Control			H2 Int Enable	H1 SVCRQ Enable	H1 Stat Ctrl	0 0	Port A Control Register
0 0 1 1 1	Port B Submode		H4 Control			H4 Int Enable	H3 SVCRQ Enable	H3 Stat Ctrl	0 0	Port B Control Register
0 1 0 0 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	**	Port A Data Register
0 1 0 0 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	**	Port B Data Register
0 1 0 1 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	***	Port A Alternate Register
0 1 0 1 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	***	Port B Alternate Register
0 1 1 0 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	****	Port C Data Register
0 1 1 0 1	H4 Level	H3 Level	H2 Level	H1 Level	H4S	H3S	H2S	H1S	****	Port Status Register
0 1 1 1 0	*	*	*	*	*	*	*	*	0 0	(null)
0 1 1 1 1	*	*	*	*	*	*	*	*	0 0	(null)

* Unused, read as zero.
** Value before $\overline{\text{RESET}}$.
*** Current value on pins.
**** Undetermined value.

APPENDIX

Table 1.3 : Register Model (sheet 2 of 2).

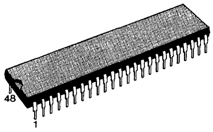
Register Select Bits 5 4 3 2 1	7	6	5	4	3	2	1	0	Register Value after RESET (hex value)	
1 0 0 0 0	TOUT/TIACK Control			Z D Ctrl	*	Clock Control		Timer Enable	0 0	Timer Control Register
1 0 0 0 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	0 F	Timer Interrupt Vector Register
1 0 0 1 0	*	*	*	*	*	*	*	*	0 0	(null)
1 0 0 1 1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	**	Counter Preload Register (high)
1 0 1 0 0	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	**	Counter Preload Register (mid)
1 0 1 0 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	**	Counter Preload Register (low)
1 0 1 1 0	*	*	*	*	*	*	*	*	0 0	(null)
1 0 1 1 1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	**	Count Register (high)
1 1 0 0 0	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	**	Count Register (mid)
1 1 0 0 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	**	Count Register (low)
1 1 0 1 0	*	*	*	*	*	*	*	ZDS	0 0	Timer Status Register
1 1 0 1 1	*	*	*	*	*	*	*	*	0 0	(null)
1 1 1 0 0	*	*	*	*	*	*	*	*	0 0	(null)
1 1 1 0 1	*	*	*	*	*	*	*	*	0 0	(null)
1 1 1 1 0	*	*	*	*	*	*	*	*	0 0	(null)
1 1 1 1 1	*	*	*	*	*	*	*	*	0 0	(null)

* Unused, read as zero.
** Value before RESET.

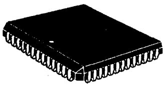
MK68564

SERIAL INPUT OUTPUT

- COMPATIBLE WITH MK68000 CPU
- COMPATIBLE WITH MK68000 SERIES DMA's
- TWO INDEPENDENT FULL-DUPLEX CHANNELS
- TWO INDEPENDENT BAUD-RATE GENERATORS
  - Crystal oscillator input
  - Single-phase TTL clock input
- DIRECTLY ADDRESSABLE REGISTERS (all control registers are read/write)
- DATA RATE IN SYNCHRONOUS OR ASYNCHRONOUS MODES
  - 0-1.25M bits/second with 5.0MHz system clock rate
- SELF-TEST CAPABILITY
- RECEIVE DATA REGISTERS ARE QUADRUPLY BUFFERED ; TRANSMIT REGISTERS ARE DOUBLY BUFFERED
- DAISY-CHAIN PRIORITY INTERRUPT LOGIC PROVIDES AUTOMATIC INTERRUPT VECTORING WITHOUT EXTERNAL LOGIC
- MODEM STATUS CAN BE MONITORED
  - Separate modem controls for each channel
- ASYNCHRONOUS FEATURES
  - 5, 6, 7, or 8 bits/character
  - 1, 1 1/2, or 2 stop bits
  - Even, odd, or no parity
  - x1, x16, x32, and x64 clock modes
  - Break generation and detection
  - Parity, overrun, and framing error detection
- BYTE SYNCHRONOUS FEATURES
  - Internal or external character synchronization
  - One or two sync characters in separate registers
  - Automatic sync character insertion
  - CDC-16 or CRC-CCITT block check generation and checking
- BIT SYNCHRONOUS FEATURES
  - Abort sequence generation and detection
  - Automatic zero insertion and deletion
  - Automatic flag insertion between messages
  - Address field recognition
  - I-field residue handling
  - Valid receive messages protected from overrun
  - CRC-16 or CRC-CCITT block check generation and checking



PDIP48
(Plastic Package)



PLCC52
(Chip Carrier)

DESCRIPTION

The MK68564 SIO (Serial Input Output) is a dual-channel, multi-function peripheral circuit, designed to satisfy a wide variety of serial data communications requirements in microcomputer systems. Its basic function is a serial-to-parallel, parallel-to-serial converter/controller ; however within that role, it is systems software configurable so that its "personality" may be optimized for any given serial data communications application.

The MK68564 is capable of handling asynchronous protocols, synchronous byte-oriented protocols (such as IBM Bisync), and synchronous bit-oriented protocols (such as HDLC and IBM SDLC). This versatile device can also be used to support virtually any serial protocol for applications other than data communications (cassette or floppy disk interface, for example).

The MK68564 can generate and check CRC codes in any synchronous mode and may be programmed to check data integrity in various modes. The device also has facilities for modem controls in each channel. In applications where these controls are not needed, the modem controls may be used for general-purpose I/O.

SIO PIN DESCRIPTION

GND :	Ground
V_{CC}:	+ 5 Volts (± 5%)
$\overline{\text{CS}}$:	Chip Select (input, active low). $\overline{\text{CS}}$ is used to select the MK68564 SIO for accesses to the internal registers. $\overline{\text{CS}}$ and $\overline{\text{IACK}}$ must not be asserted at the same time.
R/$\overline{\text{W}}$:	Read/write (input). R/$\overline{\text{W}}$ is the signal from the bus master, indicating wether the current bus cycle is a Read (high) or Write (low) cycle.
$\overline{\text{DTACK}}$:	Data Transfer Acknowledge (output, active low, three stateable). $\overline{\text{DTACK}}$ is used to signal the bus master that data is ready or that data has been accepted by the MK68564 SIO.
A1-A5 :	Address Bus (inputs). The address bus is used to select one of the internal registers during a read or write cycle.
D0-D7	Data Bus (bidirectional, threee-stateable). The data bus is used to transfer data to or from the internal registers during a read or write cycle. It is also used to pass a vector during an interrupt acknowledge cycle.
CLK :	Clock (input). This input is used to provide the internal timing for the MK68564 SIO.
$\overline{\text{RESET}}$:	Device Reset (input, active low). $\overline{\text{RESET}}$ disables both receivers and transmitters, forces TxDA and TxDB to a marking condition, forces the modem controls high and disables all interrupts. With the exception of the status registers, data registers, and the vector register, all internal registers are cleared. The vector register is reset to "0FH".
$\overline{\text{INTR}}$:	Interrupt Request (output, active low, open drain). $\overline{\text{INTR}}$ is asserted when the MK68564 SIO is requesting an interrupt. $\overline{\text{INTR}}$ is negated during an interrupt acknowledge cycle or by clearing the pending interrupt(s) through software.
$\overline{\text{IACK}}$:	Interrupt acknowledge (input, active low). $\overline{\text{IACK}}$ is used to signal the MK68564 SIO that the CPU is acknowledging an interrupt. $\overline{\text{CS}}$ and $\overline{\text{IACK}}$ must not be asserted at the same time.
$\overline{\text{IEI}}$:	Interrupt Enable In (input, active low). $\overline{\text{IEI}}$ is used to signal the MK68564 SIO that no higher priority device is requesting interrupt service.
$\overline{\text{IEO}}$:	Interrupt Enable Out (output, active low). $\overline{\text{IEO}}$ is used to signal lower priority peripherals that neither the MK68564 SIO nor another higher priority peripheral is requesting interrupt service.
XTAL1, XTAL2 :	Baud Rate Generator inputs. A crystal may be connected between XTAL1 and XTAL2, or XTAL1 may be driven with a TTL level clock. When using a crystal, external capacitors must be connectd. When driving XTAL1 with a TTL level clock, XTAL2 must be allowed to float.
$\overline{\text{RxRDYA}}$, $\overline{\text{RxRDYB}}$:	Receiver Ready (outputs, active low). Programmable DMA output for the receiver. The RxRDY pins pulse low when a character is available in the receive buffer.
$\overline{\text{TxRDYA}}$, $\overline{\text{TxRDYB}}$:	Transmitter Ready (outputs, active low). Programmable DMA output for the transmitter. The $\overline{\text{TxRDY}}$ pins pulse low when the transmit buffer is empty.
$\overline{\text{CTSA}}$, $\overline{\text{CTSB}}$:	Clear to Send (inputs, active low). If Tx Auto Enables is selected, these inputs enable the transmitter of their respective channels. If Tx Auto Enables is not selected, these inputs may be used as general purpose input pins. The inputs are Scmit-trigger buffered to allow slow rise-time input signals.
$\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$:	Data Carrier Detect (inputs, active low). If Rx Auto Enables is selected, these inputs enable the receiver of their respective channels. If Rx Auto Enables is not selected, these inputs may be used as general purpose input pins. The inputs are Schmit-trigger buffered to allow slow rise-time input signals.
RxDA, RxDB :	Receive Data (inputs, active high). Serial data input to the receiver.
TxDA, TxDB :	Transmit Data (outputs, active high). Serial data output of the transmitter.

SIO PIN DESCRIPTION (continued)

RxCA, RxCB :	Receiver Clocks (input/output). Programmable pin, receive clock input, or baud rate generator output. The inputs are Schmit-trigger buffered to allow slow rise-time input signals.
TxCA, TxCB :	Transmitter Clocks (input/output). Programmable pin, transmit clock input, or baud rate generator output. The inputs are Schmit-trigger buffered to allow slow rise-time input signals.
RTSA, RTSB :	Request to Send (outputs, active low). These outputs follow the inverted state programmed into the RTS bit. When the RTS bit is reset in the asynchronous mode, the output will not change until the character in the transmitter is completely shifted out. These pins may be used as general purpose outputs.
$\overline{DTRA}$, $\overline{DTRB}$:	Data Terminal Ready (outputs, Active low). These outputs follow the inverted state programmed into the DTR bit. These pins may also be used as general purpose outputs.
$\overline{SYNCA}$, $\overline{SYNCB}$:	Synchronization (input/output, active low). The $\overline{SYNC}$ pin is an output when Monosync, Bisync, or SDLC mode is programmed. It is asserted when a sync/flag character is detected by the receiver. The SYNC pin is a general purpose input in the Asynchronous mode and an input to the receiver in the External Sync Mode.

Figure 1a : Dual In Line Pin Configuration.

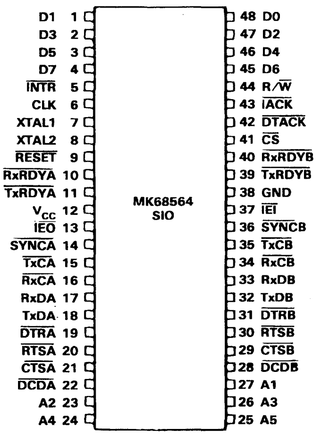
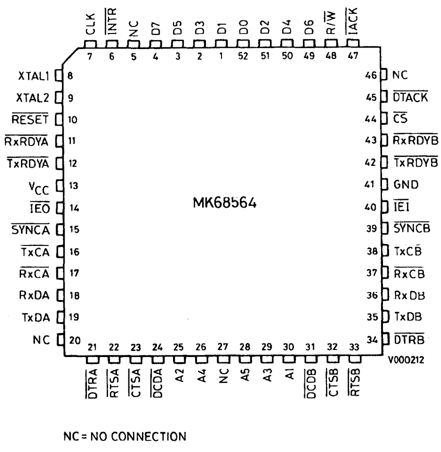


Figure 1b : Chip Carrier Pin Configuration.



SIO SYSTEM INTERFACE

INTRODUCTION

The MK68564 SIO is designed for simple and efficient interface to a MK68000 CPU system. All data transfers between the SIO and the CPU are asynchronous to the system clock. The SIO system timing is derived from the chip select input (CS) during normal read and write sequences, and from the interrupt acknowledge input (IACK) during an exception processing sequence. CS is a function of address decode and (normally) lower data strobe (LDS). IACK is a function of the interrupt level on address lines A1, A2, and A3, an interrupt acknowledge function code (FC0-FC2), and LDS.

Note : CS and IACK can never be asserted at the same time.

Note : Unused inputs should be pulled up or down, but never left floating.

READ SEQUENCE

The SIO will begin a read cycle if, on the falling edge of CS, the read-write (R/W) pin is high. The SIO will respond by decoding the address bus (A1-A5) for the register selected, by placing the contents of that register on the data bus pins (D0-D7), and by driving the data transfer acknowledge (DTACK) pin low. If the register selected is not implemented on the SIO, the data bus pins will be driven high, and then DTACK will be asserted. When the CPU has acquired the data, the CS signal is driven high, at which time the SIO will drive DTACK high and then three-state DTACK and D0-D7.

WRITE SEQUENCE

The SIO will begin a write cycle if, on the falling edge of CS, the R/W pin is low. The SIO will respond by latching the data bus, by decoding the address bus for the register selected, by loading the register with the contents of the data bus, and by driving DTACK low. When the CPU has finished the cycle, the CS input is driven high. At this time, the SIO will drive DTACK high and will then three-state DTACK. If the register selected is not implemented on the SIO, the normal write sequence will proceed, but the data bus contents will not be stored.

INTERRUPT SEQUENCE

The SIO is designed to operate as an independent, interrupting peripheral, or, when interconnected with other components, an interrupt priority daisy chain can be formed.

Independent Operation. Independent operation requires that the interrupt enable in pin (IEI) be connected to ground. The SIO starts the interrupt sequence by driving the interrupt request pin (INTR) low. The CPU responds to the interrupt by starting an interrupt acknowledge cycle, in which the SIO IACK pin is driven low. The highest priority interrupt request in the SIO, at the time IACK goes low, places its vector on the data bus pins. The SIO releases the INTR pin and drives DTACK low. When the CPU has acquired the vector, the IACK signal is driven high. The SIO responds by driving DTACK to a high level and then three-stating DTACK and D0-D7. If more than one interrupt request is pending at the start of an interrupt acknowledge sequence, the SIO will drive the INTR pin low following the completion of the interrupt acknowledge cycle. This sequence will continue until all pending interrupts are cleared. If the SIO is not requesting an interrupt when IACK goes low, the SIO will not respond to the IACK signal ; DTACK and the data bus will remain three-stated.

Daisy Chain Operation. The interrupt priority chain is formed by connecting the interrupt enable out pin (IEO) of a higher priority part to IEI of the next lower priority part. The highest priority part in the chain should have IEI tied to ground. The Daisy Chaining capability (figures 2 and 3) requires that all parts in a chain have a common IACK signal. When the common IACK goes low, all parts freeze and prioritize interrupts in parallel. Then priority is passed down the chain, via IEI and IEO, until a part which has a pending interrupt, once IEI goes low, passes a vector, does not propagate IEO, and generates DTACK.

The state of the IEI pin does not affect the SIO interrupt control logic. The SIO can generate an interrupt request any time its interrupts are enabled. The IEO pin is normally high ; it will only go low during an IACK cycle if IEI is low and no interrupt is pending in the SIO. The IEO pin will be forced high whenever IACK or IEI goes high.

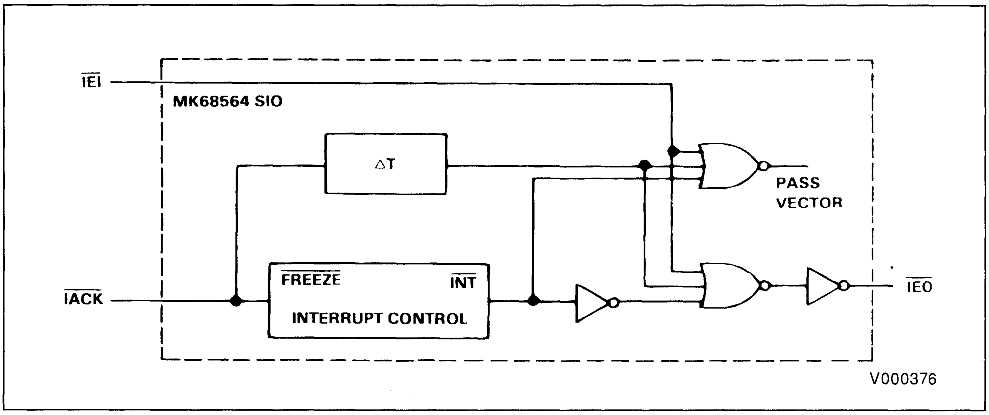


Figure 2 : Conceptual Circuit of the MK68564 SIO Daisy Chaining Logic.

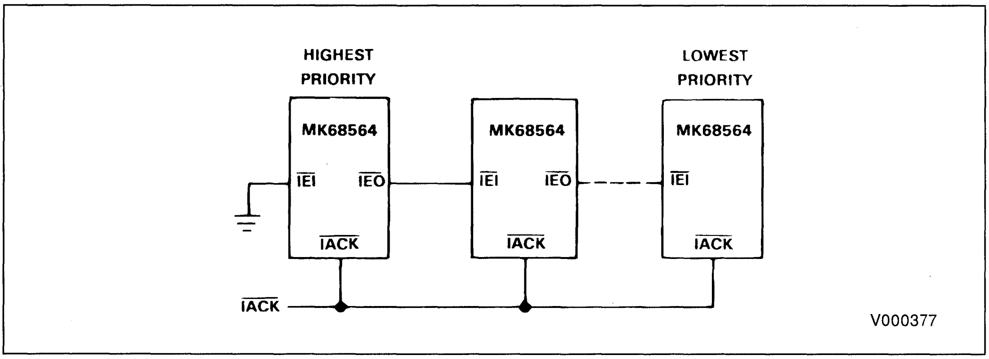


Figure 3 : Daisy Chaining.

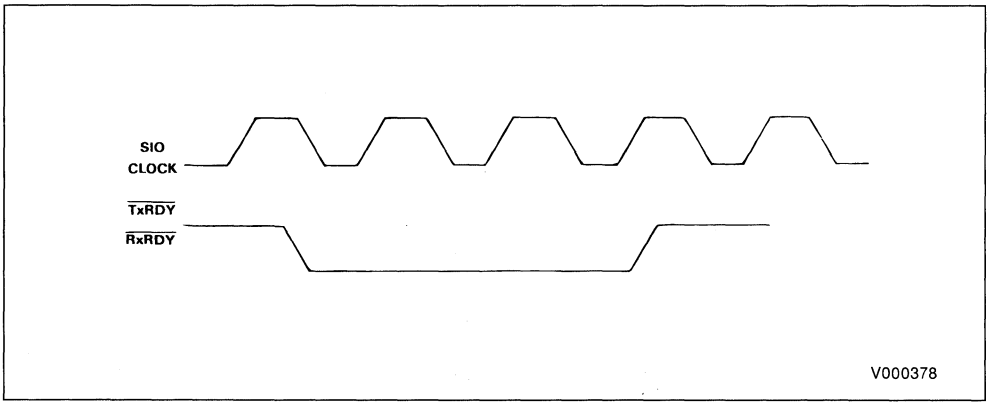


Figure 4 : DMA Interface Timing.

DMA INTERFACE

The SIO is designed to interface to the 68000 family DMA's as a 68000 compatible device, using the cycle steal mode. The SIO provides four outputs (TxRDYA, RxRDYA, TxRDYB, RxRDYB) for requesting service from the DMA. The SIO issues a request for service by pulsing the RDY pin low for three clock (CLK) cycles (see figure 4). TxRDY (when enabled) will be active when the transmit buffer becomes empty. RxRDY (when enabled) will be active when a character is available in the receive buffer. If Receive Interrupt On First Character Only is enabled during a DMA operation and a special receive condition is detected, the RxRDY pin will not become active. Instead, a special receive condition interrupt will be generated by the channel.

RESET

There are two ways of resetting the SIO : an individual, programmable channel reset and an external hardware reset.

The individual channel reset is generated by writing "18H" to the Command Register for the channel selected. All outputs associated with the channel are reset high, TxC and RxC are inputs, SYNC is an output, and TxD is forced marking. All R/W registers for the channel are reset to "00H", except the vector register and the data register, which are not affected.

Read only status register 1 is reset to "01H" (All Sent set). Break/Abort, Interrupt Pending, and Rx Character Available bits in read only status register 0 are reset ; Underrun/EOM, Hunt/Sync, and Tx Buffer Empty are set ; CTS and DCD bits are set to the inverted state of their respective input pins. Any interrupts pending for the channel are reset (any pending interrupts in the other channel will not be affected).

An external hardware reset occurs when the RESET pin is driven low for at least one clock (CLK) cycle. Both channels are reset as listed above, and the vector register is reset to "0FH".

ARCHITECTURE

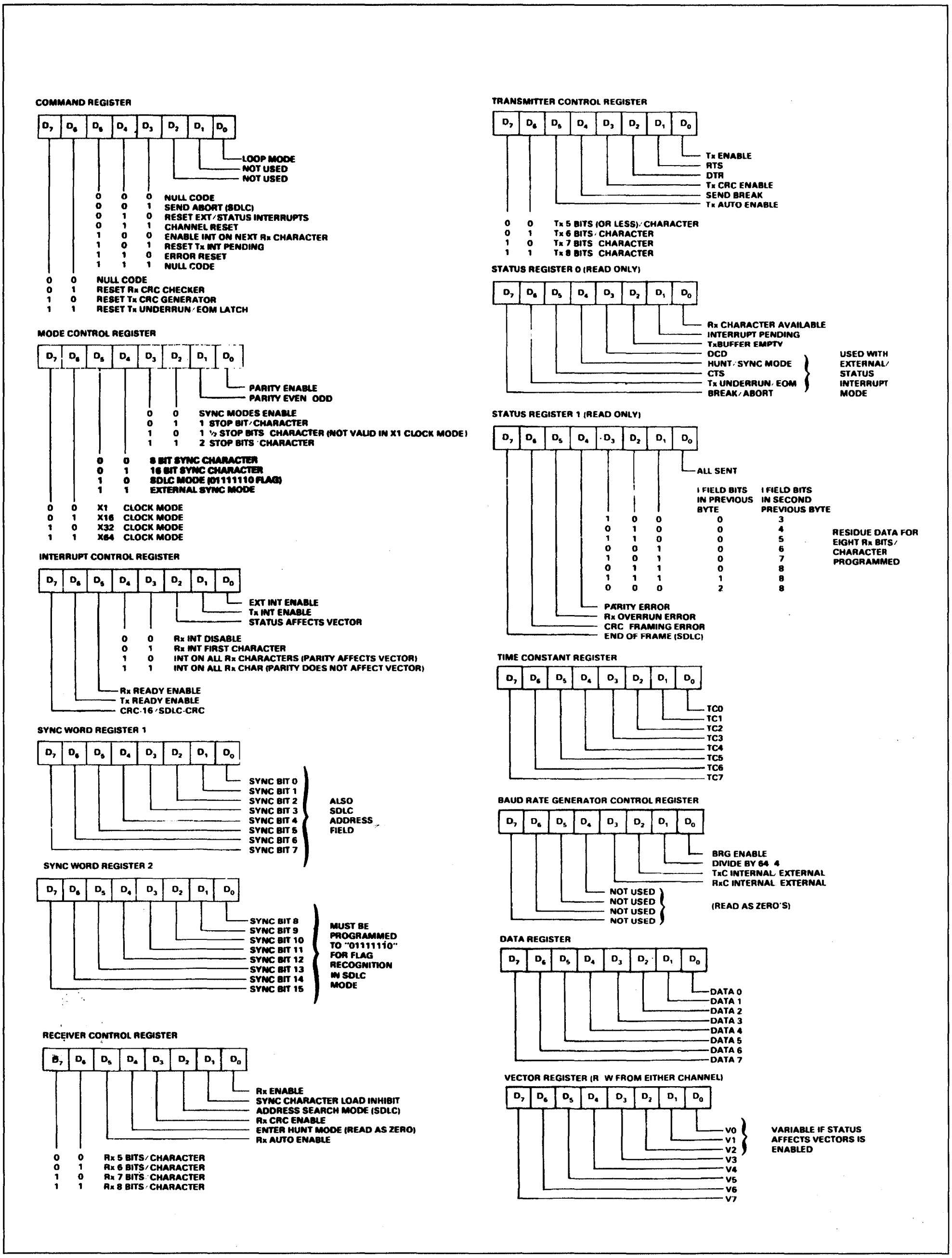
The MK68564 SIO contains two independent, full-duplex channels. Each channel contains a transmitter, receiver, modem control logic, interrupt control logic, a baud rate generator, ten Read/Write registers, and two read only status registers. Each channel can communicate with the bus master using polling, interrupts, DMA, or any combination of these three techniques. Each channel also has the ability to connect the transmitter output into the receiver without disturbing any external hardware.

Register Set. The register set is the heart of each channel. A channel is configured for different communication protocols and interface options by programming the registers. Table 1 lists all the registers available in the SIO and their addresses.

Data Register. The Data Register is composed of two separate registers : a write only register, which is the Transmit Buffer, and a read only register, which is the Receive Buffer. The Receive Buffer is also the top register of a three register stack called the receive data FIFO.

Vector Register. The Vector Register is different from the other 24 registers, because it may be accessed through either Channel A or Channel B during a R/W cycle. During an Interrupt Acknowledge cycle, the contents of the Vector Register are passed to the CPU to be used as a pointer to an interrupt service routine. If the Status Affects Vector bit is Low in the Interrupt Control Register, any data written to the Vector Register will be returned unmodified during a Read Cycle or an IACK cycle. If the Status Affects Vector bit is High, the lower three bits of the vector returned during a Read or IACK cycle are modified to reflect the highest priority interrupt pending in the SIO at that time. The upper five bits written to the Vector Register are unaffected. After a hardware reset only, this register contains a "0FH" value, which is the MK68000's uninitialized interrupt vector assignment.

Figure 5 : Register Bit Functions.



SIO INTERNAL REGISTERS

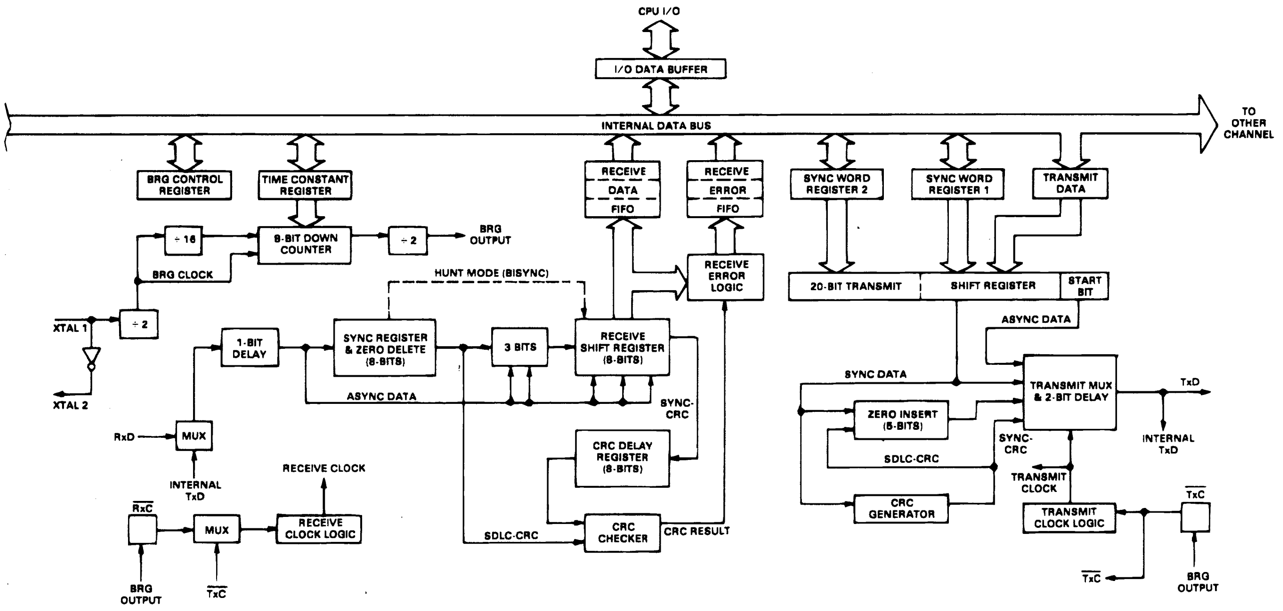
The MK68564 SIO has 25 internal registers. Each channel has ten R/W registers and two read only registers associated with it. The vector register may be accessed through either channel.

Table 1 : Register Map.

Address					Abbreviation	Channel	Register Name	Access	
5	4	3	2	1				Read/ write	Read Only
0	0	0	0	0	CMDREG	A	Command Register	X	
0	0	0	0	1	MODECTL	A	Mode Control Register	X	
0	0	0	1	0	INTCTL	A	Interrupt Control Register	X	
0	0	0	1	1	SYNC 1	A	Sync Word Register 1	X	
0	0	1	0	0	SYNC 2	A	Sync Word Register 2	X	
0	0	1	0	1	RCVCTL	A	Receiver Control Register	X	
0	0	1	1	0	XMTCTL	A	Transmitter Control Register	X	
0	0	1	1	1	STAT 0	A	Status Register 0		X
0	1	0	0	0	STAT 1	A	Status Register 1		X
0	1	0	0	1	DATARG	A	Data Register	X	
0	1	0	1	0	TCREG	A	Time Constant Register	X	
0	1	0	1	1	BRGCTL	A	Baud Rate Generator Control Reg	X	
0	1	1	0	0	VECTRG	A/B	Interrupt Vector Register (note 2)	X	
0	1	1	0	1		A	(note 1)	X	
0	1	1	1	0		A	(note 1)	X	
0	1	1	1	1		A	(note 1)	X	
1	0	0	0	0	CMDREG	B	Command Register	X	
1	0	0	0	1	MODECTL	B	Mode Control Register	X	
1	0	0	1	0	INTCTL	B	Interrupt Control Register	X	
1	0	0	1	1	SYNC 1	B	Sync Word Register 1	X	
1	0	1	0	0	SYNC 2	B	Sync Word Register 2	X	
1	0	1	0	1	RCVCTL	B	Receiver Control Register	X	
1	0	1	1	0	XMTCTL	B	Transmitter Control Register	X	
1	0	1	1	1	STAT 0	B	Status Register 0		X
1	1	0	0	0	STAT 1	B	Status Register 1		X
1	1	0	0	1	DATARG	B	Data Register	X	
1	1	0	1	0	TCREG	B	Time Constant Register	X	
1	1	0	1	1	BRGCTL	B	Baud Rate Generator Control Reg	X	
1	1	1	0	0	VECTRG	A/B	Interrupt Vector Register (note 2)	X	
1	1	1	0	1		B	(note 1)	X	
1	1	1	1	0		B	(note 1)	X	
1	1	1	1	1		B	(note 1)	X	

Notes : 1. Not Used, Read as "FFH".
2. Only One Vector Register, Accessible through Either Channel.

Figure 6 : Transmit and Receive Data Paths.



V000379

DATA PATH

The transmit and receive data paths for each channel are shown in figure 6. The receiver has three 8-bit buffer registers in a FIFO arrangement (to provide a 3-byte delay) in addition to the 8-bit receive shift register. This arrangement creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. The receiver error FIFO stores parity and framing errors and other types of status information for each of the three bytes in the receive data FIFO. The receive error FIFO is loaded at the same time as the receive data FIFO. The contents of the receive error are read through the upper four bits in Status Register 1.

Incoming data is routed through one of several paths, depending on the mode and character length. In the Asynchronous modes, serial data is entered into the 3-bit buffer, if it has a character length of seven or eight bits, or the data is entered into the 8-bit receive shift register, if it has a length of five or six bits.

In the Synchronous mode, the data path is determined by the phase of the receive process currently in operation. A Synchronous Receive operation begins with the receiver in the Hunt phase, during which time the receiver searches the incoming data stream for a bit pattern that matches the preprogrammed sync characters (or flags in the SDLC mode). If the device is programmed for Monosync Hunt, a match is made with a single sync character stored in Sync Word Register 2. In Bisync Hunt, a match is made with the dual sync characters stored in Sync Word Registers 1 and 2. In either case, the incoming data passes through the receive sync register and is compared against the programmed sync characters in Sync Word Registers 1 and 2.

In the Monosync mode, a match between the sync character programmed into Sync Word Register 2 and the character assembled in the receive sync register establishes synchronization.

In the Bysync mode, incoming data is shifted to the receive shift register, while the next eight bits of the message are assembled in the receive sync register. The match between the assembled character in the sync register and the programmed character in Sync Word Register 2, and between the character in the shift register and the programmed character in Sync Word Register 1 establishes synchronization. Once synchronization is established, incoming data bypasses the receive sync register and directly enters the 3-bit buffer.

In the SDLC mode, all incoming data passes through the receive sync register, which continuously monitors the receive data stream and performs zero deletion when indicated. Upon receiving five contiguous ones, the sixth bit is inspected. If the sixth bit is a 0, it is deleted from the data stream. If the sixth bit is a 1, the seventh bit is inspected. If the seventh bit is a 0, a Flag sequence has been received ; if the seventh bit is a 1, an Abort sequence has been received.

The reformatted data from the receive sync register enters the 3-bit buffer and is transferred to the receive shift register. Note that the SDLC receive operation also begins in the Hunt Phase, during which time the SIO tries to match the assembled character in the receive sync register with the flag pattern in Sync Word Register 2. Once the first flag character is recognized, all subsequent data is routed through the path described above, regardless of character length.

Although the same CRC checker is used for both SDLC and synchronous data, the path taken for each mode is different. In Bisync protocol, the byte-oriented operation requires that the CPU decide whether or not to include the data character in the CRC calculation. To allow the CPU ample time to make this decision, the SIO provides an 8-bit delay before the data enters the CRC checker. In the SDLC mode, no delay is provided, since CRC is calculated on all data between the opening and closing flags.

The transmitter has an 8-bit transmit data register, which is loaded from the internal bus, and a 20-bit transmit shift register, which can be loaded from Sync Word Register 1, Sync Word Register 2, and the transmit data register. Sync Word Registers 1 and 2 contain sync characters in the Monosync, Bisync, or External Sync modes, or address field (one character long) and flag, respectively, in the SDLC mode. During Synchronous modes, information contained in Sync Word Registers 1 and 2 is loaded into the transmit shift register at the beginning of the message and, as a time filler, in the middle of the message if a Transmit Underrun condition occurs. In SDLC mode, the flags are loaded into the transmit shift register at the beginning and end of the message.

Asynchronous data in the transmit shift register is formatted with start and stop bits, and it is shifted out to the transmit multiplexer at the selected clock rate.

Synchronous (Monosync, Bisync, or External Sync) data is shifted out to the transmit multiplexer and also the CRC generator at the x1 clock rate.

SDLC/HDLC data is shifted out through the zero insertion logic, which is disabled while flags are being sent. For all other fields (address, control, and frame check), a 0 is inserted following five contiguous ones in the data stream. Note that the CRC generator result (frame check) for SDLC data is also routed through the zero insertion logic.

I/O CAPABILITIES

The SIO offers the choice of Polling, Interrupt (vectored or non-vectored), and DMA Transfer modes to transfer data, status, and control information to and from the CPU or other bus master.

Polling. The Polled mode avoids interrupts. Status Registers 0 and 1 are updated at appropriate times for each function being performed (for example, CRC Error status valid at the end of the message). All the interrupt modes of the SIO must be disabled to operate the device in a polled environment.

While in its Polling sequence, the CPU examines the status contained in Status Register 0 for each channel. The state of the status bits in Status Register 0 serves as an acknowledge to the Poll inquiry. Status bits D0 and D2 indicate that a receive or transmit data transfer is needed. The rest of the status bits in Status Register 0 indicate special status conditions. The receiver error condition bits in Status Register 1 do not have to be read until the Rx Character Available status bit in Status Register 0 is set to a one.

Interrupts. The SIO offers an elaborate interrupt scheme to provide fast interrupt response in real-time applications. The interrupt vector points to an interrupt service routine in the memory. To service operations in both channels and to eliminate the necessity of writing a status analysis routine (as required for a polling scheme), the SIO can modify the interrupt vector so it points to one of eight interrupt service routines. This is done under program control by setting the Status Affects Vector bit in the Interrupt Control Register of channel A or channel B, to a one. When this bit is set, the interrupt vector is modified according to the assigned priority of the various interrupting conditions.

Note : If the Status Affects Vector bit is set in either channel, the vector is modified for both channels. This is the only control bit that operates in this manner in the SIO.

Transmit interrupts, Receive interrupts, and External/Status interrupts are the sources of interrupts. Each interrupt source is enabled under program control with Channel A having a higher priority than Channel B, and with Receiver, Transmitter, and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. This implies that the transmitter must have had a data character written into it so t can become empty. When enabled, the receiver can interrupt the CPU in one of three ways :

Interrupt On First Character Only

Interrupt On All Receive Characters

Interrupt On A Special Receive Condition.

Interrupt On First Character Only.This mode is normally used to start a software Polling loop or a DMA transfer routine using the RxRDY pin. In this mode, the SIO generates an interrupt on the first character received after this mode is selected and, thereafter, only generates an interrupt if a Special Receive Condition occurs. The Special Receive Conditions that can cause an interrupt in this mode are : Rx Overrun Error, Framing Error (in Asynchronous modes), and End Of Frame (in SDLC mode). This mode is reinitialized by the Enable Interrupt On Next Rx Character command. If a Special Receive Condition interrupt occurs in this interrupt mode, the data with the special condition is held in the receive data FIFO until an Error Reset Command is issued.

Interrupt On All Receive Characters. In this mode, an interrupt is generated whenever the receive data FIFO contains a character or a Special Receive Condition occurs. The Special Receive Conditions that can cause an interrupt in this mode are : Rx Overrun Error, Framing Error (in Asynchronous modes), End of Frame (in SDLC mode), and Parity Error (if selected).

Interrupt On A Special Receive Condition. The Special Receive Condition interrupt is not, as such, a separate interrupt mode. Before a Special Receive Condition can cause an interrupt, either the Interrupt On First Character Only or Interrupt On All Receive Characters mode must be selected. The Special Receive Condition interrupt will modify the receive interrupt vector if Status Affects Vector is enabled. The Special Receive Condition status is displayed in the upper four bits of Status Register 1. Two of the conditions causing a special receive interrupt are latched when they occur ; they are : Parity Error and Rx Overrun Error. These status bits may only be reset by an Error Reset command. When either of these conditions occur, a read of Status Register 1 will reflect any errors in the current word in the receive buffer plus any parity or overrun errors since the last Error Reset command was issued.

External/Status Interrupts. The main function of the External/Status interrupt is to monitor the signal transitions of the CTS, DCD, and SYNC pins ; however, an External/Status interrupt is also caused by a Transmit Underrun condition or by the detection of a Break (Asynchronous mode) or Abort

(SDLC mode) sequence in the received data stream. When any one of the above conditions occur, the external/status logic latches the current state of all five input conditions, and generates an interrupt. To reinitialize the external/status logic to detect another transition, a Reset External/Status Interrupts command must be issued. The Break/Abort condition allows the SIO to generate an interrupt when the Break/Abort sequence is detected and terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Break/Abort condition in external logic.

DMA Transfer

The SIO provides two output signals per channel for connection to a DMA controller ; they are TxRDY and RxRDY. The outputs are enabled under software control by writing to the Interrupt Control Register. Both outputs will pulse Low for three system clock cycles when their input conditions are active. TxRDY will be active when the Transmit Buffer becomes empty. RxRDY will be active when a character is available in the Receive Buffer. If a Special Receive Condition occurs when Interrupt On First Character Only mode is selected, a receiver interrupt will be generated and RxRDY will not become active. This will automatically inform the CPU of a discrepancy in the data transfer.

SELF TEST

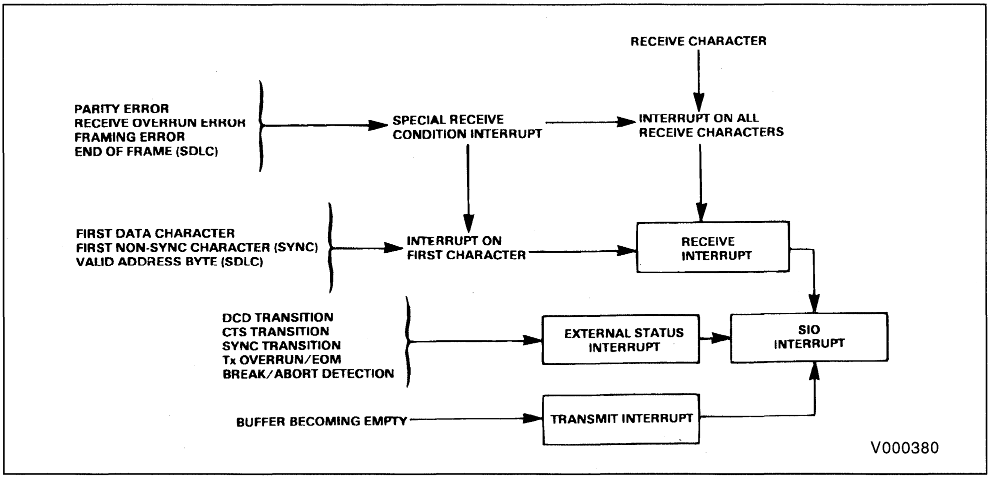
When the Loop Mode bit is set in the Command Register, the receiver shift clock input pin (RxC) and the receiver data input pin (RxD) are electrically disconnected from the internal logic. The transmit data output pin (TxD) is connected to the internal receiver data logic, and the transmit shift clock pin (TxC) is connected to the internal receiver shift clock logic. All other features of the SIO are unaffected.

BAUD RATE GENERATORS

Each channel in the SIO contains a programmable baud rate generator (BRG). Each BRG consists of an 8-bit time constant register, an 8-bit down counter, a control register, and a flip-flop on the output to provide a square wave signal out. In addition to the flip-flop on the output, there is also a flip-flop on the input clock ; therefore, the maximum output frequency of the BRG is one-forth of the input clock frequency. This maximum output frequency occurs when divide by four mode is selected, and the time constant register is loaded with the minimum count of "01H". The equation to determine the output frequency is :

$$\text{Output Frequency} = \frac{\text{Input Frequency}}{\text{(divide by selected) X (time constant value in decimal)}}$$

Figure 7 : Interrupt Structure.



For example, when the time constant register is loaded with "01H" and divide by four is selected, one output clock will occur for every four input clocks. If the time constant value loaded is "00H" (256 decimal) instead of "01H" and divide by 64 is selected, one output clock will occur for every 16384 input clocks. Note that the minimum count value is "01H" (1 decimal), and the maximum count value is "00H" (256 decimal).

The output of the baud rate generator may be programmed to drive the transmitter (BRG output on TxC), the receiver (BRG output on RxC), both (BRG output on TxC and RxC), or neither (TxC and RxC are inputs). After a reset, the baud rate generator is disabled, divide by four is selected, and TxC and RxC are inputs.

The baud rate generator should be disabled before the CPU writes to the time constant register. This is necessary because no attempt was made to synchronize the loading of a new time constant with the clock used to drive the BRG.

Figure 8 indicates the external components needed to connect a crystal oscillator to the SIO XTAL inputs. The allowed crystal parameters are also listed.

For a 3.6864MHz input signal to the baud rate generator, the time constants, listed in table 2, are loaded to obtain the desired baud rates (in x1 clock mode).

ASYNCHRONOUS OPERATION

INTRODUCTION

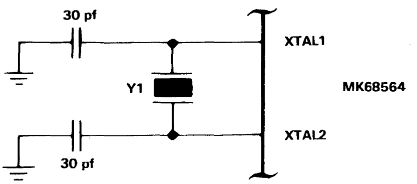
Many types of Asynchronous operations are performed by the MK68564 SIO. Figure 9 represents a typical Asynchronous message format and some of the options available on the SIO. The transmit process inserts start, stop, and parity bits to a variable data format and supplies a serial data stream to the Transmit Data output (TxD). The receiver takes the data from the Receive Data input (RxD) and strips away expected start and stop bits at a programmed clock rate. It provides error checking for overrun, parity, and carrier-loss errors, and, if desired, provides interrupts for these conditions.

To set up the SIO for Asynchronous operation, the following registers need to be initialized : Mode Control Register, Interrupt Control Register, Receiver Control Register, and Transmitter Control Register. The Mode Control Register must be programmed before the other registers to assure proper operation of the SIO. The following registers are used to transfer data or to communicate status between the SIO and the CPU or other bus master when operating in Asynchronous modes : Command Register, Status Register 0, Status Register 1, Data Register, and the Vector Register.

Table 2 : Time-Constant Values.

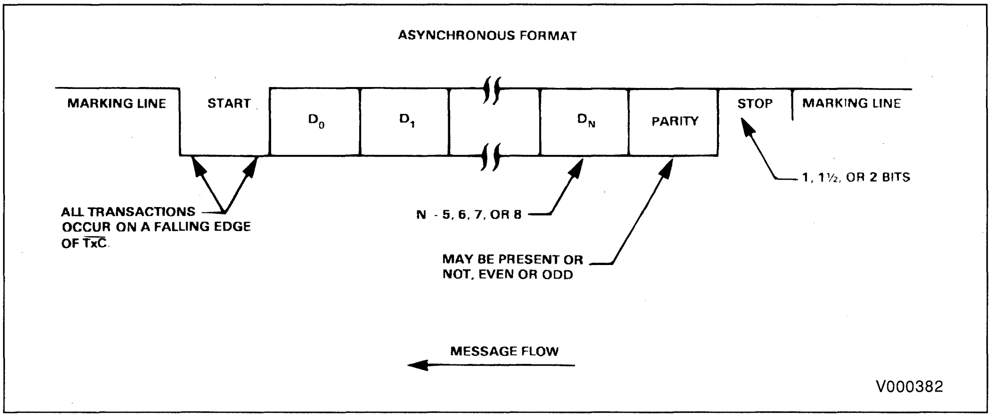
Rate	Time Constant	Divide By	Error
19200	48	4	
9600	96	4	
7200	128	4	
4800	192	4	
3600	256	4	
2400	24	64	
2000	29	64	69 %
1800	32	64	
1200	48	64	
600	96	64	
300	192	64	

Figure 8 : SIO External Oscillator Components.



CRYSTAL PARAMETERS :
Parallel Resonance, Fundamental Mode AT Cut
Rs â 150Ω (Fr = 2.8 - 5.0MHz)
Rs â 300Ω (Fr = 2.0 - 2.7MHz)
Cl = 18pf ; Cm = 0.02pF ; Ch = 5pF ; Lm = 96MHz
Fr (typ) = 2.457MHz

Figure 9 : Asynchronous Message Format.



The SIO provides five I/O lines that may be used for modem control, for external interrupts, or as general purpose I/O. The Request To Send (RTS) and Data Terminal Ready (DTR) pins are outputs that follow the inverted state of their respective bits in the Transmitter Control Register. The RTS pin can also be used to signal the end of a message in Asynchronous modes, as explained below in the transmitter section. The Data Carrier Detect (DCD), Clear To Send (CTS), and SYNC pins are inputs to the SIO in Asynchronous modes. DCD and CTS can be used as auto enables to the receiver and transmitter, respectively, or if External/Status Interrupts are enabled all three input pins will be monitored for a change of status. If these inputs change for a period of time greater than the minimum specified pulse width, an interrupt will be generated.

In the following discussion, all interrupt modes are assumed enabled.

ASYNCHRONOUS TRANSMIT

Start of Transmission. The SIO will start transmitting data when the Transmit Enable bit is set to a one, and a character has been loaded into the transmit buffer. If the TxAuto Enables bit is set, the SIO will wait for a Low on the Clear To Send input (CTS) before starting data transmission. The Tx Auto Enables feature allows the programmer to send the first data character of the message to the SIO without waiting for CTS to go Low. In all cases, the Transmit Enable bit must be set before transmission can begin. The transitions on the CTS pin will generate External/Status interrupt requests and also latch up the external/status logic. The external/status logic should be rearmed by issuing a Reset External/Status Interrupts command.

Transmit Characteristics. The SIO automatically inserts a start bit, the programmed parity bit (odd, even, or no parity), and the programmed number of stop bits to the data character to be transmitted. The transmitter can transmit from one to eight data bits per character. All characters are transmitted least-significant bit first. When the character length programmed is six or seven bits, the unused bits of the transmit buffer are automatically ignored. When a character length of five bits or less is programmed, the data loaded into the transmit buffer must be formatted as described in the Transmitter Control Register part of the Register Description section. Serial data is shifted out of the TxD pin on the falling edge of the Transmit Clock (TxC) at a rate equal to 1, 1/16th, 1/32nd, or 1/64th of TxC.

Data Transfer. The SIO will signal the CPU or other bus master with a transmit interrupt request and set the Tx Buffer Empty bit in Status Register 0, every time the contents of the transmit buffer are loaded into the transmit shift register. The interrupt request will be cleared when a new character is loaded into the transmit buffer, or a Reset Tx Interrupt Pending command (Command 5) is issued. If Command 5 is issued, the transmit buffer will have to be loaded before any additional transmit interrupt requests are generated. The Tx Buffer Empty bit is reset when a new character is loaded into the transmit buffer.

The All Sent bit in Status Register 1 is used to indicate when all data in the shift register has been transmitted, and the transmit buffer is empty. This bit is Low, while the transmitter is sending characters, and it will go High one bit time after the transmit clock that clocks out the last stop bit of the character on the TxD pin. No interrupts are generated by the All Sent bit transitions. The Request To Send

(RTS) bit in the Transmitter Control Register may also be used to signal the end of transmission. If this bit is set to a one, its associated output pin (RTS) will go Low. When this bit is reset to a zero, the RTS pin will go High one bit time after the transmit clock that clocks out the last stop bit, only if the transmit buffer is empty.

The Transmit Data output (TxD) is held marking (High) after a reset or when the transmitter has no data to send. Under program control, the Send Break command can be issued to hold TxD spacing (Low) until the command is cleared, even if the transmitter is not enabled.

ASYNCHRONOUS RECEIVE

Asynchronous operation begins when the Receiver Enable bit in the Receiver Control Register is set to a one. If the Rx Auto Enables bit is also set, the Data Carrier Detect (DCD) input pin must be Low as well. The receiver will start assembling a character as soon as a valid start bit is detected, if a clock mode other than x1 is selected. A valid start bit is a High-to-Low transition on the Receive Data input (RxD) with the Low time lasting at least one-half bit time. The High-to-Low transition starts an internal counter and, at mid-bit time, the counter output is used to sample the input signal to detect if it is still Low. When this condition is satisfied, the following data bits are sampled at mid-bit time until the entire character is assembled. The start bit detection logic is then rearmed to detect the next High-to-Low transition. If the x1 clock mode is selected, the start bit detection logic is disabled, and bit synchronization must be accomplished externally. Receive data is sampled on the rising edge of the Receiver Clock (RxC).

The receiver may be programmed to assemble five to eight data bits, plus a parity bit, into a character. The character is right-justified in the shift register and then transferred to the receive data FIFO. All data transfers to the FIFO are in eight-bit groups. If the character length assembled is less than eight bits, the receiver inserts ones in the unused bits. If parity is enabled, the parity bit is transferred with the character, unless eight bits per character is programmed, in which case, the parity bit is stripped from the character before transfer.

A Receiver Interrupt request is generated every time a character is shifted to the top of the receive data FIFO, if Interrupt On All Receive Characters mode is selected. The Rx Character Available bit in Status Register 0 is also set to a one every time a character is shifted to the top of the receive data FIFO. The Rx Character Available bit is reset to a zero when the receive buffer is read.

After a character is received, it is checked for the following error conditions :

Parity Error. If parity is enabled, the Parity Error bit in Status Register 1 is set to a one whenever the parity bit of the received character does not match the programmed parity. Once this bit is set, it remains set (latched), until an Error Reset command (Command 6) is issued. A Special Receive Condition interrupt is generated when this bit is set, if parity is programmed as a Special Receive Condition.

Framing Error. The CRC/Framing Error bit in Status Register 1 is set to a one, if the character is assembled without a stop bit (a Low level detected instead of a stop bit). This bit is set only for the character on which the framing error occurred ; it is updated at every character time. Detection of a framing error adds an additional one-half of a bit time to the character time, so the framing error is not interpreted as a new start bit. A Special Receive Condition interrupt is generated when this bit is set..

Overrun Error. If four or more characters are received before the CPU (or other bus master) reads the receive buffer, the fourth character assembled will replace the third character in the receive data FIFO. If more than four characters have been received, the last character assembled will replace the third character in the data FIFO. The character that has been written over is flagged with an overrun error in the error FIFO.

When this character is shifted to the top of the receive data FIFO, the Receive Overrun Error bit in Status Register 1 is set to a one ; the error bit is latched in the status register, and a Special Receive Condition interrupt is generated. Like Parity Error, this bit can only be reset by an Error Reset Command.

Break Condition. A break character is defined as a start bit, an all zero data word, and a zero in place of the stop bit. When a break character is detected in the receive data stream, the Break/Abort bit in Status Register 0 is set to a one, and an External/Status interrupt is requested. This interrupt is then followed by a Framing Error interrupt request when the CRC/Framing Error bit in Status Register 1 is set. A Reset External/Status Interrupts command (Command 2) should be issued to reinitialize the break detection interrupt logic. The receiver will monitor the data stream input for the termination of the break sequence. When this condition is detected, the Break/Abort bit will be reset, if Command 2 has been issued, and another External/Status interrupt request will be generated. This interrupt should also be handled by issuing Command 2 to reinitialize the external/status logic. At the end of the break sequence, a single null character will be left in the receive data FIFO. This character should be read and discarded.

Because Parity Error and Receive Overrun Error flags are latched, the error status that is read from Status Register 1 reflects an error in the current word in the receive data FIFO, plus any parity or overrun errors received since the last Error Reset command. To keep correspondence between the state of the error FIFO and the contents of the receive data FIFO, Status Register 1 should be read before the receive buffer. If the status is read after the data and more than one character is stacked in the data FIFO during the read of the receive buffer, the status flags read will be for the next word. Keep in mind that when a character is shifted up to the top of the data FIFO (the receive buffer), its error flags are shifted into Status Register 1

.An exception to the normal flow of data through the receive data FIFO occurs when the Receive Interrupt On First Character Only mode is selected. A Special Receive Condition interrupt in this mode holds the error data, and the character itself (even if read from the data FIFO) until the Error Reset command (command 6) is issued. This prevents further data from becoming available in the receiver, until Command 6 is issued, and allows CPU intervention on the character with the error even if DMA or block transfer techniques are being used.

SYNCHRONOUS OPERATION

INTRODUCTION

Before describing byte-oriented, synchronous transmission and reception, the three types of character synchronization - Monosync, Bysync, and External Sync - require some explanation. These modes use the x1 clock for both Transmit and Receive operations. Data is sampled on the rising edge of the Receive Clock input ($\overline{\text{RxC}}$). Transmitter data transitions occur on the falling edge of the Transmit Clock input ($\overline{\text{TxC}}$).

The differences between Monosync, Bisync, and External Sync are in the manner in which initial receive character synchronization is achieved. The mode of operation must be selected before sync characters are loaded, because the registers are used differently in the various modes. Figure 10 shows the formats for all three synchronous modes.

MONOSYNC. In the Monosync mode (8-bit sync mode), the transmitter transmits the sync character in Sync Word Register 1. The receiver compares the single sync character with the programmed sync character stored in Sync Word Register 2. A match implies character synchronization and enables data transfer. The $\overline{\text{SYNC}}$ pin is used as an output in this mode and is active for the part of the receive clock that detects the sync character.

BISYNC. In the Bisync mode (16-bit sync mode), the transmitter transmits the sync character in Sync Word Register 1 followed by the sync character in Sync Word Register 2. The receiver compares the two contiguous sync characters with the programmed sync characters stored in Sync Word Registers 1 and 2. A match implies character synchronization and enables data transfer. The $\overline{\text{SYNC}}$ pin is used as an output in this mode and is active for the part of the receive clock that detects the sync characters.

External Sync. In the External Sync mode, the transmitter transmits the sync character in Sync Word Register 1. Character synchronization for the receiver is established externally. The $\overline{\text{SYNC}}$ pin is an input that indicates that external character synchronization has been achieved. After the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{\text{SYNC}}$ input pin (see figure 11). The $\overline{\text{SYNC}}$ input pin must be held Low until character synchronization is lost. Character assembly begins on the rising edge of the Receive Clock that precedes the falling edge of the $\overline{\text{SYNC}}$ input pin.

In all cases, after a reset (hardware or software), the receiver is in the Hunt phase, during which time the SIO looks for character synchronization. The Hunt phase can begin only when the receiver is enabled, and data transfer can begin only when character synchronization has been achieved. If character synchronization is lost, the Hunt phase can be reentered by setting the Enter Hunt Mode bit in the Receiver Control Register. In the transmit mode, the transmitter always sends the programmed number of sync bits (8 or 16), regardless of the bits per character programmed.

In the Monosync, Bisync, and External Sync modes, assembly of received data continues until the SIO is reset, or until the receiver is disabled (by command or the $\overline{\text{DCD}}$ pin in the Rx Auto Enables mode), or until the CPU sets the Enter Hunt Mode bit.

After initial synchronization has been achieved, the operation of the Monosync, Bisync, and External Sync modes is quite similar. Any differences are specified in the following text.

To set up the SIO for Synchronous operations, the following registers need to be initialized : Mode

Control Register, Interrupt Control Register, Receiver Control Register, Transmitter Control Register, Sync Word 1, and Sync Word 2. The Mode Control Register must be programmed before other registers to assure proper operation of the SIO. The following registers are used to transfer data or communicate status between the SIO and the CPU or other bus master : Command Register, Status Register 0, Status Register 1, Data Register, and the Vector Register.

The SIO provides four I/O lines in Synchronous modes that may be used for modem control, for external interrupts, or as general purpose I/O. The Request To Send (RTS) and Data Terminal Ready (DTR) pins are outputs that follow the inverted state of their respective bits in the Transmit Control Register. The Data Carrier Detect (DCD) and Clear To Send (CTS) pins are inputs that can be used as auto enables to the receiver and transmitter, respectively. If External/Status Interrupts are enabled, the DCD and CTS pins will be monitored for a change of status. If these inputs change for a period of time greater than the minimum specified pulse width, an interrupt will be generated.

In the following discussion, all interrupt modes are assumed enabled.

SYNCHRONOUS TRANSMIT

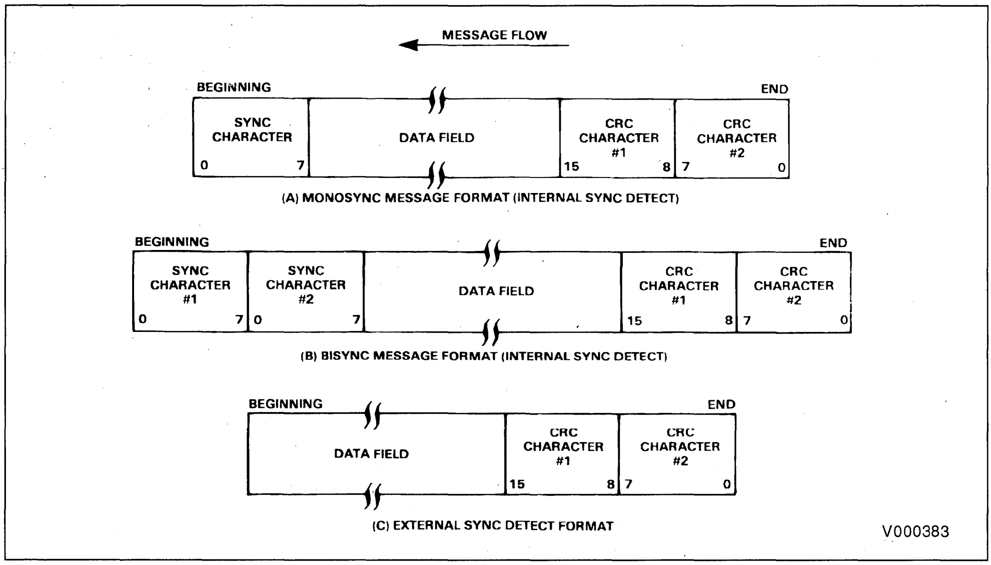
Initialization. Byte-oriented transmitter programs are usually initialized with the following parameters : odd-even or no parity, x1 clock mode, 8- or 16-bit sync character(s), CRC polynomial, Transmit Enables, interrupt modes, and transmit character length. If Parity is enabled, the transmitter will only add a parity bit to a character that is loaded into the transmit buffer ; it will not add a parity bit to the automatically inserted sync character(s) or the CRC characters.

One of two polynomials may be used with Synchronous modes, CRC-16 ($X^{16} + X^{15} + X^2 + 1$) or SDLC-CRC ($X^{16} + X^{12} + X^5 + 1$). For either polynomial (SDLC mode not selected), the CRC generator and checker are reset to all zeros. Both the receiver and transmitter use the same polynomial.

After reset (hardware or software), or when the transmitter is not enabled, the Transmit Data (TxD) output pin is held High (marking). Under program control, the Send Break bit in the Transmitter Control Register can be set to a one, forcing the TxD output pin to a Low level (spacing), even if the transmitter is not enabled. The spacing condition will persist until the Send Break bit is reset to a zero. A programmed break is effective as soon as it is written into the Transmit Control Register ; any characters in the transmit buffer and transmit shift register are lost.

If the transmit buffer is empty when the Transmit Enable bit is set to a one, the transmitter will start sending 8- or 16-bit sync characters. Continuous syncs will be transmitted on the TxD output pin, as long as no data is loaded into the transmit buffer. Note, if a

Figure 10 : Synchronous Formats.


MESSAGE FLOW
BEGINNING
END
SYNC CHARACTER
0
7
DATA FIELD
CRC CHARACTER #1
15
8
CRC CHARACTER #2
7
0
(A) MONOSYNC MESSAGE FORMAT (INTERNAL SYNC DETECT)
SYNC CHARACTER #1
SYNC CHARACTER #2
(B) BISYNC MESSAGE FORMAT (INTERNAL SYNC DETECT)
(C) EXTERNAL SYNC DETECT FORMAT
V000383

character is loaded into the transmit buffer before enabling the transmitter, that character will be sent in place of the sync character(s).

Start of Transmission. Transmission will begin with the loading of the first data character into the transmit buffer, if the transmitter is already enabled. For CRC to be calculated correctly on each message, the CRC generator must be reset to all zeros before the first data character is loaded into the transmit buffer. This is accomplished by issuing a Reset Tx CRC Generator command in the Command Register.

Synchronous Transmit Characteristics. In all Synchronous modes, characters are sent with the least-significant bits first. All data is shifted out of the Transmit Data pin (TxD) on the falling edge of the Transmit Clock (TxC). The transmitter can transmit from one to eight data bits per character. This requires right-hand justification of data written to the transmit buffer, if the selected word length is less than eight bits per character. When the programmed character length is six or seven bits, the unused bits in the transmit buffer are ignored. If a word length of five bits per character or less is selected, the data loaded into the transmit buffer must be formatted as described in the Transmit Control register part of the Register Description section.

The number of bits per character to be transmitted can be changed on the fly. Any data written to the transmit buffer, after the bits per character field is changed, are affected by the change. The same is true of any characters in the buffer at the time the bits per character field is changed. The change in the number of bits per character does not affect the character in the process of being shifted out.

A transmitted message can be terminated by CRC and sync characters, by sync characters only, or by pad characters (replacing the sync character(s) in the Sync Word Registers with pad characters). How a message is terminated is controlled by the Tx Underrun/EOM latch in Status Register 0.

Figure 11a : External Sync Timing.

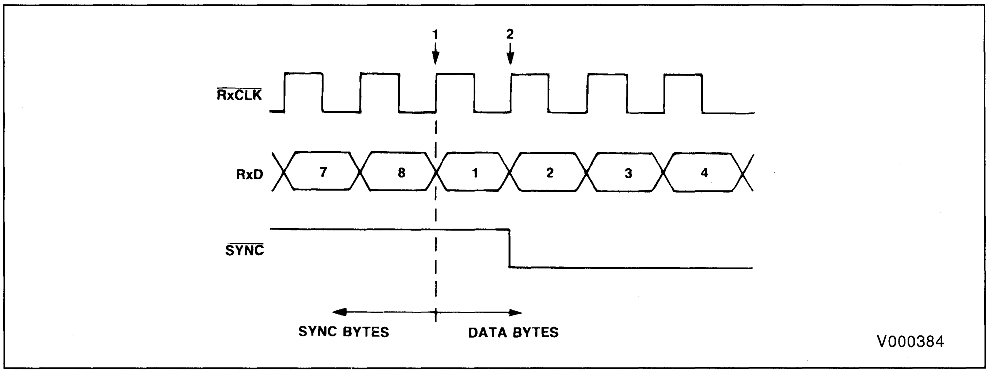
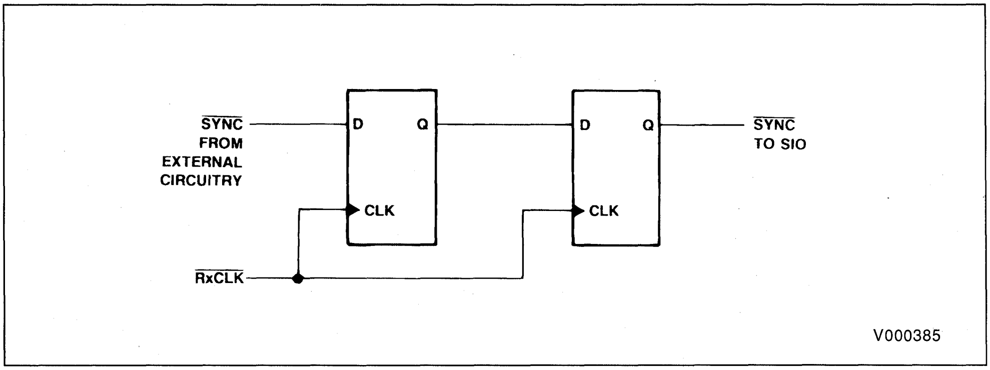


Figure 11b : Simple External Sync Delay.



Data Transfer. A Transmit Interrupt is generated each time the transmit buffer becomes empty. The interrupt may be satisfied either by writing another character into the transmit buffer or by resetting the Transmit Interrupt Pending latch with a Reset Tx Interrupt Pending command. If the interrupt is satisfied with this command, and nothing more is written into the transmit buffer, there can be no further Transmit Interrupts due to a Buffer Empty condition, because it is the process of the buffer becoming empty that causes the interrupts. This situation does cause a Transmit Underrun condition when the data in the shift register is shifted out.

Another way of detecting when the transmitter requires service is to poll the Tx Buffer Empty bit in Status Register 0. This bit is set to a one every time the data in the transmit buffer is downloaded into the transmit shift register. When data is written to the transmit buffer, this bit is reset to zero.

The SIO has all the signals and controls necessary to implement a DMA transfer routine for the transmitter. The routine may be configured to enable the DMA controller, after the first character is written to the transmit buffer, and then using the $\overline{\text{TxRDY}}$ output pin to signal the DMA that the transmitter requires service. If a data character is not loaded into the transmit buffer by the time the transmit shift register is empty, the SIO enters the Transmit Underrun condition.

Transmit Underrun/End of Message. When the transmitter has no further data to transmit, the SIO inserts filler characters to maintain synchronization. The SIO has two programmable options for handling this situation : sync characters can be inserted, or the CRC characters generated so far can be sent, followed by sync characters. These options are controlled by the state of the Transmit Underrun/EOM Latch in Status Register 0.

Following a hardware or software reset, the Transmit Underrun/EOM Latch is set to a one. This allows sync characters to be inserted when there is no data to send. CRC is not calculated on the automatically inserted sync characters. To allow CRC characters to be sent when the transmitter has no data, the Transmit Underrun/EOM Latch must be reset to zero. This latch is reset by issuing a Reset Tx Underrun/EOM Latch command in the Command Register. Following the CRC characters, the SIO sends sync characters to terminate the message.

There is no restriction as to when, in the message, the Transmit Underrun/EOM Latch can be reset, but once the reset command is issued, the 16-bit CRC is sent and followed by sync characters the first time the transmitter has no data to send. A Transmit Underrun condition will cause an External/Status Interrupt to be generated whenever the Transmit Underrun/EOM Latch is set.

For sync character insertion only, at the termination of a message, a Transmit Interrupt is generated only after the first automatically inserted sync character is loaded into the transmit shift register. The status bits in Status Register 0 indicate that the Transmit Underrun/EOM Latch and the Tx Buffer Empty bit are set.

For CRC insertion, followed by sync characters, at the termination of a message, the Transmit Underrun/EOM Latch is set, and the Tx Buffer Empty bit is reset while the CRC characters are being sent. When the CRC characters are completely transmitted, the Tx Buffer Empty status bit is set, and a Transmit Interrupt is generated, indicating to the CPU that another message can begin. This Transmit Interrupt occurs when the first sync character following the CRC characters is loaded into the transmit shift register. If no more messages are to be transmitted, the program can terminate transmission by disabling the transmitter.

CRC Generation. Setting the Tx CRC Enable bit in the Transmit Control Register initiates CRC accumulation when the program sends the first data character to the SIO. To ensure CRC is calculated correctly on each message, the Reset Tx CRC Generator command should be issued before the first data character of the message is sent to the SIO.

The Tx CRC Enable bit can be changed on the fly at any point in the message to include or exclude a particular data character from CRC accumulation. The Tx CRC Enable bit should be in the desired state when the data character is loaded from the transmit data buffer into the transmit shift register. To ensure this bit is in the proper state, the Tx CRC Enable bit should be loaded before sending the data character to the SIO.

Transmit Termination. The SIO is equipped with a special termination feature that maintains data integrity and validity. If the transmitter is disabled (by resetting the Transmit Enable bit or using the Tx Auto Enable signal) while a data or sync character is being transmitted, the character is transmitted as usual but is followed by a marking line instead of sync or CRC characters. When the transmitter is disabled, a character in the transmit buffer remains in the buffer. If the transmitter is disabled while CRC characters are being transmitted, the 16-bit transmission is completed, but the remaining bits of the CRC characters are replaced by sync characters.

Bisync Protocol Transmission. In a Bisync Protocol operation, once synchronization is achieved between the transmitter and receiver, fill characters are inserted to maintain that synchronization when the transmitter has no more data to send. The different options available in the SIO are described in the Transmit Underrun/End Of Message part of this section. If pad characters are to be sent in place of sync characters following the transmission of the CRC, the program can set the SIO transmitter to eight bits per character and then load "FFH" to the transmit buffer while the CRC characters are being sent. Alternatively, the sync characters in Sync Word Registers 1 and 2 can be redefined to be pad characters during this time. The following example is included to clarify this point :

The SIO interrupts the CPU with a Transmit Interrupt when the Tx Buffer Empty bit is set.

The CPU recognizes that the last character (ETX) of the message has already been sent to the SIO transmit buffer by examining the internal program status.

To force the SIO to send CRC, the CPU issues the Reset Tx Underrun/EOM Latch command and clears the current Transmit Interrupt with the Reset Tx Interrupt Pending command. Resetting the interrupt with this command prevents the SIO from requesting more data. The SIO then begins to send CRC (because the transmitter is in an underrun condition) and sets the Transmit Underrun/EOM Latch, which causes an External/Status Interrupt.

The CPU satisfies the External/Status Interrupt by loading pad characters into the transmit buffer and clears the interrupt by issuing the Reset External/Status Interrupt command.

The pad character will follow the CRC characters in this sequence, instead of the usual sync characters. A Transmit Interrupt is generated when the pad character is loaded into the transmit shift register.

From this point on, the CPU can send more pad characters or sync characters.

The transparent mode of operation in Bisync Protocol is made possible with the SIO's ability to change the Tx CRC Enable bit at any time during program sequencing and with the additional capability of inserting 16-bit sync characters. Exclusion of DLE (Data Link Escape) characters from CRC calculation can be achieved by disabling CRC calculations immediately preceding the DLE character transfer to the transmit buffer. In the case of a transmit underrun condition in the transparent mode, a pair of DLE-SYN characters is sent. The SIO can be programmed to send the DLE-SYNC sequence by loading a DLE character into Sync Word Register 1 and a SYNC character into Sync Word Register 2.

The SIO always transmits two sync characters (16 bits) in Bisync mode. If additional sync characters are to be transmitted before a message, the CPU can delay loading data to the transmit buffer until the required number of syncs have been sent. No CRC calculations are done on any automatically inserted sync characters. An alternate method of sending additional sync characters is to load the sync characters into the transmit buffer, in which case the transmitter will treat the characters as data. The Tx CRC Enable bit should not be set, until true data is going to be loaded into the buffer, to avoid performing CRC calculations on the additional sync characters.

SYNCHRONOUS RECEIVE

Initialization. Byte-oriented receive programs are usually initialized with the following parameters : odd-even or no parity, x1 clock mode (necessary because of the start bit detection logic), 8- or 16-bit sync character(s), CRC polynomial, Receiver Enables, interrupt modes, and receive character length. Care must be taken if Parity is enabled. The receiver will usually detect a Parity Error on all sync characters, after synchronization is achieved, and on the CRC characters.

Receiver Hunt Mode. After the SIO is initialized for a Synchronous receive operation, the receiver is in the Hunt phase. During the Hunt phase, the receiver does a bit-by-bit comparison of the incoming data stream and the sync character(s) stored in the Sync Word Register 2 (for Monosync mode) and Sync Word Registers 1 and 2 (for Bisync mode). When a match occurs, the Hunt phase is terminated, and the following data bits are assembled into the programmed character length and loaded into the receive data FIFO.

Receive Characteristics. The receiver may be programmed to assemble five to eight data bits into a character. The character is right-justified in the shift register and transferred to the receive data FIFO. All data transfers to the FIFO are in 8-bit groups. When the programmed character length is less than eight bits, the most significant bit(s) transferred with a character will be the least significant bit(s) of the next character. The programmed character length may be changed on the fly during a message ; however, care must be taken to assure that the change is effective before the number of bits specified for the character length have been assembled.

When the Sync Character Load Inhibit bit in the Receiver Control Register is set, all characters in the receive data stream that match the byte loaded into Sync Word Register 1 will be inhibited from loading into the receive data FIFO. The comparison between Sync Word Register 1 and the incoming data occurs at a character boundary time. This is an 8-bit comparison, regardless of the bits per character programmed. CRC calculations will be performed on all bytes, even if the characters are not transferred to the receive data FIFO, as long as the Rx CRC Enable bit is set.

Data Transfer and Status Monitiring. After character synchronization is achieved, the assembled characters are transferred to the receive data FIFO, and the status information for each character is transferred to the receive error FIFO. The following four modes are available to transfer the received data and its associated status to the CPU.

No Receive Interrupts Enabled. This mode is used either for polling operations or for off-line conditions. When transferring data, using a polling routine, the Rx Character Available bit in Status Register 0 should be checked to determine if a receive character is available for transfer. Only when a character is available should the receive buffer and Status Register 1 be read. The Rx Character Available bit is set when a character is loaded to the top of the receive data FIFO. This bit is reset during a read of the receive buffer.

Interrupt On First Character Only. This interrupt mode is normally used to start a DMA transfer routine or, in some cases, a polling loop. The SIO will generate an interrupt the first time a character is shifted to the top of the receive data FIFO after this mode is selected or reinitialized. An interrupt will be generated thereafter only if a Special Receive Condition is detected. This mode is reinitialized with the Enable Interrupt On Next Receive Character command. Parity Errors do not cause interrupts in this mode ; however, a Receive Overrun Error will.

Interrupt On Every Character. This interrupt mode will generate a Receiver Interrupt every time a character is shifted to the top of the receive data FIFO. A Special Receive Condition interrupt on a parity error is optional in this mode.

Special Receive Condition Interrupt. The special condition interrupt mode is not an interrupt mode as such, but works in conjunction with Interrupt On Every Character or Interrupt On First Character Only modes. When the Status Affects Vector bit in either channel is set, a Special Receive condition will modify the Receive Interrupt vector to signal the CPU of the special condition. Receive Overrun Error and Parity Error are the only Special Receive Conditions in Synchronous receive mode. The overrun and parity error status bits in Status Register 1 are latched when they occur ; they will remain latched until an Error Reset command is issued. As long as either one of these bits is set, a Special Receive Condition Interrupt will be generated at every character available time. Since these two status bits are latched, the error status in Status Register 1, when read, will reflect an error in the current word in the receive buffer, in addition to any Parity or Overrun errors received since the last Error Reset command.

CRC Error Checking and Receiver Message Termination. A CRC error check on the received message can be performed on a per character basis under program control. The Rx CRC Enable bit must set/reset by the program before the next character is transferred from the receive shift register to the receive data FIFO. This ensures proper inclusion or exclusion of data characters in the CRC check.

There is an 8-bit delay between the time a character is transferred to the receive data FIFO and the time the same character starts to enter the CRC checker. An additional 8-bit times are needed to perform CRC calculations on the character. Due to this serial nature of CRC calculations, the Receive Clock (RxC) must cycle 16 times after the second CRC character has been loaded into the receive data FIFO or 20 times (the previous 16 plus 3-bit buffer delay and 1-bit input delay) after the last bit is at the RxD input, before CRC calculation is complete. The CRC Framing Error bit in Status Register 1 will contain the comparison results of the CRC checker. The comparison results should be zero, indicating error-free transmission. The results in the status bit are valid only at the end of CRC calculation. If the result is examined before this time, it usually indicates an error (the bit is High). The comparison is made at each character available time and is valid until the character is read from the receive data FIFO.

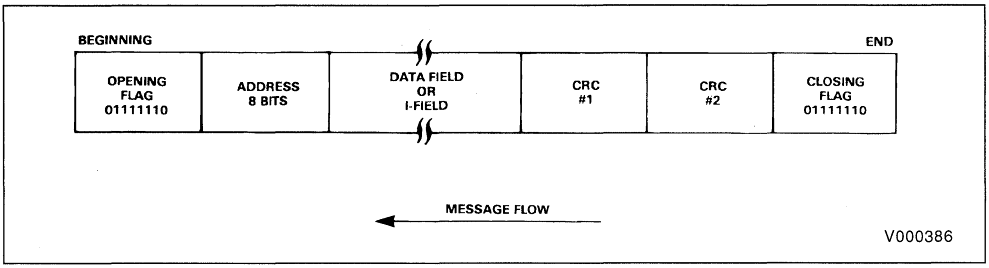
SDLC/HDLC OPERATION

INTRODUCTION

The MK68564 SIO is capable of handling both High-level Synchronous Data Link Control (HDLC) and IBM Synchronous Data Link Control (SDLC) protocols. In the following discussion, only SDLC is referenced because of the high degree of similarity between SDLC and HDLC.

The SDLC mode is considerably different from Monosync and Bisync protocols, because it is bit oriented rather than character oriented. Bit orientation makes SDLC a flexible protocol in terms of mes-

Figure 12 : Transmit/Receive SDLC/HDLC Message Format.



sage length and bit patterns. The SIO has several built-in features to handle variable message length. Detailed information concerning SDLC protocol can be found in literature on this subject, such as IBM document GA27-3093.

The SDLC message, called the frame (figure 12), is opened and closed by flags, which are similar to the sync characters used in other Synchronous protocols. The SIO handles the transmission and recognition of the flag characters that mark the beginning and end of the frame. Note that the SIO can receive shared-zero flags but cannot transmit them. The 8-bit address field of a SDLC frame contains the secondary station address. The SIO receiver has an Address Search mode, which recognizes the secondary station so that it can accept or reject a frame.

The control field of the SDLC frame is transparent to the SIO ; it is simply transferred to the CPU. The SIO handles the Frame Check sequence in a manner that simplifies the program by incorporating features such as initializing the CRC generator to all ones, resetting the CRC checker when the opening flag is detected in the receive mode, and sending the Frame Check/Flag sequence in the transmit mode. Controller hardware is simplified by automatic zero insertion and deletion logic, contained in the SIO.

To set up the SIO for SDLC operation, the following registers need to be initialized : Mode Control Register, Interrupt Control Register, Receiver Control Register, Transmitter Control Register, Sync Word Register 1, and Sync Word Register 2. The Mode Control Register must be programmed before the other registers to assure proper operation of the SIO. The following registers are used to transfer data or communicate status between the SIO and the CPU or other bus master when operating in SDLC mode : Command Register, Status Register 0, Status Register 1, Data Register, and the Vector Register.

Sync Word Register 1 contains the secondary station address, and Sync Word Register 2 stores the flag character and must be programmed to "01111110".

The SIO provides four I/O lines in SDLC mode that may be used for modem control, for external interrupts, or as general purpose I/O. The Request To Send (RTS) and Data Terminal Ready (DTR) pins are outputs that follow the inverted state of their respective bits in the Transmit Control Register. The Data Carrier Detect (DCD) and Clear To Send (CTS) pins are inputs that can be used as auto enables to the receiver and transmitter, respectively. If External/Status Interrupts are enabled, the DCD and CTS pins will be monitored for a change of status. If these inputs change for a period of time greater than the minimum specified pulse width, an interrupt will be generated.

In the following discussion, all interrupt modes are assumed enabled.

SDLC TRANSMIT

Initialization. The SIO is initialized for SDLC mode by selecting these parameters in the Mode Control Register : x1 Clock Mode, SDLC Mode, and Sync Modes Enabled. Parity is normally not used in SDLC mode, because the transmitter will not add parity to the flag character or the CRC characters, thus causing Parity Errors in the receiver. If CRC is to be calculated on the transmitted data, the SDLC-CRC polynomial must be selected in the Interrupt Control Register (CRC-16 polynomial in SDLC Mode will produce unknown results).

After reset (hardware or software), or when the transmitter is not enabled, the Transmit Data (TxD) output pin is held High (marking). Under program control, the Send Break bit in the Transmit Control Register can be set to a one, forcing the TxD output to a Low level (spacing), even if the transmitter is not enabled. The spacing condition will persist until the Send Break bit is reset to a zero. If the transmit buffer is empty when the Transmit Enable bit is set to a one, the transmitter will start sending flag characters. Continuous flags will be transmitted on the TxD output pin as long as no data is loaded into the transmit buffer.

Note : If a character is loaded into the transmit buffer before enabling the transmitter, that character will be sent in place of a flag.

An abort sequence may be transmitted at any time by issuing the Send Abort command (command 1). This causes at least eight, but less than fourteen, ones to be sent before the output reverts back to continuous flags. It is possible that the Abort sequence (eight 1's) could follow up to five continuous ones (allowed by the zero insertion logic) and, thus, cause as many as thirteen ones to be sent. Any data being transmitted and any data in the transmit buffer is lost when an abort is issued.

The zero insertion logic in the transmitter will automatically insert a 0 after five continuous ones in the data stream. This does not apply to flags or aborts.

Start of Transmission. Transmission will begin with the loading of the first character into the transmit buffer if the transmitter is already enabled. For CRC to be calculated correctly on each frame, the CRC generator must be initialized to all ones before the first character is loaded. This is accomplished by issuing a Reset Tx CRC Generator command in the Command Register. The first non-flag character transmitted is the address field. The SIO does not automatically transmit a station address, this is left to the programmer. The SIO will only transmit flags and CRC characters automatically.

SDLC Transmit Characteristics. Any length SDLC frame can be transmitted. All characters are transmitted with the least-significant bits first. All data is shifted out of the Transmit Data pin (TxD) on the falling edge of the Transmit Clock (TxC). The transmitter transmit from one to eight data bits per character. This requires right-hand justification of data written to the transmit buffer, if the word length selected is less than eight bits per character. When the programmed character length is six or seven bits, the unused bits in the transmit buffer are ignored. If a word length of five bits per character or less is selected, the data loaded into the transmit buffer must be formatted as described in the Transmit Control Register part of the Register Description section.

The number of bits per character to be transmitted can be changed on the fly. Any data, written to the transmit buffer after the bits per character field is changed, are affected by the change. The same is true of any characters in the buffer at the time the bits per character field is changed. The change in the number of bits per character does not affect the character in the process of being shifted out. Flag characters are always eigth bits in length, and CRC is always 16 bits in length, regardless of the programmed bits per character. A transmitted frame can be terminated by CRC and a flag, by a flag only, or by an abort. This is controlled by the Tx Underrun/EOM Latch and the Send Abort command.

Data Transfers. A Transmit Interrupt is generated each time the transmit buffer becomes empty. The interrupt may be satisfied either by writing another character into the transmit buffer or by resetting the Transmit Interrupt Pending latch with a Reset Tx Interrupt Pending command. If the interrupt is satisfied with this command, and nothing more is written into the transmit buffer, there are no further transmitter interrupts, and a Transmit Underrun condition will occur when the data in the shift register is shifted out. When another character is written to the buffer and loaded into the shift register, the transmit buffer can again become empty and interrupt the CPU. Following the flags in an SDLC operation, the 8-bit address field, control field, and information field may be sent to the SIO, using the Transmit Interrupt mode. The SIO transmits the frame check sequence using the Transmit Underrun feature.

When the transmitter is first enabled, the transmit buffer is already empty and obviously cannot then become empty. Therefore, no transmit interrupt can occur until after the first data character is written to the transmit buffer.

Another way of detecting when the transmitter requires service is to poll the Tx Buffer Empty bit in Status Register 0. This bit is set to a one every time the data in the transmit buffer is downloaded into the transmit shift register. When data is written to the transmit buffer, this bit is reset to zero.

The SIO has all the signals and controls necessary to implement a DMA transfer routine for the transmitter. The routine may be configured to enable the DMA controller, after the first character is written into the transmit buffer, using the TxRDY output pin to signal the DMA that the transmitter requires service. The DMA transfer can be terminated, when the DMA block count is reached, using the Tx Underrun/EOM interrupt.

Transmit Underrun/End of Message. SDLC-like protocols do not have provisions for fill characters within a message. The SIO, therefore, automatically terminates an SDLC frame when the transmit data buffer is empty, and the output shift register has no more bits to send. It does this by first sending the two bytes of CRC and the following these with one or more flags. This technique allows very high-speed transmission under DMA or CPU control, without requiring the CPU to respond quickly to the end of message situation.

The action that the SIO takes in the underrun situation depends on the state of the Transmit Underrun/EOM status bit in status Register 0. Following a reset, the Transmit Underrun/EOM bit is set to a one and prevents the insertion of CRC characters during the time there is no data to send. Consequently, flag characters are sent. If the Transmit Underrun/EOM status bit is zero when the underrun condition occurs, the 16-bit CRC character is sent, followed by one or more flag characters. The Transmit Underrun/EOM bit is reset to zero by issuing the Reset Tx Underrun/EOM Latch command in the Command Register.

The SIO begins to send a frame when data is written into the transmit buffer. Between the time the first data byte is written and the end of the message, the Reset Tx Underrun/EOM Latch command must be issued. The Transmit Underrun/EOM status bit will then be in the reset state at the end of the message (when underrun occurs), and CRC characters will automatically be sent. The transmission of the first CRC bit set the Transmit Underrun/EOM status bit to a one and generates an External/Status interrupt. Also, while CRC is being sent, the Tx Buffer Empty bit in Status Register 0 is reset to indicate that the transmit shift register is full of CRC data. When CRC has been completely sent, the Tx Buffer Empty status bit is set, and a Transmit Interrupt is generated to indicate that another message may begin. This interrupt occurs because CRC has been sent, and a flag has been loaded into the shift register. If no more messages are to be sent, the program can terminate transmission by disabling the transmitter.

Although there is no restriction as to when the Transmit Underrun/EOM bit can be reset within a message, it is usually reset after the first data character (secondary address field) is sent to the SIO. By resetting the status bit early in the message, the CPU has additional time (16 bits of CRC) to recognize if an unintentional transmit underrun situation has occured and to respond with an Abort command. Issuing the Abort command stops the flags from going on the line prematurely and eliminates the possibility of the receiver accepting the frame as valid data. This situation can happen if, at the receiving end, the data pattern immediately preceding the automatic flag insertion matches the CRC checker, giving a false CRC check result.

CRC Generation. The CRC generator must be reset to all ones at the beginning of each frame before CRC accumulation can begin. Actual accumulation begins on the first data character (address field) loaded into the transmit buffer. The Tx CRC Enable bit in the Transmit Control Register should be set to a one before the first character is loaded into the transmit buffer. In SDLC mode, all characters between the opening and the closing flags are included in CRC accumulation. The output of te CRC generator is inverted before it is transmitted.

Transmit Termination. The normal sequence at the end of a frame is

A Transmit Interrupt occurs when the last data character written to the transmit buffer is downloaded into the transmit shift register. This interrupt may be cleared by issuing a Reset Tx Interrupt Pending command.

An External/Status Interrupt occurs when the first bit of the CRC character is transmitted. This interrupt condition should first be tested to see if the interrupt was caused by the Tx Underrun/EOM bit going High and then reset by issuing a Reset External/Status Interrupts command.

A Transmit Interrupt occurs when the first bit of the flag is transmitted. This interrupt may be cleared by issuing a Reset Tx Interrupt Pending command, by loading the first character of the next message, or by disabling the transmitter.

If the transmitter is disabled while a character is being sent, that character (data or flag) is sent in the normal fashion but is followed by a marking line rather than CRC or more flag characters. If CRC characters are being sent at the time the transmitter is disabled, all 16 bits will be transmitted, followed by a marking line ; however, flags are sent in place of CRC. A character in the buffer when the transmitter is disabled remains in the buffer.

SDLC RECEIVE

Initialization. The receiver is enabled only after all of the receive parameters are initialized. After the Receiver Enable bit in the Receiver Control Register is set to a one, the receiver will be in the Hunt phase and will remain in this phase until the first flag is received. While in the SDLC mode, the receiver never re-enters the Hunt phase, unless specifically instructed to do so by the program or when an Abort character is detected in the incoming data stream.

Receiver Characteristics. The receiver may be programmed to assemble five to eight data bits into a character. The character is right-justified in the shift register and transferred to the receive data FIFO. All data transfers to the FIFO are in 8-bit groups. When the character length programmed is less than eight bits, the most significant bit(s) transferred with a character, will be the least-significant bit(s) of the next character. The character length programmed may be changed on the fly during the reception of a frame ; however, care must be taken to assure that the change is effective, before the number of bits specified for the character length has been assembled.

The address field in the SDLC frame is defined as an 8-bit field. When the Address Search Mode is selected, the receiver will compare the 8-bit character following the flag (first non-flag character) against the address programmed in Sync Word Register 1 or the hardwired global address (11111111). When the address field of the SDLC frame matches either address, data transfer will begin with the address character being loaded into the receive data FIFO. If the frame address does not match either address, the receiver will remain idle and continue checking every frame received for an address match. The address comparison is always done on the first eight bits following a flag, regardless of the bits per character programmed.

The SIO receiver is capable of matching only one address character. Once a match occurs, all data is transferred to the receive data FIFO at the programmed bits per character rate. If SDLC extended address field recognition is used (two or more address characters), the CPU program must be capable of determining whether or not the frame has a correct address field. If the correct address field is not received, the Hunt bit can be set to suspend reception and start searching for the next frame. The control field of an SDLC frame is transparent to the SIO ; it is transferred to the data FIFO as a data character. All extra zeros, inserted in the data stream by the transmitter, are automatically deleted in the receiver.

Data Transfer and Status Monitoring. After receipt of a valid flag, the assembled characters are transferred to the receive data FIFO, and the status information for each character is transferred to the receive error FIFO. The following four modes are available to transfer the received data and its associated status to the CPU.

No Receiver Interrupts Enabled. This mode is used for polling operations or for off-line conditions. When transferring data, using a polling routine, the Rx Character Available bit in Status Register 0 should be checked to determine whether or not a receive character is available for transfer. Only when a character is available should the receive buffer and Status Register 1 be read. The Rx Character Available bit is set to a one every time a character is shifted to the top of the receive data FIFO. This bit is reset when the receive buffer is read.

Interrupt On First Character Only. This interrupt mode is normally used to start a DMA transfer routine, or in some cases, a polling loop. The SIO will generate an interrupt the first time a character is shifted to the top of the receive data FIFO after this mode is selected or reinitialized. An interrupt will be generated thereafter only if a Special Receive Condition is detected. This mode is reinitialized with the Enable Interrupt On Next Received Character command. Parity Errors do not cause interrupts in this mode, but a Receive Overrun Error or an End Of Frame condition will.

Interrupt On Every Character. This interrupt mode will generate a Receiver Interrupt every time a character is shifted to the top of the receive data FIFO. A Special Receive Condition interrupt on a Parity error is optional in this mode.

Special Receive Condition Interrupt. The special condition interrupt mode is not an interrupt mode, as such, but works in conjunction with Interrupt On Every Character or Interrupt On First Character Only modes. When the Status Affects Vector bit in either channel is set, a Special Receive Condition will modify the Receive Interrupt vector to signal the CPU of the special condition. Receive Overrun Error, Parity Error, and End Of Frame are the Special Receive Conditions in SDLC mode. The Overrun and Parity error status bits in Status Register 1 are latched when they occur ; the End Of Frame bit is not latched. The two bits that are latched will remain latched and will generate a Special Receive Condition Interrupt at every character available time until an Error Reset command is issued. Since the two status bits are latched, the error status in Status Register 1, when read, will reflect an error in the current word in the receive buffer, in addition to any Parity or Overrun errors received since the last Error Reset command.

SDLC Receive CRC Checking. Control of the receiver CRC checker is automatic. It is reset by the leading flag, and CRC is calculated up to the final flag. The byte that has the End Of Frame bit set is the byte that contains the result of the CRC check. If the CRC/Framing Error bit is not set (zero), the CRC indicates a valid received message. A special check sequence is used for the SDLC check, because the transmitted CRC character is inverted. The final check must be 0001110100001111. he 2-byte CRC check characters should be read and discarded by the CPU, because the last two bits of the 2-byte SDLC CRC check characters are not transferred to the receive data FIFO due to the internal timing associated with detecting the closing flag.

Unlike Synchronous modes, the logic path in SDLC mode does not have an 8-bit delay between the time a character is transferred to the receive data FIFO and the time a character enters the CRC checker. This delay is not needed, because in SDLC, all characters between the opening and closing flags are included in the CRC calculations. When the second CRC character (six bits only) is loaded into the receive buffer, CRC calculation is complete.

SDLC Receive Termination. An SDLC frame is terminated when the closing flag is detected. The detection of the flag sets the End Of Frame bit in Status Register 1 and generates a Special Receive Condition Interrupt. In addition to the End Of Frame bit being set and the results of the CRC check, Status Register 1 has three bits of Residue code valid at this time. The Residue bits indicate the boundary between the CRC check bits and the I-field bits in the frame. A detailed description of the Residue code bits is given in the Register Description section, under Status Register 1.

Any frame can be prematurely aborted by an Abort sequence. Aborts are detected if seven or more continuous ones occur in the received data stream. This condition will cause an External/Status Interrupt to be generated with the Break/Abort bit in Status Register 0 set. After the Reset External/Status Interrupts command has been issued, a second interrupt will occur when the continuous ones condition has been cleared. This second interrupt can be used to distinguish between the Abort and Idle line conditions.

REGISTER DESCRIPTION

The following sections describe the MK68564 SIO registers. Each register is detailed in terms of bit configuration, the active states of each bit, their definitions, their functions, and their effects upon the internal hardware and external pins.

COMMAND REGISTER (CMDREG)

This register contains command and reset functions used in the programming of the SIO. This register is reset to "00H" by a channel or hardware reset. All bits, except Loop Mode, will be read as zeros during a read cycle.

D7	D6	D5	D4	D3	D2	D1	D0
CRC 1	CRC 0	CMD 2	CMD 1	CMD 0			LOOP MODE

D7, D6 : Reset Codes 1 and 0

CRC 1	CRC 0	
0	0	Null Code (no effect)
0	1	Reset Receiver CRC Checker
1	0	Reset Transmit CRC Generator
1	1	Reset Tx Underrun/End of Message Latch

Null Code. The null code has no effect on the MK68564 SIO. It is used when writing to the Command Register for some reason other than a CRC Reset.

Reset Receiver CRC Checker. It is necessary in Synchronous modes (except SDLC) to reset the receiver CRC circuitry between received messages. The CRC circuitry may be reset by one of the following : disabling the receiver, setting the Enter Hunt Mode bit in the Receiver Control Register, or issuing this Reset command. The CRC circuitry is reset automatically in SDLC mode when the End Of Frame flag is detected. This Reset command will initialize the CRC checker circuit to all ones in SDLC mode and all zeros in the other Synchronous modes.

Reset Transmit CRC Generator. This command resets the CRC generator to all ones in SDLC mode and all zeros in the other Synchronous modes. This command should be issued after the transmitter is enabled but before the first character of a message is loaded in the transmit buffer.

Reset Transmit Underrun/EOM Latch. This command resets the Underrun/EOM latch in Status Register 0 if the transmitter is enabled. The Underrun/EOM latch controls the transmission of CRC at the end of a message in Synchronous modes. When a transmit underrun occurs and this latch is low, CRC will be appended to the end of the transmission.

D5, D4, D3 : Command Codes

Command	CMD2	CMD1	CMD0	
0	0	0	0	Null Command (no effect)
1	0	0	1	Send Abort (SDLC mode)
2	0	1	0	Reset External/ Status Interrupts
3	0	1	1	Channel Reset
4	1	0	0	Enable Interrupt On Next Rx Character
5	1	0	1	Reset Tx Interrupt Pending
6	1	1	0	Error Reset
7	1	1	1	Null Command (no effect)

Command 0 (Null). The Null command has no effect on the MK68564 SIO.

Command 1 (Send Abort). This command is used in SDLC mode to transmit a sequence of eight to thirteen ones. This command always empties the transmit buffer ans sets the Tx Underrun/EOM Latch in Status Register 0 to a one

Command 2 (Reset External/Status Interrupts). After an External/Status interrupt (a change on a modem line or a Break condition, for example), the upper five bits in Status Register 0 are latched. This command reenables these bits and allows interrupts to occur again as a result of a status change. Latching the status bits captures short pulses, until the CPU has time to read the change. This command should be issued prior to enabling External/Status Interrupts.

Command 3 (Channel Reset). This command disables both the receiver and transmitter, forces TxD to a marking state ("1"), forces the modem control signals high, resets any pending interrupts from this channel, and resets all control registers. See the Reset section in the SIO System Interface Description for a more detailed list. All control registers for the channel must be rewritten after a Channel Reset command.

Command 4 (Enable Interrupt On Next Rx Character). This command is used to reactivate the Receive Interrupt On First Character Only interrupt mode. This command is normally issued after the present message is completed but before the next message has started to be assembled. The next character to enter the receive data FIFO after this command is issued will cause a receiver interrupt request.

Note : If the data FIFO has more than one character stored when this command is issued, the first previously stored character will cause the receiver interrupt request.

Command 5 (Reset Tx Interrupt Pending). When the Transmit Interrupt Enable mode is selected, the transmitter requests an interrupt when the transmit buffer becomes empty. In those cases, where there are no more characters to be sent (at the end of message, for example), issuing this command resets the pending transmit interrupt and prevents any further transmitter interrupt requests until the next character has been loaded into the transmit buffer or until CRC has been completely sent.

Command 6 (Error Reset). This command resets the upper seven bits in Status Register 1. Anytime a Special Receive Condition exists when Receive Interrupt On First Character Only mode is selected, the data with the special condition is held in the receive data FIFO until this command is issued.

Command 7 (Null). The Null command has no effect on the MK68564 SIO.

D2, D1 : Not Used (read as zeros)

D0 : Loop Mode

When this bit is set to a 1, the transmitter output is connected to the receiver input and $\overline{\text{TxC}}$ is connected to the receiver clock. $\overline{\text{RxC}}$ and RxD pins are not used by the receiver ; they are bypassed internally. $\overline{\text{RxC}}$ may still be used as the baud rate generator output in Loop Mode.

MODE CONTROL REGISTER (MODECTL)

The Mode Control Register contains control bits that affect both the receiver and the transmitter. This register must be initialized before loading the Interrupt, Tx, and Rx Control Registers, and the Sync Word Registers. This register is reset to "00H" by a channel or hardware reset.

D7	D6	D5	D4	D3	D2	D1	D0
CLOCK RATE 1	CLOCK RATE 0	SYNC MODE 1	SYNC MODE 0	STOP BITS 1	STOP BITS 0	PARITY E/O	PARITY ON/OFF

D7, D6 : Clock Rate 1 and 0

These bits specify the multiplier between the input shift clock rates ($\overline{\text{TxC}}$ x $\overline{\text{RxC}}$) and data rate. The same multiplier is used for both the transmitter and receiver, although the input clock rates may be different. In x16, x32, and x64 clock modes, the receiver start bit detection logic is enabled ; therefore, for Synchronous modes, the x1 clock rate must be specified. Any clock rate may be specified for Asynchronous mode ; however, if the x1 clock rate is selected, synchronization between the receive data and the receive clock must be accomplished externally.

CLOCK RATE 1	CLOCK RATE 0	Multiple	
0	0	x1	Clock Rate = Data Rate
0	1	x16	Clock Rate = 16 x Data Rate
1	0	x32	Clock Rate = 32 x Data Rate
1	1	x64	Clock Rate = 64 x Data Rate

D5, D4 : Sync Modes 1 and 0

These bits select the various options for character synchronization. These bits are ignored, unless Sync modes is selected in the Stop Bits filed of this register.

SYNC MODE 1	SYNC MODE 0	
0	0	8-bit Programmed Sync
0	1	16-bit Programmed Sync
1	0	SDLC Mode (01111110 flag pattern)
1	1	External Sync Mode

D3, D2 : Stop Bits 1 and 0

These bits determine the number of stop bits added to each Asynchronous character that is transmitted. The receiver always checks for one stop bit in Asynchronous mode. A special code (00) signifies that a Synchronous mode is to be selected. 1 1/2 stop bits is not allowed if x1 clock rate is selected, because it will lock up the transmitter.

STOP BIT 1	STOP BIT 0	
0	0	Sync Modes
0	1	1 Stop Bit per Character
1	0	11/2 Stop Bits per Character
1	1	2 Stop Bits per Character

D1 : Parity Even/Odd

If the Parity Enable bit is set, this bit determines whether parity is checked as even or as odd. (1 = even, 0 = odd). This bit is ignored if the Parity Enable bit is reset.

D0 : Parity Enable

If this bit is set to a one, one additional bit position beyond those specified in the bits/character control field is added to the transmitted data and is expected in the receive data. The received parity bit is transferred to the CPU as part of the data character, unless eight bits per character is selected in the Receiver Control Register.

INTERRUPT CONTROL REGISTER (INTCTL)

This register contains the control bits for the various interrupt modes and the DMA handshaking signals. This register is reset to "00H" by a channel or hardware reset.

D7	D6	D5	D4	D3	D2	D1	D0
CRC16/ SDLC	CTX RDY ENABLE	RX RDY ENABLE	RX INT MODE 1	RX INT MODE 0	STATUS AFFECTS	TX INT ENABLE	EXT INT ENABLE

D7 : CRC-16/SDLC-CRC

This bit selects the CRC polynomial used by both the transmitter and receiver. When set to a one, the CRC-16 polynomial (x16 + x15 + x2 + 1) is used ; when reset to a zero, the SDLC-CRC polynomial (x16 + x12 + x5 + 1) is used. If the SDLC mode is selected, the CRC generator and checker are preset to all ones and a special check sequence is used.

The SDLC-CRC polynomial must be selected in SDLC mode. Failure to do so will result in receiver CRC errors. When a Synchronous mode, other than SDLC, is selected, the CRC generator and checker are preset to all zeros (for both polynomials). This bit must be programmed before CRC is enabled in the receiver and transmitter control registers, to assure valid CRC generation and checking. This bit is ignored in Asynchronous modes.

D6 : Tx Ready Enable

When this bit is set to a one, the TxRDY output pin will pulse Low for three clock cycles (CLK) when the transmit buffer becomes empty. When this bit is zero, the TxRDY pin is held High.

D5 : Rx Ready Enable

When this bit is set to a one, the TxRDY output pin will pulse Low for three clockcycles (CLK) when a character is available in the receive buffer. If a Special Receive Condition is detected when the Receive Interrupt On FIrst Character Only interrupt mode is selected, the RxRDY pin will not become active ; instead, a special Receive Condition interrupt will be generated. When this bit is zero, the RxRDY pin will be held High

D4, D3 : Receive Interrupt Modes 1 and 0

Together, these two bits specify the various character-avalaible conditions that will cause interrupt requests. When receiver interrupts are enabled, a Special Receive Condition can cause an interrupt request and modify the interrupt vector. Special Receive conditions are : Rx Overrun Error, Framing Error (in async mode), End Of Frame (in SDLC mode), and Parity Error (when selected). The Rx Overrun Error and the Parity Error conditions are latched in Status Register 1 when they occur ; they are cleared by an Error Reset command (Command 4) or by a hardware or channel rest.

Rx INT MODE 1	Rx INT MODE 0	
0	0	Receive Interrupts Disabled
0	1	Receive Interrupt On First Character Only
1	0	Interrupt On All Receive Characters-parity Error is a Special Receive Condition
1	1	Interrupt On All Receive Characters-parity Error is not a Special Receive Condition

Receive Interrupts Disabled. This mode prevents the receiver from generating an interrupt request and clears any pending receiver interrupts. If a character is avalaible in the receiver data FIFO, or if a Special Receive Condition exists before or during the time receiver interrupts are disabled, and receiver interrupts are then enabled without clearing these conditions, an interrupt request will immediately be generated.

Receive Interrupt On First Character Only. The receiver requests an interrupt in this mode on the first available character (or stored FIFO character), or on a Special Receive Condition. If a Special Receive Condition occurs, the data with the special condition is held in the receive data FIFO until an Error Reset command (Command 6) is issued.

The receive Interrupt On First Character Only mode can be re-enabled by the Enable Interrupt On Next Rx Character command (Command 4). If this interrupt mode was terminated by a Special Receive Condition, the Error Reset command must be issued, before Command 4, for proper operation to resume.

Interrupt On All Receive Characters. This mode ammows an interrupt for every character received (or character in the receive data FIFO) and provides a unique vector (if Status Affects ector is enabled) when a Special Receive Condition exists. When the interrupt request is due to a special condition, the data containing that condition, the data containing data FIFO.

D2 : Status Affects Vector

When this bit is zero, the value programmed into the Vector Register is returned during a read cycle or an interrupt acknowledge cycle. If the Vector Register has not been programmed following a hardware reset, then "0FH" is returned.

When this bit is a one, the vector returned during a read cycle or an interrupt acknowledge cycle is variable. The variable field returned depends on the highest-priority pending interrupt at the start of the cycle.

The Status Affects Vector control bits from both channels are logical "or" ed together ; therefore, if either is programmed to a one, its operation affects both channels. This is the only control bit that functions in this manner on the MK68564.

V2	V1	0	Interrupt Condition
0	0	0	Ch B Transmit Buffer Empty
0	0	1	Ch B External/status Change
0	1	0	Ch B Receive Character Available
0	1	1	Ch B Special Receive Condition*
1	0	0	Ch A Transmit Buffer Empty
1	0	1	Ch A External/status Change
1	1	0	Ch A Receive Character Available
1	1	1	Ch A Special Receive Condition*

* Special Receive Conditions : Parity Error, Rx Overrun Error, Framing Error (Async), End of Frame (SDLC).

D1 : Transmit Interrupt Enable

When this bit is set to a one, the transmitter will request an interrupt whenever the transmit buffer becomes empty. When this bit is zero, no transmitter interrupts will be requested.

D0 : External/Status Interrupt Enable

When this bit is set to a one, an interrupt will be requested by the external/status logic on any of the following occurrences : a transition (high-to-low or low-to-high) on the DCD, CTS, or SYNC input pins, a break/abort condition that has been detected and terminated, or at the beginning of CRC transmission when the Transmit Underrun/EOM latch in Status Register 0 becomes set. When this bit is zero, no External/Status interrupts will occur.

If this bit is set when an External/Status condition is pending, an interrupt will be requested. It is recommended that a Reset External/Status Interrupts command (Command 2 in the Command Register) be issued prior to enabling External/Status interrupts.

SYNC WORD REGISTER 1 (SYNC 1)

This register is programmed to contain the transmit sync character in the Monosync mode, the first eight bits of the 16-bit sync character in the Bysinc mode, or the transmit sync character in the External Sync mode. This register is not used in Asynchronous mode. In the SDLC mode, this register is programmed to contain the secondary address field used to compare against the address field of the SDLC frame. The SIO does not automatically transmit the station address at the beginning of a response frame. This register is reset to "00H" by a channel or hardware reset.

D7	D6	D5	D4	D3	D2	D1	D0
SYNC/ SDLC7	SYNC/ SDLC6	SYNC/ SDLC5	SYNC/ SDLC4	SYNC/ SDLC3	SYNC/ SDLC2	SYNC/ SDLC1	SYNC/ SDLC0

SYNC WORD REGISTER 2 (SYNC 2)

This register is programmed to contain the receive sync character in the Monosync mode, the last eight bits of the 16-bit sync character in the Bisync mode, or a flag character (01111110) in the SDLC mode. This register is not used in the External Sync mode and the Asynchronous mode. This register is reset to "00H" by a channel or hardware reset.

D7	D6	D5	D4	D3	D2	D1	D0
SYNC/ SDLC 15	SYNC/ SDLC 14	SYNC/ SDLC 13	SYNC/ SDLC 12	SYNC/ SDLC 11	SYNC/ SDLC 10	SYNC/ SDLC 9	SYNC/ SDLC 8

RECEIVER CONTROL REGISTER (RCVCTL)

This register contains the control bits and parameters for the receiver logic. This register is reset to "00H" by a channel or hardware reset.

D7	D6	D5	D4	D3	D2	D1	D0
RX BITS CHAR 1	RX BITS CHAR 0	RX AUTO ENAB.	HUNT MODE	RX CRC ENAB.	ADDR. SEARCH	STRIP SYNC	RX ENABLE

D7, D6 : Receiver Bits/Character 1 and 0

The state of these two bits determines the number of bits to be assembled as a character in the received serial data stream. If Parity is enabled, one additional bit will be added to each character. The number of bits per character can be changed while a character is being assembled but only before the number of bits currently programmed is reached. All data is right-justified in the shift register and transferred to the receive data FIFO in 8-bit groups.

In Asynchronous mode, transfers are made at character boundaries, and all unused bits of character are set to a one. In Synchronous modes and SDLC mode, an 8-bit segment of the serial data stream is transferred to the data FIFO when the internal counter reaches the number of bits per character programmed. For less than eight bits per character, no parity, the MSB bit(s) of the first transfer will be the LSB bit(s) of the next transfer.

RX BITS CHAR 1	RX BITS CHAR 0	Bits/character (no parity)	Bits/character (parity)
0	0	5	6
0	1	6	7
1	0	7	8
1	1	8	9

D5 : Receiver Auto Enables

When this bit is set to a one, and the Receiver Enable bit is also set, a Low on the DCD input pin becomes the enable for the receiver. When this bit is zero, the DCD pin is simply an input to the SIO, and its status is displayed in Status Register 0.

D4 : Enter Hunt Mode

This bit, when written to a one, rearms the receiver synchronization logic and forces the comparison of the received bit stream to the ontents of Sync Word Register 1 and/or Sync Word Register 2, depending upon which Synchronous mode is selected, until bit synchronization is achieved. The SIO automatically enters the Hunt mode after a channel or hardware reset, after an Abort condition is detected, or when the receiver is disabled. When the Hunt mode is entered, the Hunt/Sync bit in Status Register 0 is set to a one. When synchronization is achieved, the Hunt/Sync bit is reset to a zero. If External/Status interrupts are enabled, an interrupt request will be generated on both transitions of the Hunt/Sync bit. Enter Hunt Mode has no affect in Asynchronous modes. This bit is not latched and will always be read as a zero.

D3 : Receiver CRC Enable

This bit, when set to a one in a Synchronous mode other than SDLC, is used to initiate CRC calculation at the beginning of the last byte transferred from the receiver shift register to the receive data FIFO. This operation occurs independently of the number of bytes in the receive data FIFO. As long as this bit is set, CRC will be calculated on all characters received (data or sync). When a particular byte is to be excluded from CRC calculation, this bit should be reset to a zero before the next byte is transferred to the receive data FIFO. If this feature is used, care- must be taken to ensure that eight bits per character are selected in the reciever because of an inherent eight-bit delay from the receiver shift register to the CRC checker.

When this bit is set to a one in SDLC mode, the SIO will calculate CRC on all bits between the opening and closing flags. There is no delay from the receiver shift register to the CRC checker in SDLC mode. This bit is ignored in Asynchronous modes.

D2 : Address Search Mode

Setting this bit to a one in SDLC mode forces the comparison of the first non-flag character of a frame with the address programmed in Sync Word Register 1 or the global address (11111111). If a match does not occur, the frame is ignored, and the receiver remains idle until the next frame is detected. No receiver interrupts can occur in this mode, unless there is an address match. This bit is ignored in all modes except SDLC.

D1 : Sync Character Load Inhibit

When this bit is set to a one in any Synchronous mode except SDLC, the SIO compares the byte in Sync Word Register 1 with the byte about to be loaded into the receiver data FIFO. If the two bytes are equal, the load is inhibited, and no receiver interrupt will be generated by this character. CRC calculation is performed on all bytes, whether they are loaded into the data FIFO or not, when the receiver CRC is enabled. Note that the register used in the comparison contains the transmit sync character in Monosync and External sync modes. This bit is ignored in SDLC mode because all flag characters are automatically striped in this mode without performing CRC calculations on them.

If this bit is set to a one in Asynchronous modes, any character received matching the contents of Sync Word Register 1 will not be loaded into the receive data FIFO, and no receiver interrupt will be generated for the character.

D0 : Receiver Enable

When this bit is set to a one, receiver operation begins if Rx Auto Enables mode is not selected. This bit should be set only after all receiver parameters are established, and the receiver is completely initialized. When this bit is zero, the receiver is disabled ; the receiver CRC checker is reset, and the receiver is in the Hunt mode.

TRANSMITTER CONTROL REGISTER (XMTCTL)

This register contains the control bits and parameters for the transmitter logic. This register is reset to "00H" by a channel or hardware reset.

D7	D6	D5	D4	D3	D2	D1	D0
TX BITS CHAR 1	TX BITS CHAR 0	TX AUTO ENABLES	SEND BREAK	TX CRC ENABLE	DTR	RTS	TX ENABLE

D7, D6 Transmit Bits/Character 1 and 0

The state of these two bits determine the number of bits in each byte transferred from the transmit buffer to the transmit shift register. All data written to the transmit buffer must be right-justified with the least-significant bits first. The Five Or Less mode allows transmission of one to five bits per character ; however, the CPU should format the data characters as shown. If Parity is enabled, one additional bit per character will be transmitted.

TX BITS/ CHAR 1	TX BITS/ CHAR 0	Bits/character (no parity)
0	0	Five or Less
0	1	6
1	0	7
1	1	8

D7	D6	D5	D4	D3	D2	D1	D0	Five or Less
1	1	1	1	0	0	0	D	Sends One Data Bit
1	1	1	0	0	0	D	D	Sends Two Data Bits
1	1	0	0	0	D	D	D	Sends Three Data Bits
1	0	0	0	D	D	D	D	Sends Four Data Bits
0	0	0	D	D	D	D	D	Sends Five Data Bits

D5 : Transmit Auto Enables

When this bit is set to a one, and the Transmit Enable bit is also set, a Low on the CTS input pin will enable the transmitter. When this bit is zero, the CTS pin is simply an input to the SIO, and its status is displayed in Status Register 0.

D4 : Send Break

When set to a one, this bit immediately forces the Transmit Data output pin (TxD) to a spacing condition (continuous 0's), regardless of any data being transmitted at the time. This bit functions, whether the transmitter is enabled or not. When this bit is reset to zero, the transmitter will continue to send the contents of the transmit shift register. The shift register may contain sync characters, data characters, or all ones.

D3 : Transmitter CRC Enable

This bit determines if CRC calculations are performed on a transmitted data character. If this bit is a one at the time a character is loaded from the transmit buffer to the transmit shift register, CRC is calculated on the character. CRC is not calculated on any automatically inserted sync characters. CRC is not automatically appended to the end of a message unless this bit is set, and the Transmit Underrun/EOM status bit in Status Register 0 is reset when a Transmit Underrun condition occurs. If this bit is a zero when a character is loaded from the transmit buffer into the transmit shift register, no CRC calculations are performed on the character. This bit is ignored in Asynchronous modes.

D2 : Data Terminal Ready (DTR)

This is the control bit for the DTR output pin. When this bit is set to a one, the DTR pin goes Low : when this bit is reset to a zero, the DTR pin goes High.

D1 : Request To Send (RTS)

This is the control bit for the RTS output pin. In Synchronous modes, when this bit is set to a one, the RTS pin goes Low ; when this bit is reset to a zero, the RTS pin goes High. In Asynchronous modes, when this bit is set, the RTS pin goes Low ; when this bit is reset, the RTS pin will go High only after all the bits of the character are transmitted, and the transmit buffer is empty.

D0 : Transmitter Enable

Data is not transmitted until this bit is set to a one, until the Send Break bit is reset and, if Tx Auto Enables mode is selected, until the CTS pin is Low. To transmit sync or flag characters in Synchronous modes, this bit has to be set when the transmit buffer is empty. Data or sync characters in the process of being transmitted are completely sent if this bit is reset to zero after transmission has started. If this bit is reset during the transmission of a CRC character, sync or flag characters are sent instead of the CRC character.

STATUS REGISTER 0 (STAT 0) READ ONLY

This register contains the status of the receive and transmit buffers and the status bits for the five sources of External/Status interrupts.

D7	D6	D5	D4	D3	D2	D1	D0
BREAK/ ABORT	UNDERRUN /EOM	CTS	HUNT/ SYNC	DCD	TX BUFR EMPTY	INTERPT PENDING	RX CHAR AVAIL

D7 : Break/Abort

This bit is reset by a channel or hardware reset. In Asynchronous modes, this bit is set when a Break sequence (null character plus framing error) is detected in the received data stream. An External/Status interrupt, if enabled, is generated when Break is detected. The interrupt service routine must issue a Reset External/Status Interrupt command (Command 2) to the SIO, so the break detection logic can recognize the termination of the Break sequence.

The Break/Abort bit is reset to a zero when the termination of the Break sequence is detected in the incoming data stream. The termination of the Break sequence also causes the generation of an External/Status interrupt. Command 2 must be issued to enable the break detection logic to look for the next Break sequence. A single extraneous null character is present in the receiver after the termination of a break ; it should be read and discarded.

In SDLC mode, this bit is set by the detection of an Abort sequence (seven or more ones) in the received data stream. The External/Status Interrupt is handled the same way as in the case of a Break sequence. The Break/Abort bit is not used in the other Synchronous modes.

D6 : Transit Underrun/EOM

This bit is set to a one following a hardware or channel reset, when the transmitter is disabled or when a Send Abort command (Command 1) is issued. This bit can only be reset by the Reset Transmit Underrun/EOM Latch command in the Command Register. This bit is used to control the transmission of CRC at the end of a message in Synchronous modes. When a transmit underrun condition occurs and this bit is low. CRC will be appended to the end of the transmission, and this bit will be set. Only the 0-to-1 transition of this bit causes an External/Status interrupt, when enabled. This bit is not used in Asynchronous modes.

D5 : Clear To Send (CTS)

This bit indicates the inverted state of the CTS input pin at the time of the last change of any of the five External/Status bits. Any transition of the CTS input causes the CTS bit to be latched and generates an External/Status interrupt request, if enabled. To read the current state of the CTS pin, this bit must be read immediately following a Reset External/Status Interrupts command (command 2).

D4 : Hunt/Sync

In Asynchronous modes, this bit indicates the inverted state of the SYNC input pin at the time of the last change of any of the five External/Status bits. Any transition of the SYNC input causes the Hunt/Sync bit to be latched and generates an External/Status interrupt request, if enabled. To read the current state of the SYNC pin, this bit must be read immediately following a Reset External/Status Interrupt command (command 2).

In External sync mode, the SYNC pin is used by external logic to signal character synchronization is achieved, the SYNC pin is driven Low on the second rising edge of the Receive Clock (RxC) on which the last bit of the sync character was received. Once the SYNC pin is Low, it should be held Low until the end of the message and the driven back High. Both transitions on the SYNC pin cause External/Status interrupt requests, if enabled. The inverted state of the SYNC pin is indicated by this bit.

In Monosync, Bisync, and SDLC modes, this bit indicates when the receiver is in the Hunt mode. This bit is set to a one following a hardware ir channel reset, after the Enter Hunt Mode bit is written High, when the receiver is disabled, or when an Abort sequence (SDLC mode) is detected. This bit will remain in this state until character synchronization is achieved. External/Status interrupt requests will be generated on both transitions of the Hunt/Sync bit.

D3 : Data Carrier Detect (DCD)

This bit indicates the inverted state of the DCD input pin at the time of the last change of any of the five External/Status bits. Any transition of the DCD input causes the DCD bit to be latched and generates an External/Status interrupt request, if enabled. To read the current state of the DCD pin, this bit must be read immediately following a Reset External/Status Interrupts command (command 2).

D2 : Transmit Buffer Empty

This bit is set to a one, when the transmit buffer becomes empty, and when the last CRC bit is transmitted in Synchronous or SDLC modes. This bit is reset when the transmit buffer is loaded or while the CRC character is being sent in Synchronous or SDLC modes. This bit is set to a one following a hardware or channel reset.

D1 : Interrupt Pending

Any interrupt condition, pending in the interrupt control logic for this channel, will set this bit to a one. This bit is reset to zero by a hardware channel reset, or when all the interrupt conditions are cleared.

D0 : Receive Character Available

This bit is set to a one when a character becomes available in the receive data FIFO. This bit is reset to zero when the receive data FIFO (receive buffer) is read, or by a hardware or channel reset.

STATUS REGISTER 1 (STAT 1) READ ONLY

This register contains the Special Receive Condition status bits and the Residue codes for the I-field in the SDLC receive mode. The All Sent bit is set High, and all other bits are reset to a Low by a channel or hardware reset.

D7	D6	D5	D4	D3	D2	D1	D0
END OF FRAME	CRC/ FRAME ERROR	RX OVER-RUN ERR	PARITY ERROR	RESIDUE CODE 2	RESIDUE CODE 1	RESIDUE CODE 0	ALL SENT

D7 : End Of Frame (SDLC)

This bit is used only in SDLC mode. When set to a one, this bit indicates that a valid closing flag has been received and that the CRC/Framing Error bit and Residue codes are valid. If receiver interrupts are enabled, a Special Receive Condition interrupt will also be generated. This bit can be reset by issuing an Error Reset command (command 6). This bit is also updated by the first character of the following frame. This bit is a zero in all modes except for SDLC.

D6 : CRC/Framing Error

In Asynchronous modes, if a Framing Error occurs, this bit is set to a one for the receive character in which the framing error occurred. When this bit is set to a one, a Special Receive Condition interrupt will be requested, if receiver interrupts are enabled.

Detection of a Framing Error adds an additional one-half bit time to the character time, so that the Framing Error is not interpreted as a new start bit.

In Synchronous and SDLC modes, this bit indicates the result of comparing the received CRC value to the appropriate check value. A zero indicates that a match has occurred. This bit is usually set since most bit combinations result in a non-zero CRC, except for a correctly completed message. Receiver interrupts are not requested by the CRC Error bit.

The CRC/Framing bit is not latched in any receiver mode. It is always updated when the next character is received. An Error Reset command (command 6) will always reset this bit to zero.

D5 : Receive Overrun Error

This bit indicates that the receive data FIFO has overflowed. Only the character that has been written over is flagged with this error. When the character is read, the error condition is latched until reset by the Error Reset command (command 6). If receiver interrupts are enabled, the overrun character and all subsequent characters received, until the Error Reset command is issued, will generate a Special Receive Condition interrupt request.

D4 : Parity Error

When parity is enabled, this bit is set to a one for those characters whose parity does not match the programmed sense (even/odd). This bit is latched so that once an error occurs, it remains set until the Error Reset command (command 6) is issued. If parity is a Special Receive Condition, a Parity is a Special Receive Condition, a Parity Error will cause a Special Receive Condition interrupt request on the character containing the error and on all subsequent characters until the Error Reset command is issued.

D3, D2, D1 : Residue Codes 2, 1, and 0

In those cases of the SDLC receive mode, where the I-field is not an integral multiple of the character length, these three bits indicate the length of the residual I-field read in the previous bytes. These codes are meaningful only for the transfer in which the End Of Frame bit is set. This field is set to 000 by a channel or hardware reset and can leave this state only if SDLC mode is selected, and a character is received.

FOR EIGHT BITS PER CHARACTER

Residue Code 2	Residue Code 1	Residue Code 0	I-Field Bits In Previous Byte	I-Field Bits In Second Previous Byte
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

I-Fiel Bits are Right-justified in all Cases.

If a receive character length, different from eight bits, is used for the I-field, a table similar to the previous one may be constructed for each different character length. For no residue (that is, the last character boundary coincides with the boundary of the I-field and CRC field), the Residue codes are as follows :

Bits Per Character	Residue Code 2	Residue Code 1	Residue Code 0
8 Bits Per Character	0	1	1
7 Bits Per Character	0	0	0
6 Bits Per Character	0	1	0
5 Bits Per Character	0	0	1

D0 : All Sent

This bit is only active in Asynchronous modes ; it is always High in Synchronous or SDLC modes. This bit is Low while the transmitter is sending characters : it will go High only after all the bits of the character are transmitted, and the transmit buffer is empty.

DATA REGISTER (DATARG)

The Data Register is actually two separate registers ; a write only register that is the Transmit Buffer, and a read only register that is the Receiver Buffer. The Receiver Buffer is also the top register of a three register stack called the receive data FIFO. The Data Register is not affected by a channel or hardware reset.

D7	D6	D5	D4	D3	D2	D1	D0
DATA 7	DATA 6	DATA 5	DATA 4	DATA 3	DATA 2	DATA 1	DATA 0

TIME CONSTANT REGISTER (TCREG)

This register contains the time constant used by the down counter in the baud rate generator. The time constant may be changed at any time, but the new value does not take effect until the next time the time constant is loaded into the down counter. It is recommended that the BRG be disabled before writing to this register, as no attempt was made to synchronize the loading of a new time constant with the clock used to drive the BRG. This register is reset to "00H" by a channel or hardware reset.

D7	D6	D5	D4	D3	D2	D1	D0
TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0

BAUD RATE GENERATOR CONTROL REGISTER (BRGCTL)

This register contains the control bits used to program the baud rate generator and to select the BRG output mode. This register is reset to "00H" by a channel or hardware reset.

D7	D6	D5	D4	D3	D2	D1	D0
				RxC INT/EXT	TxC INT/EXT	DIVIDE BY 64/4	BRG ENABLE

D7, D6, D5, D4 : Not Used (read as zeros)

D3 : Receiver Clock, Internal/External

This bit determines the direction of the RxC pin. When this bit is set to a one, the RxC pin is the output of the baud rate generator. If this bit is a zero, the RxC pin is an input, and an external source must supply the receiver clock. The receiver clock is always the signal on the RxC pin, except in Loop Mode, when the transmitter clock is connected internally to the receiver clock.

D2 : Transmitter Clock, Internal/External

This bit determines the direction of the TxC pin. When this bit is set to a one, the TxC pin is the output of the baud rate generator. If this bit is a zero, the TxC pin is an input, and an external source must supply the transmitter clock. The transmit clock is always the signal on the TxC pin.

D1 : Divide By 64/4

This bit specifies the minimum BRG input clock cycles to output clock cycle. This minimum occurs when the Time Constant Register is loaded with a "01H" value. When this bit is set to a one, 64 input clocks are required for every output clock. When this bit is a zero, four input clocks are required for every output clock.

D0 : Baud Rate Generator Enable

This bit controls the operation of the baud rate generator. When this bit is set to a one, the BRG will start counting down from the value left in the down counter when this bit was last reset to zero. If the Time Constant Register is loaded while this bit is reset, the new time constant value is loaded immediately into the down counter. The baud rate generator is disabled from counting when this bit is reset.

INTERRUPT VECTOR REGISTER (VECTRG)

This register is used to hold a vector that is passed to the CPU during an interrupt acknowledge cycle. This register can also be accessed through a read/write cycle. If the Status Affects Vector bit in the Interrupt Control Register is disabled, the value programmed into the Vector Register will be passed to the CPU during an interrupt acknowledge cycle or a read cycle. If the Status Affects Vector bit in either channel is enabled, the lower three bits of this register are modified, according to the table listed in the Interrupt Control Register description. With Status Affects Vector on, and no interrupt pending in the SIO, the lower three bits will be read as 011. Only one Vector Register exists in the SIO, but it can be accessed through either channel. This register is reset to "0FH" by a hardware reset only.

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2 *	V1 *	V0 *

* Variable if Status Affects Vectors is Enabled.

MK68564 ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
T_A	Temperature Under Bias	– 25 to 100	°C
T_{stg}	Storage Temperature	– 65 to 150	°C
V_I	Voltage on Any Pin with Respect to Ground	– 3 to 7	V
P_D	Power Dissipation	1.5	W

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC ELECTRICAL CHARACTERISTICS
(V_{CC} = 5.0V ± 5%, T_A = 0 to 70°C)

Symbol	Parameter	Min.	Max.	Unit.
V_{IH}	Input High Voltage ; all Inputs	V_{SS} + 2.0	V_{CC}	V
V_{IL}	Input Low Voltage ; all Inputs	V_{SS} – 0.3	V_{SS} + 0.8	V
I_{LL}	Power Supply Current ; Outputs Open		190	mA
I_{IN}	Input Leakage Current (V_{IN} = 0 to 5.25)		± 10	µA
I_{TSI}	Three-state Input Current $\overline{\text{DTACK}}$, D0-D7, $\overline{\text{SYNC}}$, $\overline{\text{TxC}}$, $\overline{\text{RxC}}$ 0 < V_{IN} < V_{CC}, $\overline{\text{INTR}}$		20 ± 10	µA µA
V_{OH}	Output High Voltage (I_{LOAD} = – 400 µA, V_{CC} = MIN) $\overline{\text{DTACK}}$, D0-D7 (I_{LOAD} = – 150 µA, V_{CC} = MIN) All Other Outputs (except XTAL2 & $\overline{\text{INTR}}$)*	V_{SS} + 2.4		V
V_{OL}	Output Low Voltage (I_{LOAD} = 5.3mA, V_{CC} = MIN) $\overline{\text{INTR}}$, $\overline{\text{DTACK}}$, D0-D7 (I_{LOAD} = 2.4mA, V_{CC} = MIN) All Other Outputs (except XTAL2)*		0.05	V

* XTAL2 Special

INTR (Open drain)

CAPACITANCE
T_A = 25°C, F = 1MHz Unmeasured Pins Returned to Ground.

Symbol	Parameter	Test Conditions	Max.	Unit.
C_{IN}	Input Capacitance $\overline{\text{CS}}$, $\overline{\text{IACK}}$ All Others	Unmeasured Pins Returned to Ground	15 10	pF pF
C_{OUT}	Tri-state Output Capacitance		10	pF

AC ELECTRICAL CHARACTERISTICS
(V_{CC} = 5.0 VDC ± 5%, GND = 0 VDC, T_A = 0 to 70°C)

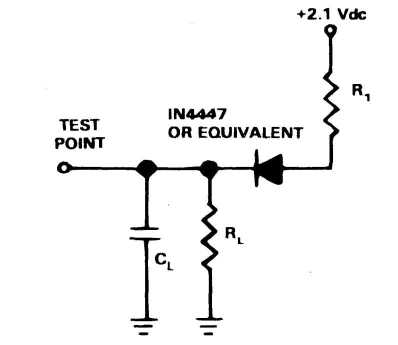
Number	Parameter	4.0 MHz		5.0 MHz		Unit	Notes
		Min.	Max.	Min.	Max.		
1	CLK Period	250	1000	200	1000	ns	
2	CLK Width High	105		80		ns	
3	CLK Width Low	105		80		ns	
4	CLK Fall Time		30		30	ns	
5	CLK Rise Time		30		30	ns	
6	$\overline{\text{CS}}$ Low to CLK High (setup time)	0		0		ns	1
7	A1-A5 Valid to $\overline{\text{CS}}$ Low (setup time)	0		0		ns	
8	DATA Valid to $\overline{\text{CS}}$ Low (write cycle)	0		0		ns	
9	$\overline{\text{CS}}$ Width High	50		50		ns	1
10	$\overline{\text{DTACK}}$ Low to A1-A5 Invalid (hold time)	0		0		ns	
11	$\overline{\text{DTACK}}$ Low to DATA Invalid (write cycle hold time)	0		0		ns	
12	$\overline{\text{CS}}$ High to $\overline{\text{DTACK}}$ High (delay)		55		50	ns	
13	CLK High to $\overline{\text{DTACK}}$ Low		320		295	ns	
14	R/$\overline{\text{W}}$ Valid to $\overline{\text{CS}}$ Low (setup time)	0		0		ns	
15	$\overline{\text{DTACK}}$ Low to R/$\overline{\text{W}}$ Invalid (hold time)	0		0		ns	
16	CLK Low to DATA Out		450		450	ns	
17	$\overline{\text{CS}}$ High to DATA Out Invalid (hold time)	0		0		ns	11
18	$\overline{\text{CS}}$ High to $\overline{\text{DTACK}}$ High Impedance		105		100	ns	
19	$\overline{\text{DTACK}}$ Low to $\overline{\text{CS}}$ High	0		0		ns	
20	DATA Valid to $\overline{\text{DTACK}}$ Low	70		70		ns	
21	$\overline{\text{IACK}}$ Width High	50		50		ns	1
22	$\overline{\text{IACK}}$ Low to CLK High (setup time)	0		0		ns	1
23	CLK Low to $\overline{\text{INTR}}$ Disabled		410		410	ns	2
24	CLK Low to DATA Out		330		330	ns	2
25	$\overline{\text{DTACK}}$ Low to $\overline{\text{IACK}}$, $\overline{\text{IEI}}$, High	0		0		ns	
26	$\overline{\text{IACK}}$ High to $\overline{\text{DTACK}}$ High	55		50		ns	
27	$\overline{\text{IACK}}$ High to $\overline{\text{DTACK}}$ High Impedence		105		100	ns	
28	$\overline{\text{IACK}}$ High to DATA Out Invalid (hold time)	0		0		ns	
29	DATA Valid to $\overline{\text{DTACK}}$ Low	195		195		ns	2
30	CLK Low to $\overline{\text{IEO}}$ Low		220		220	ns	3
31	$\overline{\text{IEI}}$ Low to $\overline{\text{IEO}}$ Low		140		140	ns	3
32	$\overline{\text{IEI}}$ High to $\overline{\text{IEO}}$ High		190		190	ns	4
33	$\overline{\text{IACK}}$ High to $\overline{\text{IEO}}$ High		190		190	ns	4
34	$\overline{\text{IACK}}$ High to $\overline{\text{INTR}}$ Low		200		200	ns	5
35	$\overline{\text{IEI}}$ Low to CLK Low (setup time)	10		10		ns	
36	$\overline{\text{IEI}}$ Low to $\overline{\text{INTR}}$ Disabled		425		425	ns	6
37	$\overline{\text{IEI}}$ Low to DATA Out Valid		225		225	ns	6
38	DATA Out Valid to $\overline{\text{DTACK}}$ Low	55		55		ns	6
39	$\overline{\text{IACK}}$ High to DATA Out High Impedence		120		90	ns	

AC ELECTRICAL CHARACTERISTICS (continued)
(V_{CC} = 5.0 VDC ± 5%, GND = 0 VDC, T_A = 0 to 70°C)

Number	Parameter	4.0 MHz		5.0 MHz		Unit	Notes
		Min.	Max.	Min.	Max.		
40	$\overline{\text{CS}}$ HIGH TO DATA Out High Impedence		120		90	ns	
41	$\overline{\text{CS}}$ or $\overline{\text{IACK}}$ High to CLK Low	100		100		ns	7
42	$\overline{\text{TxRDY}}$ or $\overline{\text{RxRDY}}$ Width Low		3		3	CLK's	8, 10
43	CLK High $\overline{\text{TxRDY}}$ or $\overline{\text{RxRDY}}$ Low		300		300	ns	
44	CLK High to $\overline{\text{TxRDY}}$ or $\overline{\text{RxRDY}}$ High		300		300	ns	
	$\overline{\text{IACK}}$ High to $\overline{\text{CS}}$ Low or $\overline{\text{CS}}$ High to IACK Low (not shown)	50		50		ns	1
45	$\overline{\text{CTS}}$, $\overline{\text{DCD}}$, $\overline{\text{SYNC}}$ Pulse Width High	200		200		ns	
46	$\overline{\text{CTS}}$, $\overline{\text{DCD}}$, $\overline{\text{SYNC}}$ Pulse Width Low	200		200		ns	
47	$\overline{\text{TxC}}$ Period	1000	DC	800	DC	ns	9
48	$\overline{\text{TxC}}$ Width Low	180	DC	180	DC	ns	
49	$\overline{\text{TxC}}$ Width High	180	DC	180	DC	ns	
50	$\overline{\text{TxC}}$ Low to TxD Delay (X1 Mode)		300		300	ns	
51	$\overline{\text{TxC}}$ Low to $\overline{\text{INTR}}$ Low Delay	5	9	5	9	CLK's	10
52	$\overline{\text{RxC}}$ Period	1000	DC	800	DC	ns	9
53	$\overline{\text{RxC}}$ Width Low	180	DC	180	DC	ns	
54	$\overline{\text{RxC}}$ Width High	180	DC	180	DC	ns	
55	RxD to $\overline{\text{RxC}}$ High Setup Time (X1 mode)	0		0		ns	
56	RxC High to RxD Hold Time (X1 mode)	140		140		ns	
57	$\overline{\text{RxC}}$ High to $\overline{\text{INTR}}$ Low Delay	10	13	10	13	CLK's	10
58	$\overline{\text{RxC}}$ High to $\overline{\text{SYNC}}$ Low Delay (output modes)	4	7	4	7	CLK's	10
59	$\overline{\text{RESET}}$ Low	1		1		CLK	10
60	XTAL 1 Width High (TTL in)	100		80		ns	
61	XTAL 1 Width Low (TTL in)	100		80		ns	
62	XTAL 1 Period (TTL in)	250	2000	200	2000	ns	
63	XTAL 1 Period (crystal in)	250	1000	200	1000	ns	

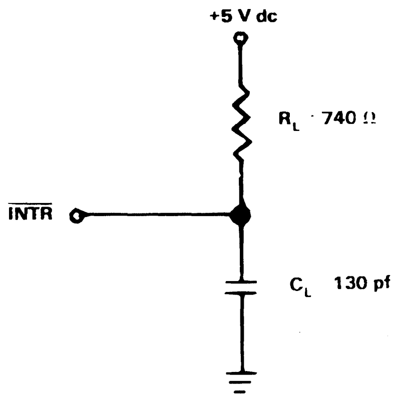
Notes :
1. This specification only applies if the SIO has completed all operations initiated by the previous bus cycle, when $\overline{\text{CS}}$ or IACK was asserted. Following a read, write, or interrupt acknoledge cycle, all operations are complete within two CLK cycles after the rising edge of CS or IACK. If CS or IACK is asserted prior to the completion of the internal operations, the new bus cycle will be postponed.
2. If $\overline{\text{IEI}}$ meets the setup time to the falling edge of CLK, 1 1/2 cycles following the clocking in of $\overline{\text{IACK}}$.
3. No internal interrupt request pending at the start of an interrupt acknoledge cycle.
4. Time starts when first signal goes invalid (high).
5. If an internal interrupt is pending at the end of the interrupt acknoledge cycle.
6. If Note 2 timing is not met.
7. If this spec is met, the delay listed in Note 1 will be one CLK cycle instead of two.
8. Ready signals will be negated asynchronous to the CLK, if the condition causing the assertion of the signals is cleared.
9. If RxC and $\overline{\text{TxC}}$ are asynchronous to the System Clock, the maximum clock rate into RxC and $\overline{\text{TxC}}$ should be no more than one-fifth the System Clock rate. If RxC and $\overline{\text{TxC}}$ are synchronized to the falling edge of the System Clock, the maximum clock rate into RxC and $\overline{\text{TxC}}$ can be one-fourth the System Clock rate.
10. System Clock.
11. Due to the dynamic nature of the internal data bus, if $\overline{\text{CS}}$ is held low for more than a few hundred milliseconds the read data may go to OOH before the end of the cycle.

Figure 13 : Output Test Load.



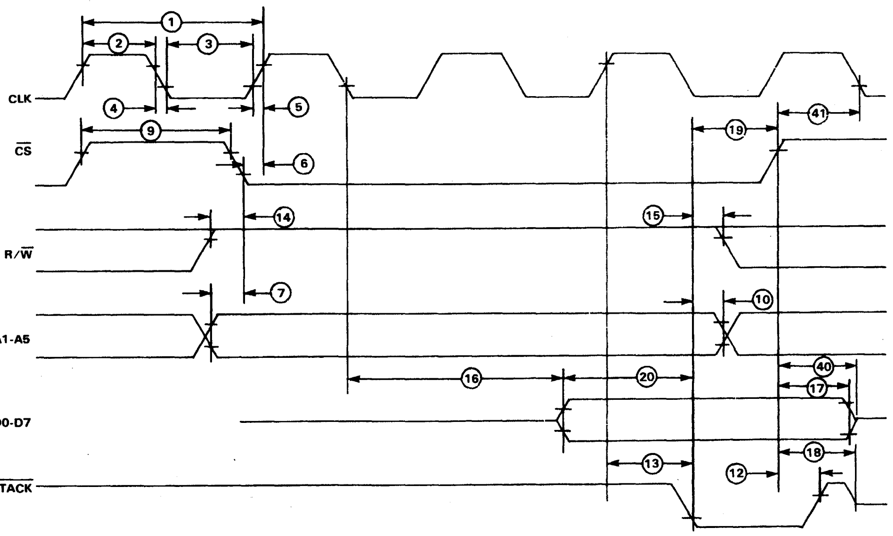
For all Outputs Except	DTACK, D0-D7
	INTR, XTAL2
	C_L = 130pf
	R_L = 16KΩ
	R_1 = 450Ω
for	DTACK, D0-D7
	C_L = 130pf
	R_L = 6KΩ
	R_1 = 200Ω

Figure 14 : INTR Test Load.



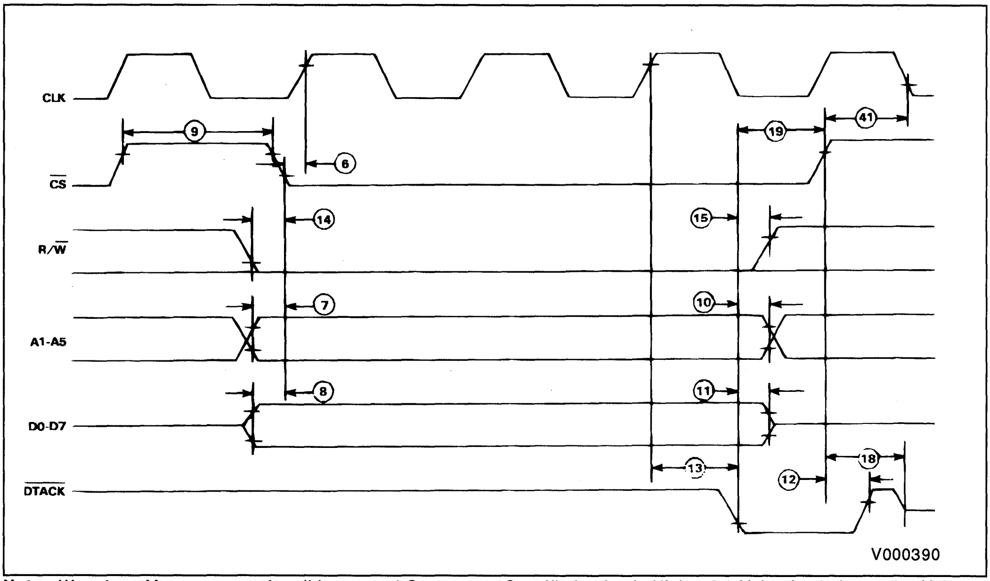
Note : XTAL2 Output Test Load is a Crystal.

Figure 15 : Read Cycle.



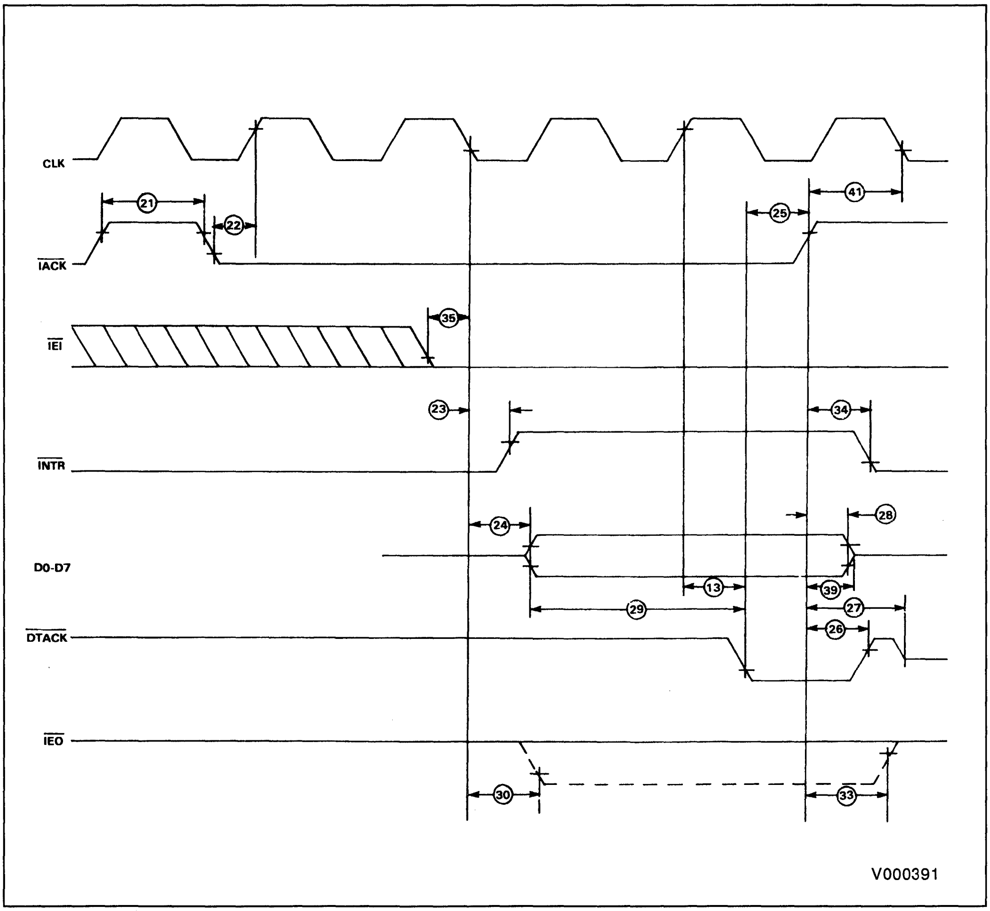
Note : Waveform Measurement for all Inputs and Outputs are Specified at Logic High = 2.0 Volts, Logic Low = 0.8 Volts.

Figure 16 : Write Cycle.



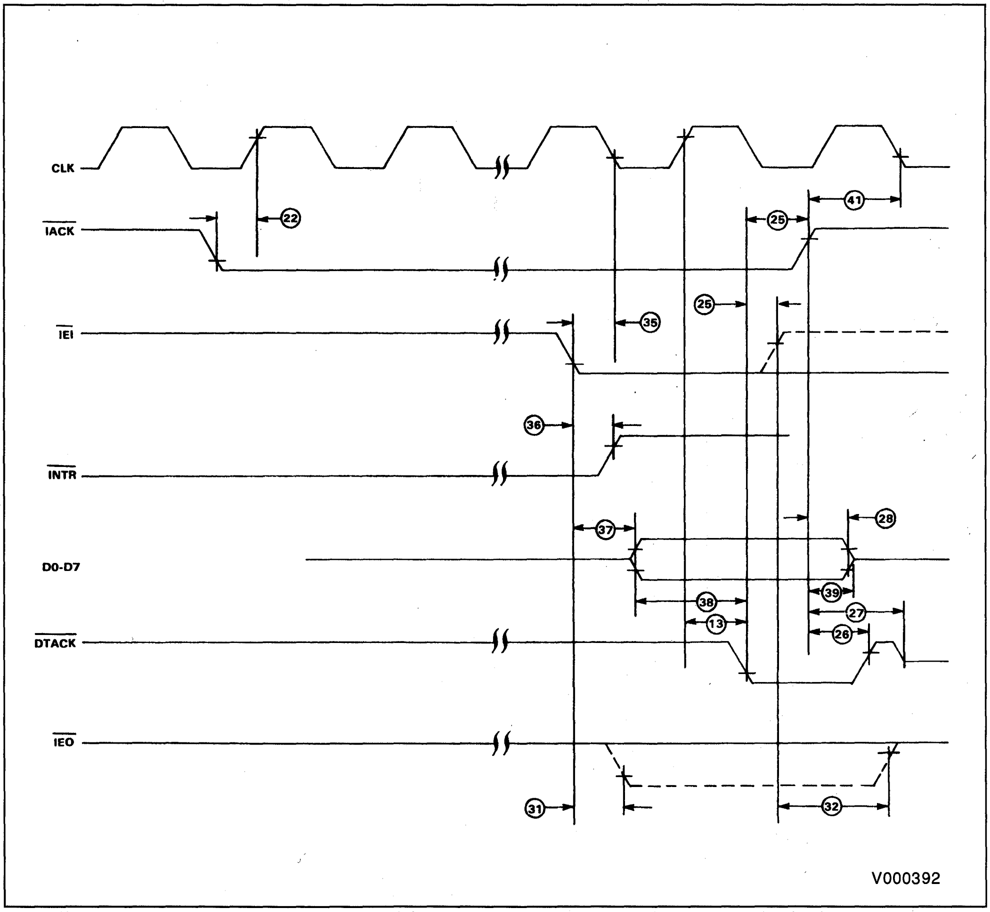
Note : Waveform Measurements for all Inputs and Outputs are Specified at Logic High = 2.0 Volts, Logic Low = 0.8 Volts.

Figure 17 : Interrupt Acknoledge Cycle ($\overline{\text{IEI}}$ low).



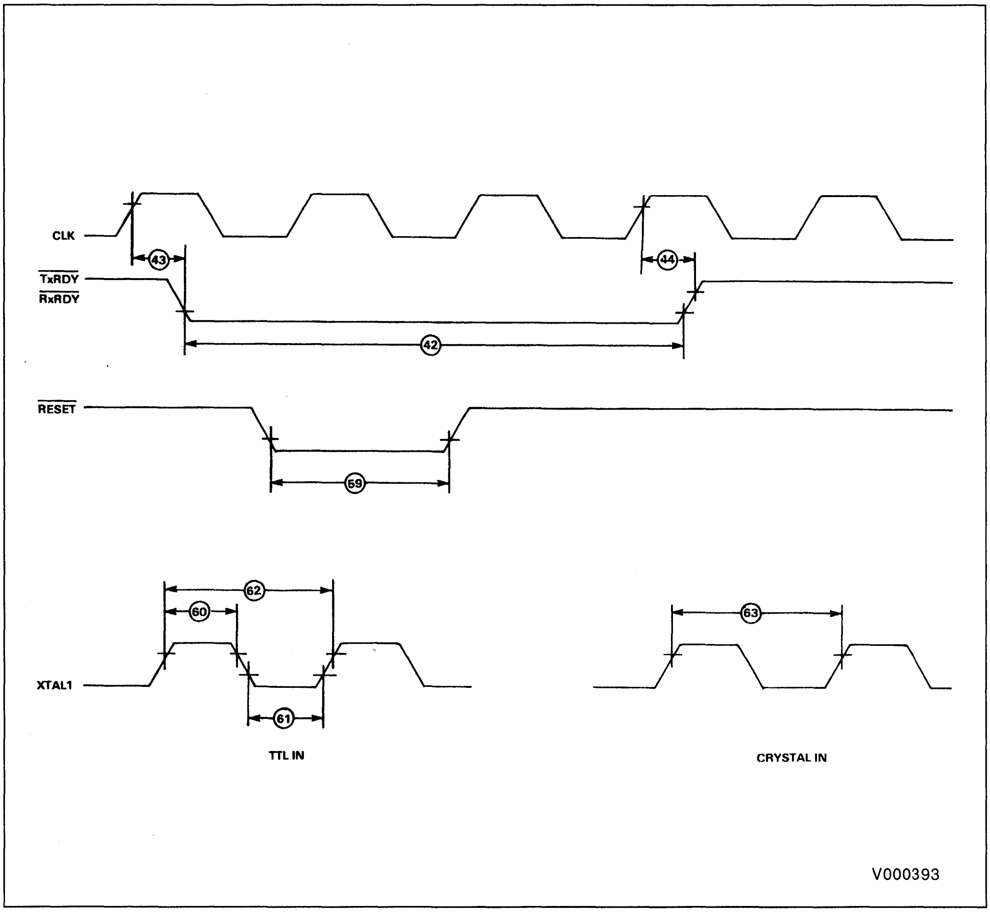
Note : Waveform Measurements for all Inputs and Outputs are Specified at Logic High = 2.0 Volts, Logic Low = 0.8 Volts.

Figure 18 : Interrupt Acknoledge Cycle ($\overline{\text{IEI}}$ high).



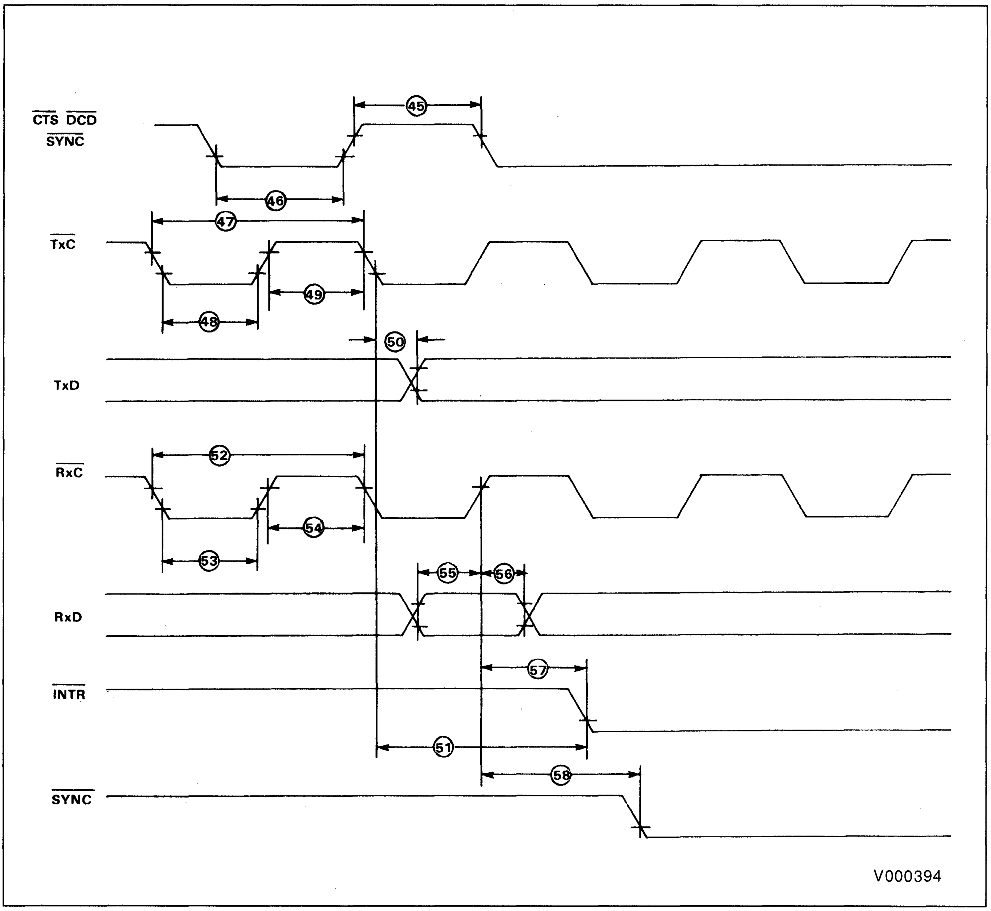
Note : Waveform Measurements for all Inputs and Outputs are Specified at Logic High = 2.0 Volts, Logic Low = 0.8 Volts.

Figure 19 : DMA Interface Timing.



Note : Waveform Measurements for all Inputs and Outputs are Specified at Logic High = 2.0 Volts, Logic Low = 0.8 Volts.

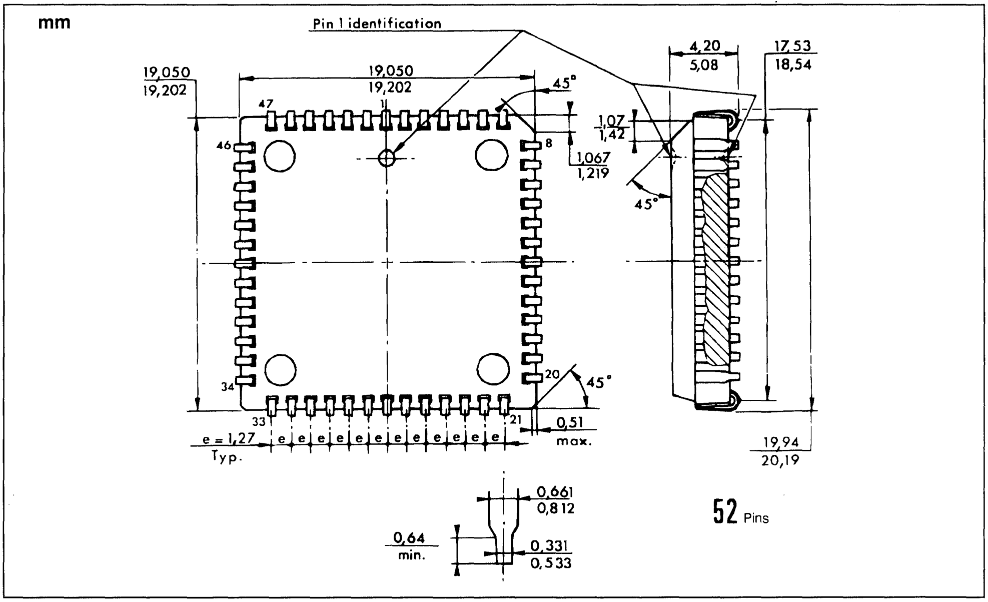
Figure 20 : Serial Interface Timing.



Note : Waveform Measurements for all Inputs and Outputs are Specified at Logic High = 2.0 Volts, Logic Low = 0.8 Volts.

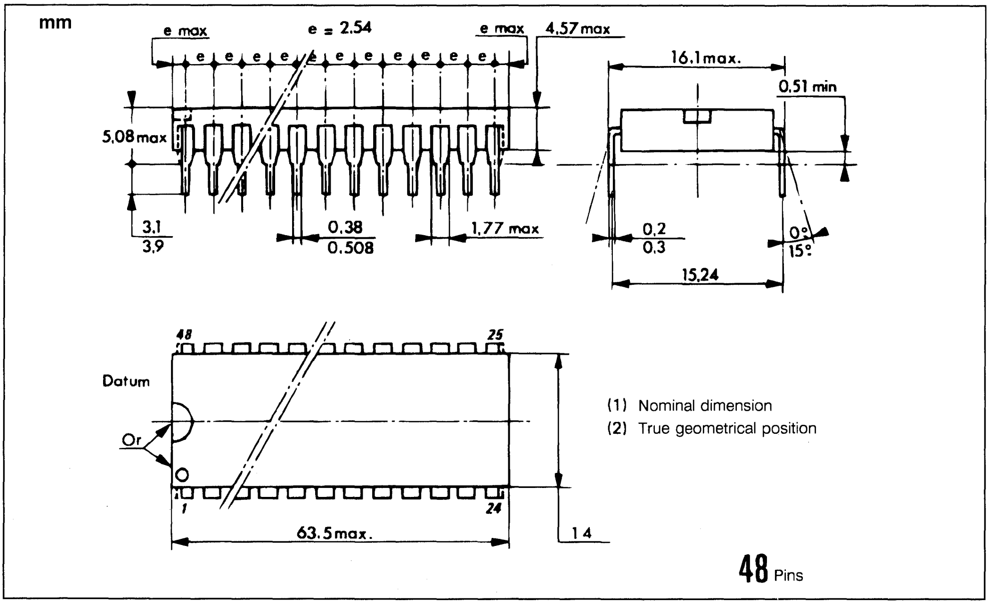
MK68564 52-PIN

Plastic Leader Chip Carrier (Q)



MK68564 48-PIN

Plastic Dual-IN-Line Package



MK68564 ORDERING INFORMATION

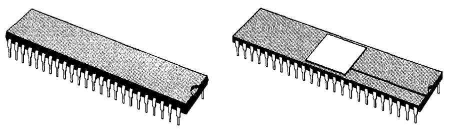
Part No.	Package Type	Max. Clock Frequency	Temperature Range
MK68564N-04	Plastic	4.0 MHz	0° to 70 °C
MK68564N-05	Plastic	5.0 MHz	0° to 70 °C
MK68564Q-04	PLCC	4.0 MHz	0° to 70 °C
MK68564Q-05	PLCC	5.0 MHz	0° to 70 °C

SGS-THOMSON
MICROELECTRONICS

MK68901

MULTI–FUNCTION PERIPHERAL

- 8 INPUT/OUTPUT PINS
  - Individually programmable direction
  - Individual interrupt source capability
    - Programmable edge selection
- 16 SOURCE INTERRUPT CONTROLLER
  - 8 Internal sources
  - 8 External sources
  - Individual source enable
  - Individual source masking
  - Programmable interrupt service modes
    - Polling
    - Vector generation
      - Optional In-service status
  - Daisy chaining capability
- FOUR TIMERS WITH INDIVIDUALLY PROGRAMMABLE PRESCALING
  - Two multimode timers
    - Delay mode
    - Pulse width measurement mode
    - Event counter mode
  - Two delay mode timers
  - Independent clock input
  - Time out output option
- SINGLE CHANNEL USART
  - Full Duplex
  - Asynchronous to 65 kbps
  - Byte synchronous to 1 Mbps
  - Internal/External baud rate generation
  - DMA handshake signals
  - Modem control
  - Loop back mode
- 68000 BUS COMPATIBLE
- 48 PIN DIP OR 52 PIN PLCC



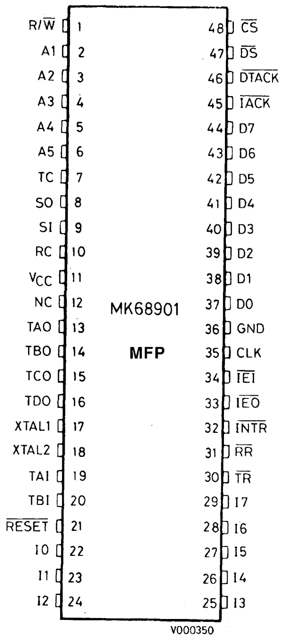
DESCRIPTION

The MK68901 MFP (Multi-Function Peripheral) is a combination of many of the necessary peripheral functions in a microprocessor system.
Included are :

Eight parallel I/O lines
Interrrupt controller for 16 sources
Four timers
Single channel full duplex USART

The use of the MFP in a system can significantly reduce chip count, thereby reducing system cost. The MFP is completely 68000 bus compatible, and 24 directly addressable internal registers provide the

Figure 1 : Pin connections.



necessary control and status interface to the programmer.

The MFP is a derivative of the MK3801 STI, a Z80 family peripheral.

PIN DESCRIPTION

GND :	Ground
V_{CC} :	+5 volts (± 5%)
$\overline{\text{CS}}$:	Chip Select (input, active, low). $\overline{\text{CS}}$ is used to select the MK68901 MFP for accesses to the internal registers. $\overline{\text{CS}}$ and $\overline{\text{IACK}}$ must not be asserted at the same time.
$\overline{\text{DS}}$:	Data Strobe (input, active low). $\overline{\text{DS}}$ is used as part of the chip select and interrupt acknowledge functions.
R/$\overline{\text{W}}$:	Read/Write (input). R/$\overline{\text{W}}$ is the signal from the bus master indicating whether the current bus cycle is a Read (High) or Write (Low) cycle.
$\overline{\text{DTACK}}$:	Data Transfer Acknowledge. (output, active low, tri-stateable) $\overline{\text{DTACK}}$ is used to signal the bus master that data is ready, or that data has been accepted by the MK68901 MFP.
A1-A5 :	Address Bus (inputs). The adress bus is used to adress one of the internal registers during a read or write cycle.
D0-D7 :	Data Bus (bi-directional, tri-stateable). The data bus is used to receive data from or transmit data to one of the internal registers during a read or write cycle. It is also used to pass a vector during an interrupt acknowledge cycle.
CLK :	Clock (input). This input is used to provide the internal timing for the MK68901 MFP.
$\overline{\text{RESET}}$:	Device reset. (input, active low). Reset disables the USART receiver and transmitter, stops all timers and forces the timer outputs low, disables all interrupt channels and clears any pending interrupts. The General Purpose Interrupt/I/O lines will be placed in the tri-state input mode. All internal registers (except the timer, USART data registers, and transmit status register) will be cleared.
$\overline{\text{INTR}}$:	Interrupt Request (output, active low, open drain). $\overline{\text{INTR}}$ is asserted when the MK68901 MFP is requesting an interrupt. $\overline{\text{INTR}}$ is negated during an interrupt acknowledge cycle or by clearing the pending interrupt(s) through software.
$\overline{\text{IACK}}$:	Interrupt Acknowledge (input, active low). $\overline{\text{IACK}}$ is used to signal the MK68901 MFP that the CPU is acknowledging an interrupt. $\overline{\text{CS}}$ and $\overline{\text{IACk}}$ must not be asserted at the same time.
$\overline{\text{IEI}}$:	Interrupt Enable In (input, active low). $\overline{\text{IEI}}$ is used to signal the MK68901 MFP that no higher priority device is requesting interrupt service.
$\overline{\text{IEO}}$:	Interrupt Enable Out (output, active low). $\overline{\text{IEO}}$ is used to signal lower priority peripherals that neither the MK68901 MFP nor another higher priority peripheral is requesting interrupt service.
10-17 :	General Purpose Interrupt I/O lines. These lines may be used as interrupt inputs and/or I/O lines. When used as interrupt inputs, their active edge is programmable. A data direction register is used to define which lines are to be Hi-Z inputs and which lines are to be push-pull TTL compatible outputs.
SO :	Serial Output. This is the output of the USART transmitter.
SI :	Serial Input. This is the input to the USART receiver.
RC :	Receiver Clock. This input controls the serial bit rate of the USART receiver.
TC :	Transmitter Clock. This input controls the serial'bit rate of the USART transmitter.
$\overline{\text{RR}}$:	Receiver Ready. (output, active low) DMA output for receiver, which reflects the status of Buffer Full in port number 15.
$\overline{\text{TR}}$:	Transmitter Ready. (output, active low) DMA output for transmitter, which reflects the status of Buffer Empty in port number 16.
TAO,TBO, TCO,TDO:	Timer Outputs. Each of the four timers has an output which can produce a square wave. The output will change states each timer cycle ; thus one full period of the timer out signal is equal to two timer cycles. TAO or TBO can be reset (logic "O") by a write to TACR, or TBCR respectively.
XTAL1, XTAL2 :	Timer Clock inputs. A crystal can be connected between XTAL1 and XTAL2, or XTAL1 can be driven with a TTL level clock. When driving XTAL1 with a TTL level clock, XTAL2 must be allowed to float.

When using a crystal, external capacitors are required. See figure 33. All chip accesses are independent of the timer clock.

TAI,TBI : Timer A, B inputs. Used when running the timers in the event count or the pulse width measurement mode. The interrupt channels associated with 14 and 13 are used for TAI and TBI, respectively. Thus, when running a timer in the pulse width measurement mode, 14 or 13 can be used for I/O only.

Figure 3 : MK68901 Block Diagram.

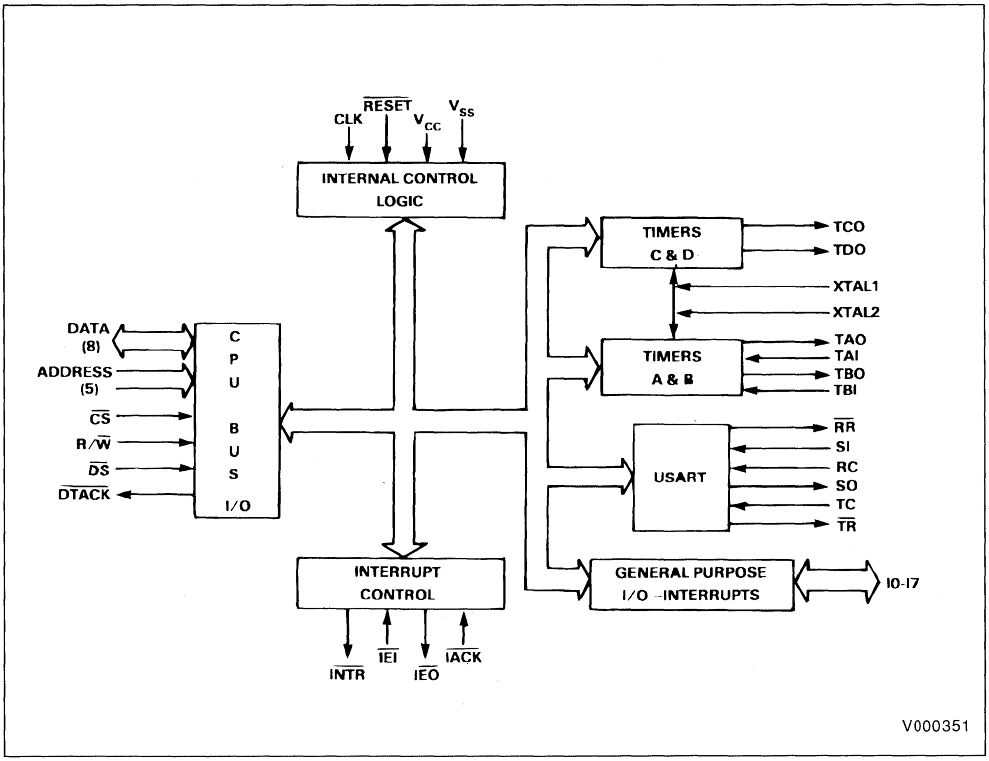


Figure 4 : Register Map.

Address Port N°.	Abbreviation	Register Name
0	GPIP	GENERAL PURPOSE I/O
1	AER	ACTIVE EDGE REGISTER
2	DDR	DATA DIRECTION REGISTER
3	IERA	INTERRUPT ENABLE REGISTER A
4	IERB	INTERRUPT ENABLE REGISTER B
5	IPRA	INTERRUPT PENDING REGISTER A
6	IPRB	INTERRUPT PENDING REGISTER B
7	ISRA	INTERRUPT IN-SERVICE REGISTER A
8	ISRB	INTERRUPT IN-SERVICE REGISTER B
9	IMRA	INTERRUPT MASK REGISTER A
A	IMRB	INTERRUPT MASK REGISTER B
B	VR	VECTOR REGISTER
C	TACR	TIMER A CONTROL REGISTER
D	TBCR	TIMER B CONTROL REGISTER
E	TCDCR	TIMERS C AND D CONTROL REGISTER
F	TADR	TIMER A DATA REGISTER
10	TBDR	TIMER B DATA REGISTER
11	TCDR	TIMER C DATA REGISTER
12	TDDR	TIMER D DATA REGISTER
13	SCR	SYNC CHARACTER REGISTER
14	UCR	USART CONTROL REGISTER
15	RSR	RECEIVER STATUS REGISTER
16	TSR	TRANSMITTER STATUS REGISTER
17	UDR	USART DATA REGISTER

INTERRUPTS

The General Purpose I/O-Interrupt Port (GPIP) provides eight I/O lines that may be operated either as inputs or outputs under software control. In addition, each line may generate an interrupt in either a positive going edge or a negative going edge of the input signal.

The GPIP has three associated registers. One allows the programmer to specify the Active Edge for each bit that will trigger an interrupt. Another register specifies the Data Direction (input or output) associated with each bit. The third register is the actual data I/O register used to input or output data to the port. These three registers are illstrated in figure 5.

The Active Edge Register (AER) allows each of the General Purpose Interrupts to provide an interrupt on either a 1-0 transition or a 0-1 transition. Writing a zero to the appropriate bit of the AER causes the associated input to produce an interrupt on the 1-0 transition. The edge bit is simply one input to an exclusive-or gate, with the other input coming from the input buffer ant the output going to a 1-0 transition detector. Thus, depending upon the state of the input, writing the AER can cause an interrupt-producing transition, which will cause an interrupt on the associated channel, if that channel is enabled. One would then normally configure the AER before enabling interrupts via IERA and IERB.

Note : Changing the edge bit, with the interrupt enabled, may cause an interrupt on that channel.

The Data Direction Register (DDR) is used to define 10-17 as inputs or as outputs on a bit by bit basis. Writing a zero into a bit of the DDR causes the corresponding Interrupt-I/O pin to be a Hi-Z input. Writing a one into a bit of the DDR causes the corresponding pin to be configured as a push-pull output. When data is written into the GPIP, those pins defined as inputs will remain in the Hi-Z state while those pins defined as outputs will assume the state (high or low) of their corresponding bit in the GPIP. When the GPIP is read, the data read will come directly from the corresponding bit of the GPIP register for all pins defined as output, while the data read on all pins defined as inputs will come from the input buffers.

Each individual function in the MK68901 is provided with a unique interrupt vector that is presented to the system during the interrupt acknowledge cycle. The interrupt vector returned during the interrupt acknowledge cycle is shown in figure 6, while the vector register is shown in figure 7.

There are 16 vector addresses generated internally by the MK68901, one for each of the 16 interrupt channels.

The Interrupt Control Registers (figure 8) provide control of interrupt processing for all I/O facilities of the MK68901. These registers allow the programmer to enable or disable any or all of the 16 interrupts, providing masking for any interrupt, and provide access to the pending and in-service status of the interrupt. Optional end-of-interrupt modes are available under software control. All the interrupts are prioritized as shown in figure 9.

Figure 5 : General Purpose I/O Registers.

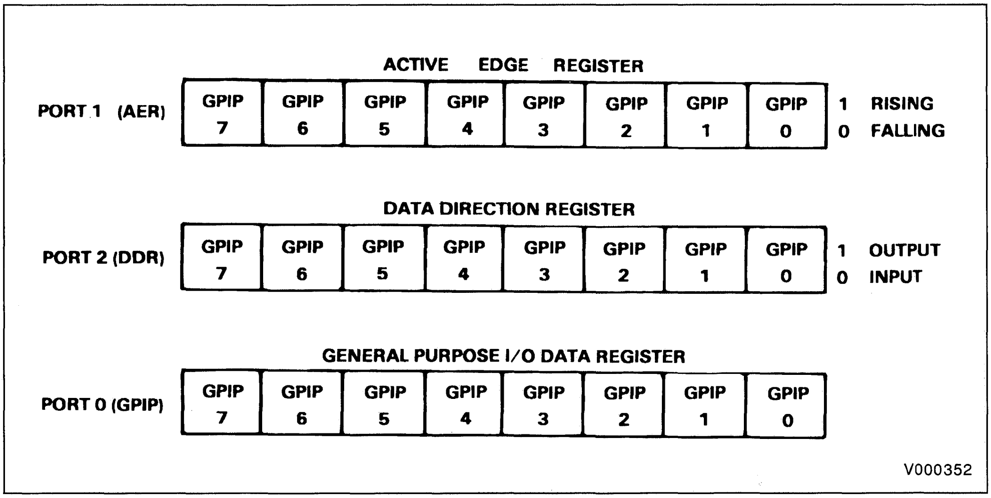


Figure 6 : Interrupt Vector.

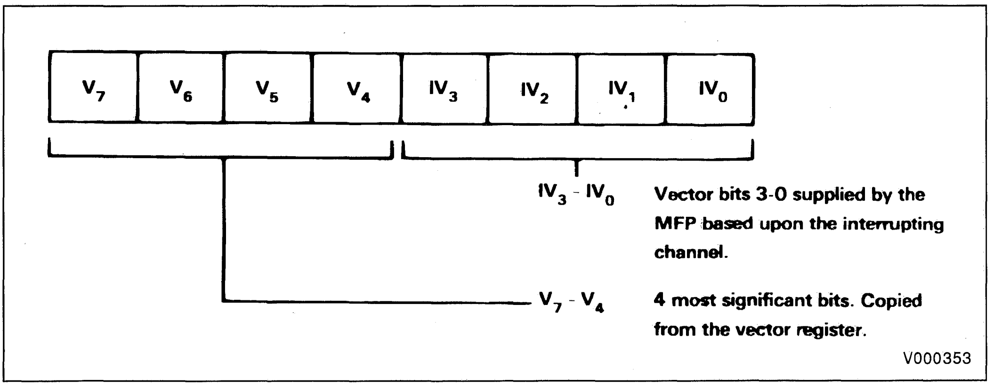

V_7
V_6
V_5
V_4
IV_3
IV_2
IV_1
IV_0
IV_3 - IV_0 Vector bits 3-0 supplied by the MFP based upon the interrupting channel.
V_7 - V_4 4 most significant bits. Copied from the vector register.
V000353

Figure 7 : Vector Register.

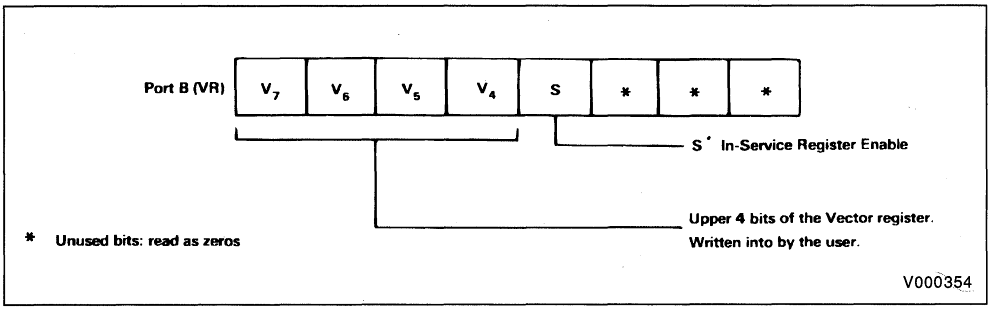

Port B (VR)
V_7
V_6
V_5
V_4
S
*
*
*
S' In-Service Register Enable
Upper 4 bits of the Vector register. Written into by the user.
* Unused bits: read as zeros
V000354

Figure 8 : Interrupt Control Registers.

INTERRUPT ENABLE REGISTERS

ADDRESS		7	6	5	4	3	2	1	0
PORT 3	A (IERA)	GPIP 7	GPIP 6	TIMER A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	TIMER B
PORT 4	B (IERB)	GPIP 5	GPIP 4	TIMER C	TIMER D	GPIP 3	GPIP 2	GPIP 1	GPIP 0

INTERRUPT PENDING REGISTERS

		7	6	5	4	3	2	1	0
PORT 5	A (IPRA)	GPIP 7	GPIP 6	TIMER A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	TIMER B
PORT 6	B (IPRB)	GPIP 5	GPIP 4	TIMER C	TIMER D	GPIP 3	GPIP 2	GPIP 1	GPIP 0

WRITING 0 CLEAR
WRITING 1 UNCHANGED

INTERRUPT IN-SERVICE REGISTERS

		7	6	5	4	3	2	1	0
PORT 7	A (ISRA)	GPIP 7	GPIP 6	TIMER A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	TIMER B
PORT 8	B (ISRB)	GPIP 5	GPIP 4	TIMER C	TIMER D	GPIP 3	GPIP 2	GPIP 1	GPIP 0

INTERRUPT MASK REGISTERS

		7	6	5	4	3	2	1	0
PORT 9	A (IMRA)	GPIP 7	GPIP 6	TIMER A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	TIMER B
PORT A	B (IMRB)	GPIP 5	GPIP 4	TIMER C	TIMER D	GPIP 3	GPIP 2	GPIP 1	GPIP 0

1 UNMASKED 0 MASKED

V000355

Figure 9 : Interrupt Control Register Definitions

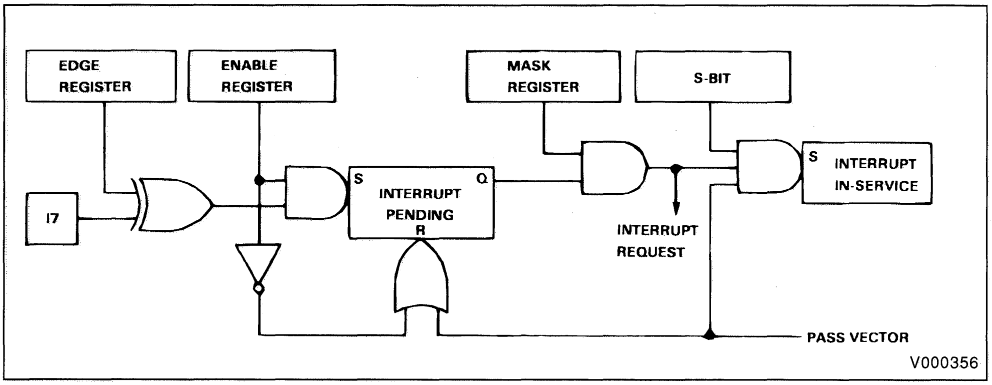
Priority	Channel	Description
HIGHEST	1111	General Purpose Interrupt 7(I7)
	1110	General Purpose Interrupt 6(I6)
	1101	Timer A
	1100	Receive Buffer Full
	1011	Receive Error
	1010	Transmit Buffer Empty
	1001	Transmit Error
	1000	Timer B
	0111	General Purpose Interrupt 5(I5)
	0110	General Purpose Interrupt 4(I4)
	0101	Timer C
	0100	Timer D
	0011	General Purpose Interrupt 3(I3)
	0010	General Purpose Interrupt 2(I2)
	0001	General Purpose Interrupt 1(I1)
LOWEST	0000	General Purpose Interrupt 0(I0)

Interrupts may be either polled or vectored. Each channel may be individual enabled or disabled by writing a one or a zero in the appropriate bit of Interrupt Enable Registers (IERA, IERB - see figure 8 for all registers in this section). When disabled, an interrupt channel is completely inactive. Any internal or external action which would normally produce an interrupt on that channel is ignored and any pending interrupt on that channel will be cleared by disabling that channel. Disabling an interrupt channel has no effect on the corresponding bit in Interrupt In-Service Registers (ISRA, ISRB) ; thus, if the In-service Registers are used and an interrupt is in service on that channel when the channel is disabled, it will remain in service until cleared in the normal manner. IERA and IERB are also readable.

When an interrupt is received on an enabled channel, its corresponding bit in the pending register will be set. When that channel is acknowledged it will pass its vector, and the corresponding bit in the Interrupt Pending Register (IPRA or IRPB) will be cleared. IPRA and IPRB are readable ; thus by polling IPRA and IPRB, it can be determined whether a channel has a pending interrupt. IPRA and IPRB are also writeable and a pending interrupt can be cleared without going through the acknowledge sequence by writing a zero to the appropriate bit. This allows any one bit to be cleared, without altering any other bits, simply by writing all ones except for the bit position to be cleared to IPRA or IPRB. Thus a fully polled interrupt scheme is possible. Note : writing a one to IPRA, IPRB has no effect on the interrupt pending register.

The interrupt mask registers (IMRA and IMRB) may be used to block a channel from making an interrupt request. Writing a zero into the corresponding bit of the mask register will still allow the channel to receive an interrupt and latch it into its pending bit (if that channel is enabled), but will prevent that channel from making an interrupt request. If that channel is causing an interrupt request at the time the corresponding bit in the mask register is cleared, the request will cease. If no other channel is making a request, INTR will go inactive. If the mask bit is reenabled, any pending interrupt is now free to resume its request unless blocked by a higher priority request for service. IMRA and IMRB are also readable . A conceptual circuit of an interrupt channel is shown in figure 10.

Figure 10 : A Conceptual Circuit of an Interrupt Channel.



There are two end-of-interrupt modes : the automatic end-of-interrupt mode and the software end-of-interrupt mode. The mode is selected by writing a one or a zero to the S bit of the Vector Register (VR). If the S bit of the VR is a one, all channels operate in the software end-of-interrupt mode. If the S bit is a zero, all channels operate in the automatic end-of-interrupt mode, and a reset is held on all in-service bits. In the automatic end-of-interrupt mode, the pending bit is cleared when that channel passes its vector. At that point, no further history of that interrupt remains in the MK68901 MFP. In the software end-of-interrupt mode, the in-service bit is set and the pending bit is cleared when the channel passes its vector. With the in-service bit set, no lower priority channel is allowed to request an interrupt or to pass its vector during an acknowledge sequence ; however, a lower priority channel may still receive an interrupt and latch it into the pending bit. A higher priority channel may still request an interrupt and be acknowledged. The in-service bit of a particular channel may be cleared by writing a zero to the corresponding bit in ISRA or ISRB. Typically, this will be done at the conclusion of the interrupt routine just before the return. Thus no lower priority channel will be allowed to request service until the higher priority channel is complete, while channels of still higher priority will be allowed to request service. While the in-service bit is set, a second interrupt on that channel may be received and latched into the pending bit, though no service request will be made in response to the second interrupt until the in-service bit is cleared. ISRA and ISRB may be read at any time. Only a zero may be written into any bit of ISRA and ISRB ; thus the in-service bits may be cleared in software but cannot be set in software. This allows any one bit to be cleared, without altering any other bits, simply by writing all ones except for the bit position to be cleared to ISRA or ISRB, as with IPRA and IPRB.

Figure 11 a : A Conceptual Circuit of the MK68901 MFP Daisy Chaining.

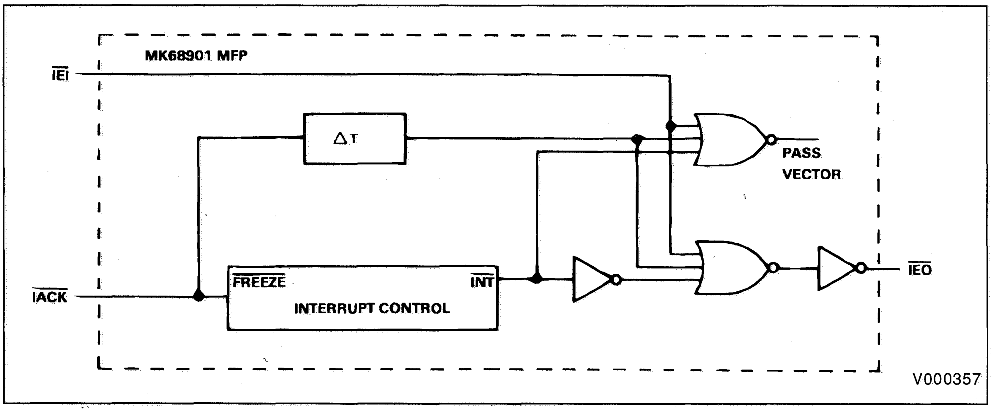
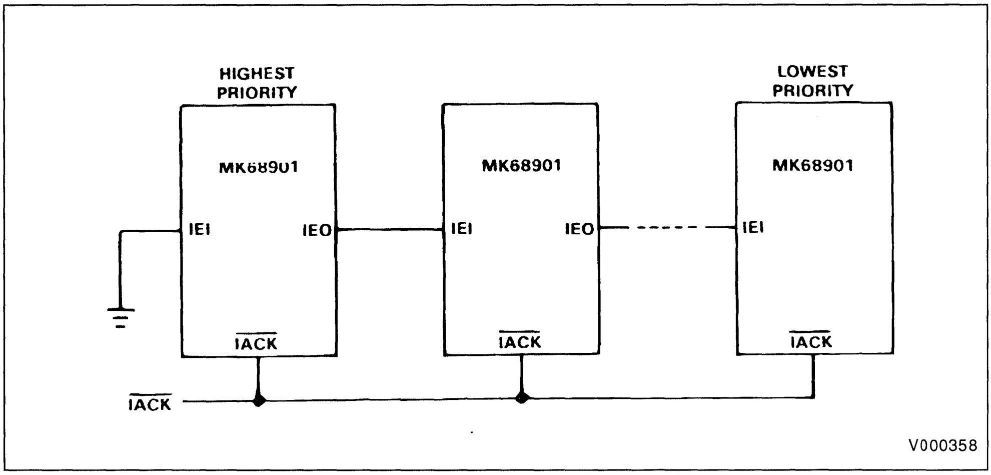


Figure 11 b : Daisy Chaining.



Each interrupt channel responds with a discrete 8-bit vector when acknowledged. The upper four bits of the vector are set by writing the upper four bits of the VR. The four low order bits (bit 3-bit 0) are generated by the interrupting channel.

To acknowledge an interrupt, $\overline{\text{IACK}}$ goes low, the $\overline{\text{IEI}}$ input must go low (or be tied low) and the MK68901 MFP must have an acknowledgeable interrupt pending. The Daisy Chaining capability (figure 11) requires that all parts in a chain have a common $\overline{\text{IACK}}$. When the common $\overline{\text{IACK}}$ goes low, all parts freeze and prioritize interrupts in parallel. Then priority is passed down the chain, via $\overline{\text{IEI}}$ and $\overline{\text{IEO}}$, until a part which has a pending interrupt is reached. The part with the pending interrupt, passes a vector, does not propagate $\overline{\text{IEO}}$, and generates $\overline{\text{DTACK}}$.

Figure 9 describes the 16 prioritized interrupt channels. As chown, General Purpose Interrupt 7 has the highest priority, while General Purpose Interrupt 0 is assigned the lowest priority. Each of these channels may be reprioritized, in effect, by selectively masking interrupts under software control. The binary numbers under "channel" correspond to the modified bits IV3, IV2, IV1 and IV0, respectively, of the Interrupt Vector for each channel (see figure 6).

Each channel has an enable bit contained in IERA or IERB, a pending latch contained in IPRA or IPRB, a mask bit contained in IMRA or IMRB, and an in–service latch contained in ISRA or ISRB. Additionally, the eight General Purpose Interrupts each have an edge bit contained in the Active Edge Register (AER), a bit to define the line as input or output contained in the Data Direction Register (DDR) and an I/O bit in the General Purpose Interrupt-I/O Port (GPIP).

TIMERS

There are four timers on the MK68901 MFP. Two of the timers (Timer A and Timer B) are full function timers which can perform the basic delay function and can also perform event counting, pulse width measurement, and waveform generation. The other two timers (Timer C and Timer D) are delay timers only. One or both of these timers can be used to supply the baud rate clocks for the USART. All timers are prescaler/counter timers with a common independent clock input (XTAL1, XTAL2). In addition, all timers have a time-out output, function that toggles each time the timer times out.

The four timers are programmed via three Timer Control Registers and four Timer Data Registers. Timers A and B are controlled by the control registers TACR and TBCR, respectively (see figure 12), and by the data registers TADR and TBDR (figure 13). Timers C and D are controlled by the control register TCDCR (see figure 14) and two data registers TCDR and TDDR. Bits in the control registers allow the selection of operational mode, prescale, and control white the data registers are used to read the timer or write into the time constant register. Timer A and B input pins TAI and TBI, are used for the event and pulse width modes for timers A and B.

With the timer stopped, no counting can occur. The timer contents will remain unaltered while the timer is stopped (unless reloaded by writing the Timer Data Register), but any residual count in the prescaler will be lost.

Figure 12 : Timer A and B Control Registers.

Port C (TACR)	*	*	*	TIMER A RESET	AC_3	AC_2	AC_1	AC_0
Port D (TBCR)	*	*	*	TIMER B RESET	BC_3	BC_2	BC_1	BC_0

C_3	C_2	C_1	C_0	
0	0	0	0	Timer Stopped
0	0	0	1	Delay Mode, : 4 Prescale
0	0	1	0	Delay Mode, 10 Prescale
0	0	1	1	Delay Mode, : 16 Prescale
0	1	0	0	Delay Mode, : 50 Prescale
0	1	0	1	Delay Mode, : 64 Prescale
0	1	1	0	Delay Mode, : 100 Prescale
0	1	1	1	Delay Mode, : 200 Prescale
1	0	0	0	Event Count Mode
1	0	0	1	Pulse Width Mode, : 4 Prescale
1	0	1	0	Pulse Width Mode, : 10 Prescale
1	0	1	1	Pulse Width Mode, 16 Prescale
1	1	0	0	Pulse Width Mode, : 50 Prescale
1	1	0	1	Pulse Width Mode, : 64 Prescale
1	1	1	0	Pulse Width Mode, 100 Prescale
1	1	1	1	Pulse Width Mode, : 200 Prescale

V000359

* Unused bits : read as zeros.

In the delay mode, the prescaler is always active. A count pulse will be applied to the main timer unit each time the prescribed number of timer clock cycles has elapsed. Thus, if the prescaler is programmed to divide by ten, a count pulse will be applied to the main counter every ten cycles of the timer clock.

Each time a count pulse is applied to the main counter, it will decrement its contents. The main counter is initially loaded by writing to the Timer Data Register. Each count pulse will cause the current count to decrement. When the timer has decremented down to "01" , the next count pulse will not cause it to decrement to "00". Instead, the next count pulse will cause the timer to be reloaded from the Timer Data Register. Additionally, a "Time out" pulse will be produced. This Time Out pulse is coupled to the timer interrupt channel, and, if that channel is enabled, an interrupt will be produced. The Time Out pulse is also coupled to the timer output pin and will cause the pin to change states. The output will remain in this new state until the next Time Out pulse occurs. Thus the output will complete one full cycle for each two Time Out pulses.

If, for example, the prescaler were programmed to divide by ten, and the Timer Data Register were loaded with 100 (decimal), the main counter would decrement once for every ten cycles of the timer clock. A Time Out pulse will occur (hence an interrupt if that channel is enabled) every 1000 cycles of the timer clock, and the timer output will complete one full cycle every 2000 cycles of the timer clock.

The main counter is an 8-bit binary down counter. It may be read at any time by reading the Timer Data Register. The information read is the information last clocked into the timer read register when the DS pin had last gone high prior to the current read cycle. When written, data is loaded into the Timer Data Register, and the main counter, if the timer is stopped. If the Timer Data Register is written while the timer is running, the new word is not loaded into the timer until it counts through H"01". However, if the timer is written while it is counting through H"01", an indeterminate value will be written into the timer constant register. This may be circumvented by ensuring that the data register is not written when the count is H"01".

If the main counter is loaded with "01", a Time Out Pulse will occur every time the prescaler presents a count pulse to the main counter. If loaded with "00", a Time Out pulse will occur every 256 count pulses.

Figure 13 : Timer Data Registers (A, B, C, and D).

Port F (TADR)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Port 10 (TBDR)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Port 11 (TCDR)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Port 12 (TDDR)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

V000360

Figure 14 : Timer C and D Register.

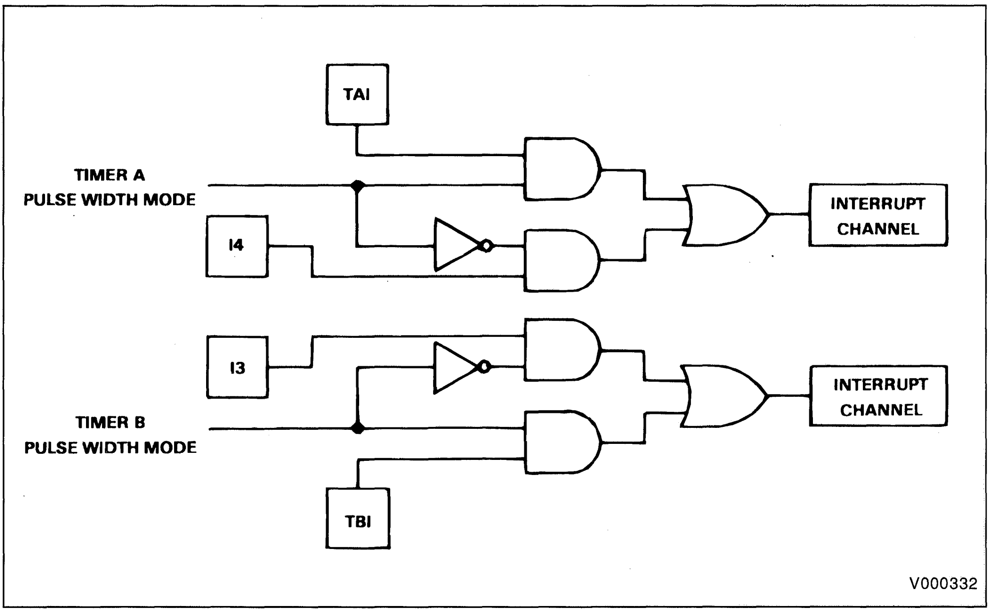
Port E (TCDCR)	*	CC_2	CC_1	CC_0	*	DC_2	DC_1	DC_0

C_2	C_1	C_0	
0	0	0	Timer Stopped
0	0	1	Delay Mode, ÷ 4 Prescale
0	1	0	Delay Mode, ÷ 10 Prescale
0	1	1	Delay Mode, ÷ 16 Prescale
1	0	0	Delay Mode, ÷ 50 Prescale
1	0	1	Delay Mode, ÷ 64 Prescale
1	1	0	Delay Mode, ÷ 100 Prescale
1	1	1	Delay Mode, ÷ 200 Prescale

V000361

* Unused bits : read as zeros.

Figure 15 : A Conceptual Circuit of the MFP Timers in the Pulse Width Measurement Mode.



Changing the prescale value with the timer running can cause the first Time Out pulse to occur at an indeterminate time, (no less than one nor more than 200 timer clock cycles times the number in the time constant register), but subsequent Time Out pulses will then occur at the correct interval.

In addition to the delay mode described above, Timers A and B can also function in the Pulse Width Measurement mode or in the Event Count mode. In either of these two modes, an auxiliary control signal is required. The auxiliary control input for Timer A is TAI, and for Timer B, TBI is used. The interrupt channels associated with I4 and I3 are used for TAI and TBI, respectively, in Pulse Width mode. See Figure 15.

The pulse width measurement mode functions much like the delay mode. However, in this mode, the auxiliary control signal on TAI or TBI acts as an enable to the timer. When the control signal on TAI or TBI is inactive, the timer will be stopped. When it is active, the prescaler and main counter are allowed to run. Thus the width of the active pulse on TAI or TBI is determined by the number of timer counts which occur while the pulse allows the timer to run. The active state of the signal on TAI or TBI is dependent upon the associated Interrupt Channel's edge bit (GPIP 4 for TAI and GPIP 3 for TBI : see Active Edge Register in figure 5). If the edge bit associated with the TAI or TBI input is a one, it will be active high ; thus the timer will be allowed to run when the input is at a high level. If the edge bit is a zero, the TAI or TBI input will be active low. As previously stated, the interrupt channel (I3 or I4) associated with the input still functions when the timer is used in the pulse width measurement mode. However, if the timer is programmed for the pulse width measurement mode, the interrupt caused by transitions on the associated TAI or TBI input will occur on the opposite transition.

For example, if the edge bit associated with the TAI input (AER-GPIP 4) is as one, an interrupt would normally be generated on the 0-1 transition of the I4 input signal. If the timer associated with this input (Timer A) is placed in the pulse width measurement mode, the interrupt will occur on the 1-0 transition of the TAI signal instead. Because the edge bit (AER-GPIP 4) is a one, Timer A will be allowed to count while the input is high. When the TAI input makes the high to low transition, Timer A will stop, and it is at this point that the interrupt will occur (assuming that the channel is enabled). This allows the interrupt to signal the CPU that the pulse being measured has terminated ; thus Timer A may now be read to determine the pulse width. (Again note that I3 and I4 may still be used for I/O when the timer is in the pulse width measurement mode). If Timer

A is reprogrammed for another mode, interrupts will again occur on the transition, as normally defined by the edge bit. Note that, like changing the edge bit, placing the timer into or taking it out of the pulse width mode can produce a transition on the signal to the interrupt channel and may cause an interrupt. If measuring consecutive pulses, it is obvious that one must read the contents of the timer and then reinitialize the main counter by writing to the timer data register. If the timer data register is written while the pulse is going to the active state, the write operation may result in an indeterminate value being written into the main counter. If the timer is written after the pulse goes active, the timer counts from the previous contents, and when it counts through H"01", the correct value is written into the timer. The pulse width then includes counts from before the timer was reloaded.

In the event count mode, the prescaler is disabled. Each time the control input on TAI or TBi makes an active transition as defined by the associated Interrupt Channel's edge bit, a count pulse will be generated, and the main counter will decrement. In all other respects, the timer functions as previously described. Altering the edge bit while the timer is in the event count mode can produce a count pulse. The interrupt channel associated with the input (I3 for I4 for TAI) is allowed to function normally. To count transitions reliably, the input must remain in each state (1/O) for a length of time equal to four periods of the timer clock ; thus signals of a frequency up to one fourth of the timer clock can be counted.

The manner in which the timer output pins toggle states has previously been described. All timer outputs will be forced low by a device RESET. The output associated with Timers A and B will toggle on each Time Out pulse regardless of the mode the timers are programmed to. In addition, the outputs from Timers A and B can be forced low at any time by writing a "1" to the reset location in TACR and TBCR, respectively. The output will be forced to the low state during the WRITE operation, and at the conclusion of the operation, the output will again be free to toggle each time a Time Out pulse occurs. This feature will allow waveform generation.

During reset, the Timer Data Registers and the main counters are not reset. Also, if using the reset option on Timers A or B, one must make sure to keep the other bits in the correct state so as not to affect the operation of Timers A and B.

USART

Serial Communication is provided by a full-duplex double-buffered USART, which is capable of either asynchronous or synchronous operation. Variable word length and start/stop bit configurations are available under software control for asynchronous operation. For synchronous operation, a Sync Word is provided to establish synchronization during receive operations. The Sync Word will also be repeatedly transmitted when no other data is available for transmission. Moreover, the MK68901 allows stripping of all Sync Words received in synchronous operation. The handshake control lines RR (Receiver Ready) and TR (Transmitter Ready) allow DMA operation. Separate receive and transmit clocks are available, and separate receive and transmit status and data bytes allow independent operation of the transmit and receive sections.

The USART is provided with three Control/Status Registers and a Data Register. The USART Data Register form is illustrated in figure 16. The programmer may specify operational parameters for the USART via the Control Register, as shown in figure 17. Status of both the Receiver and Transmitter sections is accessed by means of the two Status Registers, as shown in figures 18 and 19. Data written to the Data Register is passed to the transmitter, while reading the Data Register will access data received by the USART.

Figure 16 : USART Data Register.

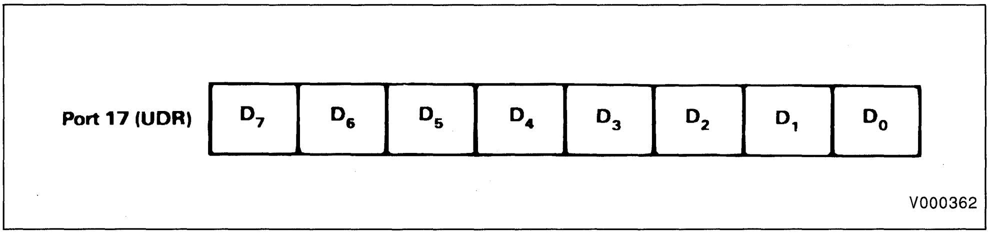
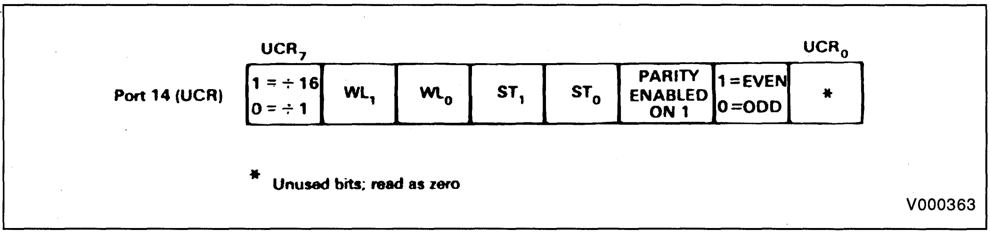


Figure 17 : USART Control Register (UCR).



÷ 16/÷ 1 : When this bit is zero, data will be clocked into and out of the receiver and transmitter at the frequency of their respective clocks. When this bit is loaded with a one, data will be clocked into and out of the receiver and transmitter at one sixteenth the frequency of their respective clocks. Additionally, when placed in the divide by sixteen mode, the receiver data transition resynchronization logic will be enabled.

WL0-WL1 :Word Length Control. These two bits set the length of the data word (exclusive of start bits, stop bits, and parity bits as follows:

WL1	WL0	Word Length
0	0	8 Bits
0	1	7 Bits
1	0	6 Bits
1	1	5 Bits

ST0-ST1 :Start/stop bit control (format control). These two bits set the format as follows :

ST1	ST0	Start Bits	Stop Bits	Format
0	0	0	0	SYNC
0	1	1	1	ASYNC
†1	0	1	$1^1/_2$	ASYNC
1	1	1	2	ASYNC

PARITY : Parity Enabled. When set ("1"), parity will be checked by the receiver, parity will be calculated, and a parity bit will be inserted by the transmitter. When cleared ("0") no parity check will be made and no parity bit will be inserted for transmission.

For a word length of 8 the MFP calculates the parity and appends it when transmitting a sync character. For shorter lengths, the parity must be stored in the Sync Character Register (SCR) along with the sync character.

E/O : Even-Odd. When set ("1"), even parity will be used if parity is enabled. When cleared ("0"), odd parity will be used if parity is enabled.

Note that the synchronous or asynchronous format may be selected independently of a ÷ 1 or ÷ 16 clock. Thus it is possible to clock data synchronously into the device but still use start and stop bits. In this mode, all normal asynchronous format features still apply. Data will be shifted in after a start bit is encountered, and a stop bit will be checked to determine proper framing. If a transmit underrun condition occurs, the output will be placed in a marking state, etc. It is conversely possible to clock data in asynchronously using a synchronous format. There is data transition detection logic built into the receive clock circuitry which will re-synchronize the internal shift clock on each data transition so that, with sufficienty frequent data transitions, start bits are not required. In this mode, all other common synchronous features function normally. This re-synchronization logic is only active in ÷ 16 clock mode.

RECEIVER

The receiver section of the USART is configured by the UCR as previously described. The status of the receiver can be determined by reading and writing to the Receiver Status Register (RSR). The RSR is configured as follows :

Figure 18 : Receiver Status Register (RSR).

	RSR$_7$							RSR$_0$
Port 15 (RSR)	BUFFER FULL	OVERRUN ERROR	PARITY ERROR	FRAME ERROR	FOUND/$\overline{\text{SEARCH}}$ OR BREAK DETECT	MATCH/CHARACTER IN PROGRESS	SYNC STRIP ENABLE	RECEIVER ENABLE

V000364

BF : Buffer Full. This bit is set when the incoming word is transferred to the receive buffer. The bit is cleared when the receive buffer is read by reading the UDR. This bit of the RSR is read only.

OE : Overrun Error. This flag is set if the incoming word is completely received and due to be tranferred to the receive buffer, but the last word in the receive buffer has not yet been read. When this condition occurs, the word in the receive buffer is not overwritten by the new word. Note that the status flags always reflect the status of the data word currently in the receive buffer. As such, the OE flag is not actually set until the good word currently in the buffer has been read. The interrupt associated with this error will also not be generated unti the old word in the receive buffer has been read.

OE flag is cleared by reading the receiver status register, and new data words cannot be shifted to the receive buffer until this is done.

PE : Parity Error. This flag is set if the word received has a parity error. The flag is set when the received word is tranferred from the shift register to the receive buffer if the error condition exists. The flag is cleared when the next word which does not have a parity error is tranferred to the receive buffer.

FE : Frame Error. This flag only applies to the asynchronous format. A frame error is defined as a non-zero data word which is not followed by a stop bit. Like the PE flag, the FE flag is set or cleared when a word is transferred to the receive buffer.

F/S : Found/$\overline{\text{Search}}$. This combination control bit and flag bit is only used with the synchronous format. It can be set or cleared by writing to this bit of the RSR. When this bit is cleared, the receiver is placed in the search mode. In this mode, a bit by bit comparison of the incoming data to the character in the Sync Character Register (SCR) is made. The word length counter is disabled. When a match is found, this bit will be set automatically, and the word length counter will start as sync has not been achieved. An interrupt will be generated on the receive error channel when the match occurs. The word just shifted in will, or necessity, be equal to the sync character, and it will not be transferred to the receive buffer.

B : Break. This flag is used only when the asynchronous format is selected. This flag will be set when an all zero data word, followed by no stop bit, is received. The flag will stay set until both a non-zero bit is received and the RSR has been read at least once since the flag was set. Break indication will not occur if the receive buffer is full.

M/CIP : Match/Character in Progress. If the synchronous format is selected, this flag is the Match flag. It will be set each time the word transferred to the receive buffer matches the sync character. It will be reset each time the word transferred to the receive buffer does not match the sync character. If the asynchronous format is selected, this flag represents Character in Progress. It will be set upon a start bit detect and cleared at the end of the word.

SS : Sync Strip Enable. If this bit is set to a one, data words that match the sync character will not be loaded into the receive buffer, and no buffer full signal will be generated.

RE : Receiver Enable. This control bit is used to enable or disable the receiver. If a zero is written to this bit of the RSR, the receiver will turn off immediately. All flags including the F/S bit will be cleared. If a one is written to this bit, normal receiver operation is enabled. The receive clock has to be running before the receiver is enabled.

There are two interrupt channels associated with the receiver. One channel is used for the normal Buffer Full condition, while the other channel is used whenever an error condition occurs. Only one interrupt is generated per word received, but dedicating two channels allows separate vectors : one for the normal condition, and one for an error condition. If the error channel is disabled, an interrupt will be generated via the Butter Full Channel, whether the word received is normal or in error. Those conditions which produce an interrupt via the error channel are : Overrun, Parity Error, Frame Error, Sync Found, and Break. If a received word has an error associated with it, and the error interrupt channel is enabled, an interrupt will occur on the error channel only.

Each time a word is transferred into the receive buffer, a corresponding set of flags is latched into the RSR. No flags (except CIP) are allowed to change until the data word has been read from the receive buffer. Reading the receive buffer allows a new data word to be transferred to the receive buffer when it is received. Thus one should first read the RSR then read the receive buffer (UDR) to ensure that the flags just read match the data word just read. If done in the reverse order, it is possible that subsequent to reading the data word from the receive buffer, but prior to reading the RSR, a new word may be received and transferred to the receive buffer and, with it, its associated flags latched into the RSR. Thus, when the RSR is read, those flags may actually correspond to a different data word. It is good practice, also to read the RSR prior to a data read as, when an overrun error occurs, the receiver will not assemble new characters until the RSR has been read.

As previously stated, when overrun occurs, the OE flag will not be set and the associated interrupt will not be generated until the receive buffer has been read. If a break occurs, and the receive buffer has not yet been read, only the B flag will be set (OE will not be set). Again, this flag will not be set until the last valid word has been read from the receive buffer. If the break condition ends and another whole data word is received before the receive buffer is read, both the B and OE flags will be set once the receive buffer is read.

If a break occurs while the OE flag is set, the B flag will also be set.

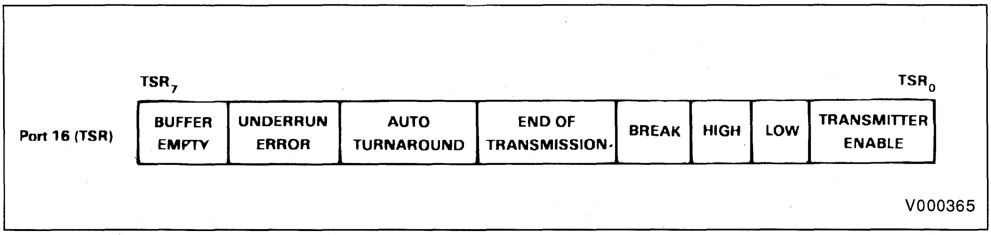
A break generates an interrupt when the condition occurs and again when the condition ends. If the break condition ends before it is acknowledged by reading the RSR, the receiver error interrupt indicating end of break will be generated once the RSR is read.

Anytime the asynchronous format is selected, start bit detection is enabled. New data is not shifted into the shift register until a zero bit is detected. If a ÷ 16 clock is selected, along with the asynchronous format, false start bit detection is also enabled. Any transition has to be stable for 3 positive going edges of the receive clock to be called a valid transition. For a start bit to be good, a valid 0-1 transition must not occur for 8 positive clock transitions after the initial valid 1-0 transition.

After a good start bit has been detected, valid transitions in the data are checked for continously. When a valid transition is detected, the counter is forced to state zero, and no more transition checking is started until state four. At state eight, the "previous state" of the transition checking logic is clocked into the receiver.

As a result of this resynchronization logic, it is possible to run with asynchronous clocks without start and stop bits if there are sufficient valid transitions in the data stream. This logic also makes the unit more tolerant of clock skew for normal asynchronous communications than a device which employs only start bit synchronization.

Figure 19 : Transmitter Status Register (TSR).



TRANSMITTER

The transmitter section of the USART is configured as to format, word length, etc. by the UCR, as previously described. The status of the transmitter can be determined by reading or writing the Transmitter Status Register (TSR). The TRS is configured as follows :

BE : Buffer Empty. This status bit is set when the word in the transmit buffer is transferred to the output shift register and thus the transmit buffer may be reloaded with the next data word. The flag is cleared when the transmit buffer is reloaded. The transmit buffer is loaded by writing to the UDR.

UE : This bit is set when the last word has been shifted out of the transmit shift register before a new word has been loaded into the transmit buffer. It is not necessary to clear this bit before loading the UDR.

This bit may be cleared by either reading the TSR or by disabling the transmitter. After the setting of the UE bit, one full transmitter clock cycle is required before this bit can be cleared by a read. The timing in some systems may allow a read of the TSR before the required clock cycle has been completed. This would result in the UE bit not being cleared until the following read. To avoid this problem, a dummy read of the TSR should be performed at the end of he UE service routine.

Only one underrun error may be generated between loads of the UDR regardless of the number of transmitter clock cycles between UDR loads.

AT : This bit causes the receiver to be enabled at the end of the transmission of the last word in the transmitter if the transmitter has been disabled.

END : End or Transmission. When the transmitter is turned off with a character still in the output shift register, transmission will continue until that character is shifted out. Once it has cleared the output register, the END bit will be set. If no character is being transmitted when the transmitter is disabled, the transmitter will stop at the next rising edge of the internal shift clock, and END will immediately be set. The END bit is cleared by re-enabling the transmitter.

B : Break. This control bit will cause a break to be transmitted. When a "1" is written to the B bit of the TSR, a break will be transmitted upon completion of the character (if any) currently being transmitted. A break will continue to be transmitted until the B bit is cleared by writing a "0" tot his bit of the TSR. At that time, normal transmission will resume. The B bit has no function in the synchronous format. Setting the "B" bit to a one keeps the "BE" bit from being set to a one. So, if there were a word in the buffer at the start of break, it would remain there until the end of break, at which time it would be transmitted (if the transmitter is still enabled). If the buffer were not full at the start of break, it could be written at any time during the break. If the buffer is empty at the end of break, the underrun flag will be set (unless the transmitter is disabled).

The BREAK bit cannot be set until the transmitter has been enabled and the transmitter has had sufficient time (one clock cycle) to perform the internal reset and initialization functions.

H,L : High and Low. These two control bits are used to configure the transmitter output, when the transmitter is disabled, as follows :

H	L	Output State
0	0	Hi-Z
0	1	Low ("0")
1	0	High
1	1	Loop-Connects transmitter output to receiver input, and TC to Receiver Clock (RC and SI are not used ; they are bypassed internally). In loop back mode, transmitter output goes high when disabled.

Figure 20 : SYNC Character Register.

Port 13 (SCR)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

V000366

Altering these two bits after Transmitter Enable (XE) is set will alter the output state until END is false. These bits should be set prior to enabling the transmitter. The state of these bits determine the state of the first transmitted character after the transmitter is enabled. If the high impedance mode was selected prior to the transmitter being enabled, the first bit transmitted is indeterminate.

XE : Transmitter Enable. This control bit is used to enable or disable the transmitter. When set, the transmitter is enabled. When cleared, the transmitter will be disabled. If disabled, any word currently in the output register will continue to be transmitted when XE is cleared, the transmitter will turn off at the end of the break character boundary, and no end of break stop bit is transmitted. The transmit clock must be running before the transmitter is enabled A "one" bit always precedes the first word out of the transmitter after the transmitter is enabled. There is a delay between the time the transmitter enable bit is written an when the transmitter reset goes low ; therefore, the H & L bits should be written with the desired state prior to enabling the transmitter.

Like the receiver section, there are two separate interrupt channels associated with the transmitter. The buffer Empty condition causes an interrupt via one channel, while the Underrun and END conditions will cause an interrupt via the second channel. When underrun occurs in the synchronous format, the character in the SCR will be transmitted until a new words is loaded into the transmit buffer. In the asynchronous format, a "Mark" will be continuously transmitted when underrun occurs.

The transmit buffer can be loaded prior to enabling the transmitter. When the transmitter is disabled, any character currently in the process of being transmitted will continue to conclusion, but any character in the transmit buffer will not be transmitted and will remain in the buffer. Thus no buffer empty interrupt will occur nor will the BE flag be ste. If the buffer were already empty, the BE flag would be set and would remain set. When the transmitter is disabled with a character in the output register but with no character in the transmit buffer, an Underrun Error will not occur when the character in progress concludes.

Often it is necessary to send a break for some particular period. To aid in timing a break transmission, a transmission, a transmit error interrupt will be generated at every normal character boundary time during a break transmission. The status register information is unaffected by this error condition interrupt. It should be noted that an underrun error, if present, must be cleared from the TSR, and the interrupt pending register must be cleared of pending transmitter errors at the beginning of the break transmission or no interrupts will be generated at the character boundary time.

It the synchronous format is selected, the sync character should be loaded into the Sync Character Register (SCR) as shouwn in figure 20. This character is compared to the received serial data during a Search, and will be continuously transmitted during an underrun condition.

All flags in the RSR or TSR will continue to function as described whether their associated interrupt channel is disabled or enabled. All interrupt channels are edge triggered and, in many cases, it is the actual output of a flag bit or flag bits which is coupled to the interrupt channel. thus, if a normal interrupt producing condition occurs while the interrupt channel is disabled, no interrupt would be produced even if the channel was subsequently enabled, because a transition did not occur while the interrupt channel was enabled. that particular flag bit would have to occur a second time before another "edge" was produced, causing an interrupt to be generated.

Error conditions in the USART are determined by monitoring the Receive Status Register and the Transmitter Status Register. These error conditions are only valid for each word boundary and are not latched. When executing block tranfers or data, it is necessary to save any errors so that they can be checked at the end of a block. In order to save error conditions during data transfer, the MK68901 MFP interrupt controller may be used by enabling error interrupt for the desired channel (Receive error or Transmit error) and by masking these bits off. Once the tranfer is complete, the Interrupt Pending Register can be polled, to determine the precence of a pending error interrupt, and therefore an error.

Unused bits in the sync character register are zeroed out ; therefore, word length should be set up prior to writing the sync word in some cases. Sync word length is the word length plus one when parity is enabled. The user has to determine the parity of the sync word when the word length is not 8 bits. The MK68901 MFP does not add a parity bit to the sync word if the word length is less than 8 bits. The extra bit in the sync word is transmitted as the parity bit. With a word length of eight, and parity selected, the parity bit for the sync word is computed an added on by the MK68901 MFP.

$\overline{\text{RR}}$ RECEIVER READY

$\overline{\text{RR}}$ is asserted when the Buffer Full bit is set in the RSR unless a parity error or frame error is detected by the receiver.

$\overline{\text{TR}}$ TRANSMITTER READY

$\overline{\text{TR}}$ is asserted when the Buffer Empty bit is set in the TSR unless a break is currently being transmitted.

REGISTER ACCESSES

All register accesses are dependent on CLK as shown in the timing diagrams. To read a register, $\overline{\text{CS}}$ and $\overline{\text{DS}}$ must be asserted, and R/$\overline{\text{W}}$ must by high. The internal read control signal is essentially the combination of CS, $\overline{\text{DS}}$, and RD/WR. Thus, the read operation will begin when CS and $\overline{\text{DS}}$ go active and will end when either CS or $\overline{\text{DS}}$ goes inactive. The address bus must be stable prior to the start of the operation and must remain stable until the end of the operation. Unless a read operation or interrupt acknowledge cycle is in progress the data bus (D_0-D_7) will remain in the tri-state condition.

To write a register, $\overline{\text{CS}}$ and $\overline{\text{DS}}$ must be asserted and R/$\overline{\text{W}}$ must be low. The address must be stable prior to the start of the operation and must remain stable until the end of the operation. After the MK68901 asserts $\overline{\text{DTACK}}$, the CPU negates $\overline{\text{DS}}$,. At this time, the MFP latches the data bus and writes the contents into the appropriate register. Also when $\overline{\text{DS}}$ is negated, the MFP rescinds $\overline{\text{DTACK}}$.

For an interrupt acknowledge, the operation starts when $\overline{\text{IACK}}$ goes low, and ends when IACK goes high. The data bus is tri-stated when either $\overline{\text{IACK}}$ or DS goes high.

When $\overline{\text{CS}}$ or $\overline{\text{IACK}}$ are asserted the MFP starts an internal cycle. $\overline{\text{DS}}$ is needed to enable the address and data buffers. It is recommended taht CS and IACK be gated by $\overline{\text{DS}}$ so that DS is always present whenever an MFP bus cycle starts.

MK68901 ELECTRICAL SPECIFICATIONS – PRELIMINARY

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
T_A	Temperature under Bias	– 25 to + 100	°C
T_{stg}	Storage Temperature	– 65 to + 150	°C
V_I	Voltage on Any Pin with Respect to Ground	– 0.3 to + 7	V
P_D	Power Dissipation	1.5	W

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 operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C ; V_{CC} = + 5V ± 5% Unless Otherwise Specified

Symbol	Parameter	Test Condition	Min.	Max.	Unit
V_{IH}	Input High Voltage		2.0	V_{CC} + .3	V
V_{IL}	Input Low Voltage		– 0.3	0.8	V
V_{OH}	Output High Voltage (except $\overline{\text{DTACK}}$)	I_{OH} = – 120µA	2.4		V
V_{OL}	Output Low Voltage (except $\overline{\text{DTACK}}$)	I_{OL} = 2.0mA		0.5	V
I_{LL}	Power Supply Current	Outputs Open		180	mA
I_{LI}	Input Leakage Current	V_{IN} = 0 to V_{CC}		± 10	µA
I_{LOH}	Tri-state Output Leakage Current in Float	V_{OUT} = 2.4 to V_{CC}		10	µA
I_{LOL}	Tri-state Output Leakage Current in Float	V_{OUT} = 0.5V		– 10	µA
I_{OH}	$\overline{\text{DTACK}}$ Output Source Current	V_{OUT} = 2.4		– 400	µA
I_{OL}	$\overline{\text{DTACK}}$ Output Sink Current	V_{OUT} = 0.5		5.3	mA

All voltages are referenced to ground.

CAPACITANCE

T_A = 25°C, f = 1MHz unmeasured pins returned to ground.

Symbol	Parameter	Test Condition	Max.	Unit
C_{IN}	Input Capacitance	Unmeasured pins returned to ground	10	pF
C_{OUT}	Tri-state Output Capacitance		10	pF

AC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0Vdc ± 5%, GND = 0Vdc, T_A = 0°C to 70°C)

Number	Characteristic	Value				Unit	Fig.	Note
		MK68901-4		MK68901-5				
		Min.	Max.	Min.	Max.			
1	$\overline{CS}$, $\overline{DS}$ Width High	50		35		ns	21,22	5
2	R/$\overline{W}$, A1-A5 Valid to Falling $\overline{CS}$ (setup)	0		0		ns	21,22	
3	Data Valid Prior to Falling CLK	280		0		ns	22	
4	$\overline{CS}$, $\overline{IACK}$ Valid to Falling Clock (setup)	50		45		ns	21-24	3
5	CLK Low to $\overline{DTACK}$ Low		220		180	ns	21,22	
6	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to $\overline{DTACK}$ High		60		55	ns	21-24	
7	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to $\overline{DTACK}$ Tri-state		100		95	ns	21-24	
8	$\overline{DTACK}$ Low to Data Invalid (hold time)	0		0		ns	22	
9	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to Data Tri-state		50		50	ns	21,23,24	
10	$\overline{CS}$ or $\overline{DS}$ High to R/$\overline{W}$, A1-A5 Invalid (hold time)	0		0		ns	21,22	
11	Data Valid from $\overline{CS}$ Low		310		260	ns	21	3,6
12	Read Data Valid to $\overline{DTACK}$ Low (setup)	50		50		ns	21	
13	$\overline{DTACK}$ Low to $\overline{DS}$, $\overline{CS}$ or $\overline{IACK}$ High (hold time)	0		0		ns	21-23	
14	$\overline{IEI}$ Low to Falling $\overline{CLK}$ (setup)	50		50		ns	23,24	
15	$\overline{IEO}$ Valid from Clock Low (delay)		180		180	ns	23	1
16	Data Valid from Clock Low (delay)		300		300	ns	23	
17	$\overline{IEO}$ Invalid from $\overline{IACK}$ High (delay)		150		150	ns	23, 24	
18	$\overline{DTACK}$ Low from Clock High (delay)		180		165	ns	23, 24	
19	$\overline{IEO}$ Valid from $\overline{IEI}$ Low (delay)		100		100	ns	24	1
20	Data Valid from $\overline{IEI}$ Low (delay)		220		220	ns	24	
21	Clock Cycle Time	250	1000	200	1000	ns	21	
22	Clock Width Low	110		90		ns	21	
23	Clock Width High	110		90		ns	21	
24	$\overline{CS}$, $\overline{IACK}$ Inactive to Rising Clock (setup)	100		80		ns	21-23	4,5
25	I/O Minimum Active Pulse Width	100		100		ns	25	
26	$\overline{IACK}$ Width High	2		2		T_{CLK}	23-24	2
27	I/O Data Valid from Rising $\overline{CS}$ or $\overline{DS}$		450		450	ns	26	
28	Receiver Ready Delay from Rising RC		600		600	ns	27	
29	Transmitter Ready Delay from Rising TC		600		600	ns	28	
30	Timer Output Low from Rising Edge of $\overline{CS}$ or $\overline{DS}$ (A & B) (reset T_{OUT})		450		450	ns	29	7
31	T_{OUT} Valid from Internal Timeout		2 t_{CLK} + 300		2 t_{CLK} + 300	ns	29	2
32	Timer Clock Low Time	110		90		ns	29	

AC ELECTRICAL CHARACTERISTICS (continued)
(V_{CC} = 5.0Vdc ± 5%, GND = 0Vdc, T_A = 0°C to 70°C)

Number	Characteristic	Value				Unit	Fig.	Note
		MK68901-4		MK68901-5				
		Min.	Max.	Min.	Max.			
33	Timer Clock High Time	110		90		ns	29	
34	Timer Clock Cycle Time	250	1000	200	1000	ns	29	
35	$\overline{\text{RESET}}$ Low Time	2		1.8		µs	30	
36	Delay to Falling $\overline{\text{INTR}}$ from External Interrupt Active Transition		380		380	ns	25	
37	Transmitter Internal Interrupt Delay from Falling Edge of TC		550		550	ns	28	
38	Receiver Buffer Full Interrupt Transition Delay from Rising Edge of RC		800		800	ns	27	
39	Receiver Error Interrupt Transition Delay from Falling Edge of RC		800		800	ns	27	
40	Serial in Set Up Time to Rising Edge of RC (divide by one only)	80		70		ns	27	
41	Data Hold Time from Rising Edge of RC (divide by one only)	350		325		ns	27	
42	Serial Output Data Valid from Falling Edge of TC (÷1)		440		420	ns	28	
43	Transmitter Clock Low Time	500		450		ns	28	
44	Transmitter Clock High Time	500		450		ns	28	
45	Transmitter Clock Cycle Time	1.05	∞	0.95	∞	µs	28	
46	Receiver Clock Low Time	500		450		ns	27	
47	Receiver Clock High Time	500		450		ns	27	
48	Receiver Clock Cycle Time	1.05	∞	0.95	∞	µs	27	
49	$\overline{\text{CS}}$, $\overline{\text{IACK}}$, $\overline{\text{DS}}$ Width Low		80		80	T_{CLK}	29	2
50	Serial Output Data Valid from Falling Edge of TC (÷16)		490		370	ns	28	

Notes :
1. $\overline{\text{IEO}}$ only goes low if no acknowledgeable interrupt is pending. If $\overline{\text{IEO}}$ goes low, $\overline{\text{DTACK}}$ and the data bus remain tri-stated.
2. T_{CLK} refers to the clock applied to the MFP CLK input pin. t_{CLK} refers to the timer clock signal, regardless of whether that signal comes from the XTAL 1/XTAL2 crystal clock inputs or the TAI or TBI timer inputs.
3. If the setup time is not met, $\overline{\text{CS}}$ or $\overline{\text{IACK}}$ will not be recognized until the next falling CLK.
4. If this setup time is met (for consecutive cycles), the minimum hold-off time of one clock cycle will be obtained. If not met, the hold-off will be two clock cycles.
5. $\overline{\text{CS}}$ is latched internally, therefore if spec's 1 and 24 are met then $\overline{\text{CS}}$ may be reasserted before the rising clock and still terminate the current bus cycle. The new bus cycle will be delayed by the MK68901 until all appropriate internal operations have completed.
6. Although $\overline{\text{CS}}$ and $\overline{\text{DTACK}}$ are synchronized with the clock, the data out during a read cycle is asynchronous to the clock, relying only on $\overline{\text{CS}}$ for timing.
7. Spec. 30 applies to timer outputs TAO and TBO only.

TIMER A.C. CHARACTERISTICS

Definitions :
Error = Indicated Time Value - Actual Time Value
tpsc = t_{CLK}x Prescale Value

INTERNAL TIMER MODE

Single Interval Error (free running) (note 2)	± 100ns
Cumulative Internal Error	0
Error between Two Timer Reads	± (tpsc + $4t_{CLK}$)
Start Timer to Stop Timer Error	+ ($2t_{CLK}$ + 100ns) to - (tpsc + $6t_{CLK}$ + 100ns)
Start Timer to Read Timer Error	+ 0 to – (tpsc + $6t_{CLK}$ + 400ns)
Start Timer to Interrupt Request Error (note 3)	- $2t_{CLK}$ to – ($4t_{CLK}$ + 800ns)

PULSE WIDTH MEASUREMENT MODE

Measurement Accuracy (note 1)	+ $2t_{CLK}$ to – (tpsc + $4t_{CLK}$)
Minimum Pulse Width	$4t_{CLK}$

EVENT COUNTER MODE

Minimum Active Time of TAI, TBI	$4t_{CLK}$
Minimum Inactive Time of TAI, TBI	$4t_{CLK}$

Notes : 1. Error may be cumulative if repetitively performed.
2. Error with respect to T_{OUT} or INT if note 3 is true.
3. Assuming it is possible for the timer to make an interrupt request immediately.

Figure 21 : Read Cycle.

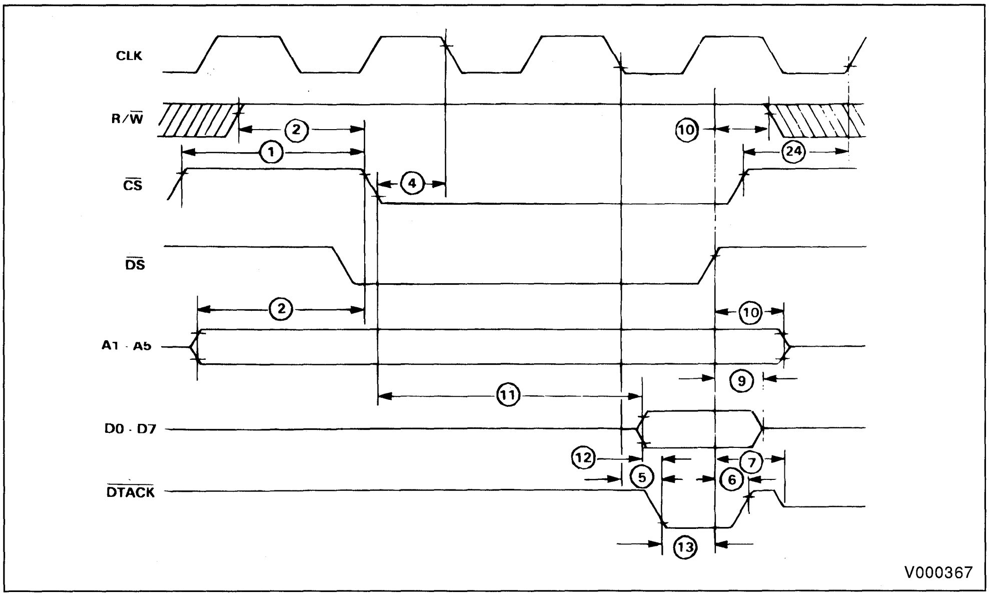
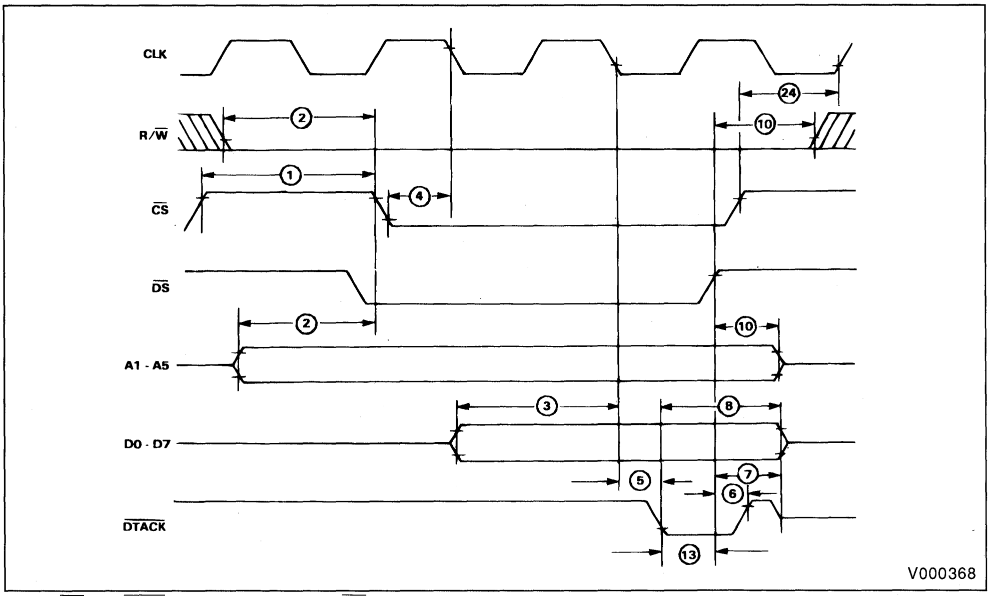


Figure 22 : Write Cycle.



Note : $\overline{CS}$ and $\overline{IACK}$ must be a function of $\overline{DS}$.

SGS-THOMSON MICROELECTRONICS

Figure 23 : Interrupt Acknowledge ($\overline{IEI}$ low).

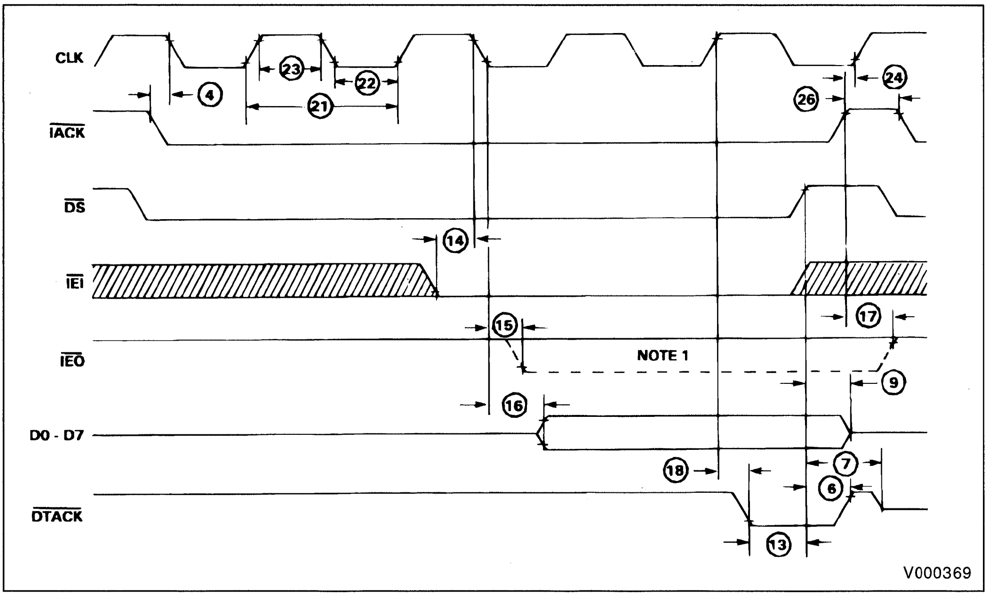
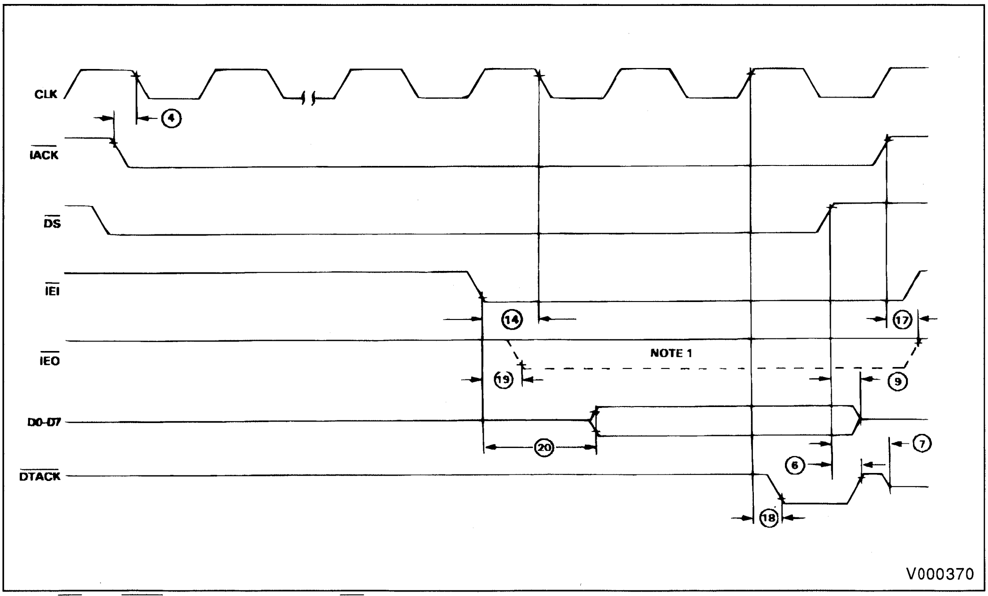
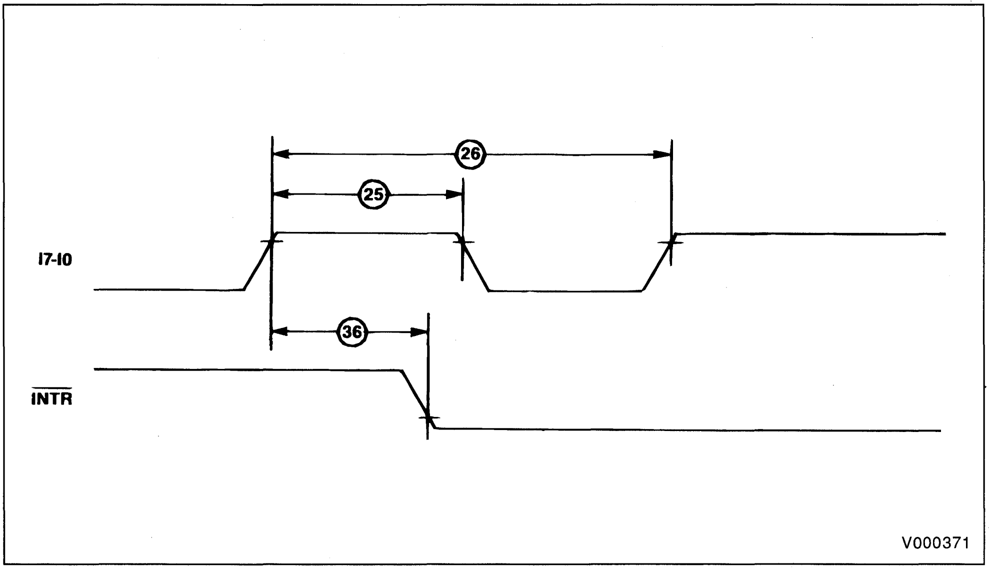


Figure 24 : Interrupt Acknowledge Cycle ($\overline{IEI}$ high).



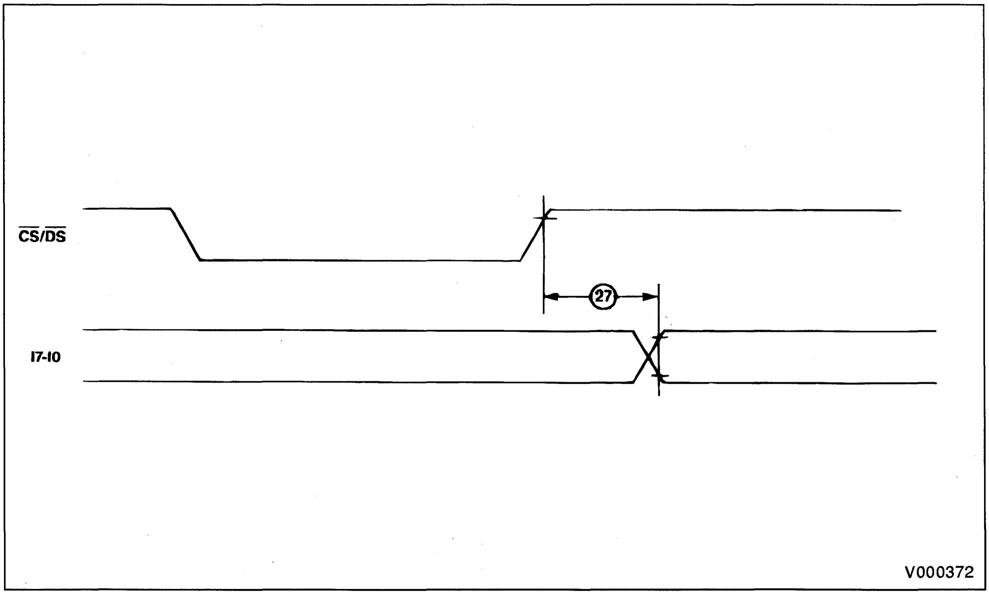
Note : $\overline{CS}$ and $\overline{IACK}$ must be a function of $\overline{DS}$.

Figure 25 : Interrupt Timing.



Note : Active edge is assumed to be the rising edge.

Figure 26 : Port Timing.



SGS-THOMSON MICROELECTRONICS

Figure 27 : Receiver Timing.

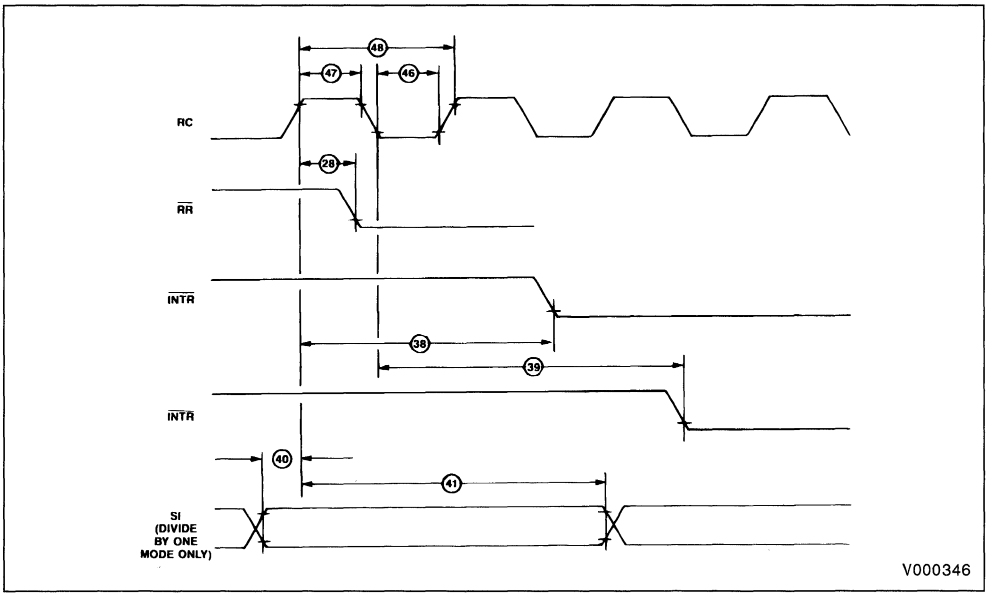


Figure 28 : Transmitter Timing.

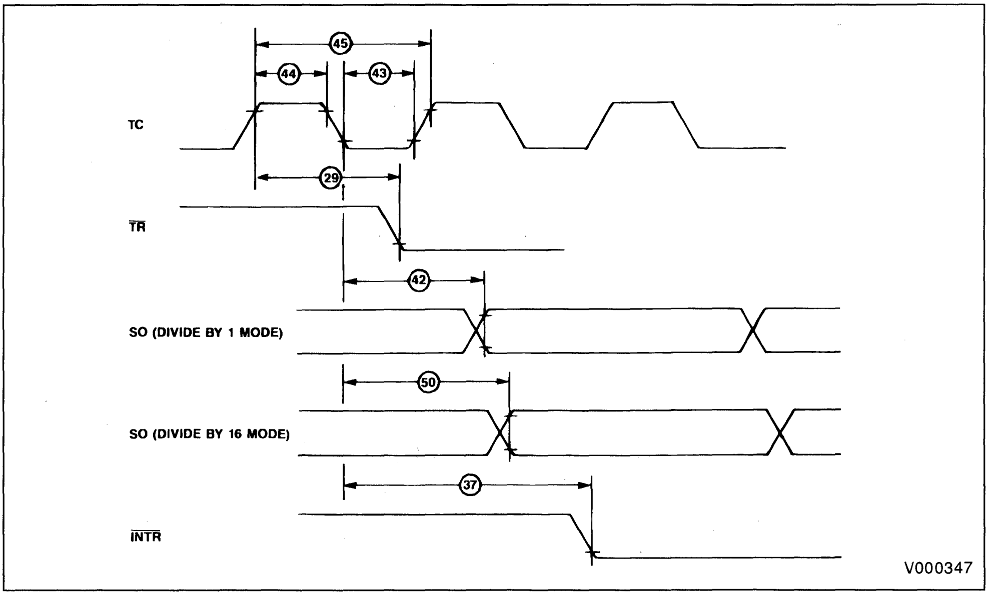


Figure 29 : Timer Timing.

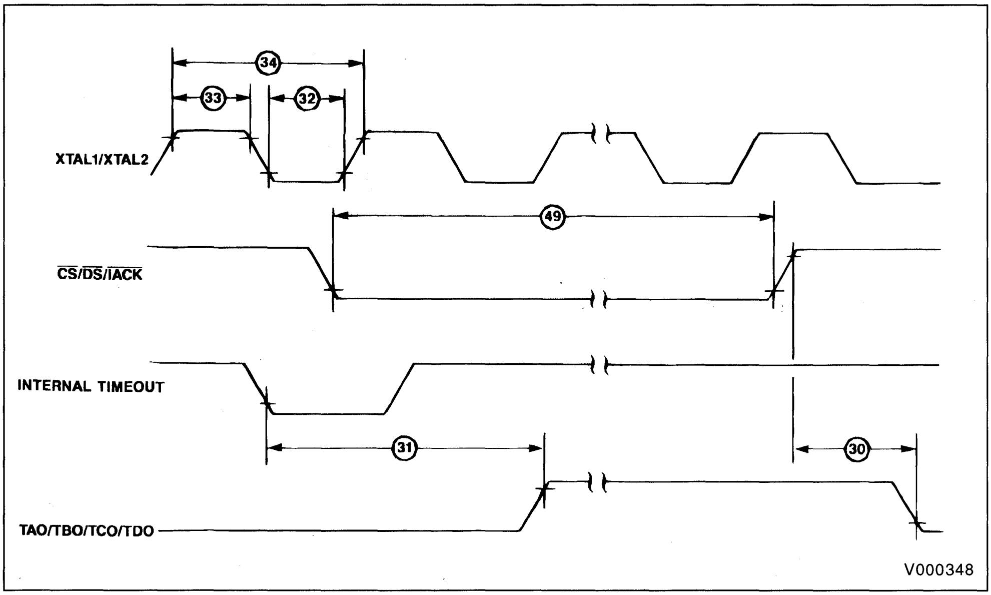


Figure 30 : Reset Timing.

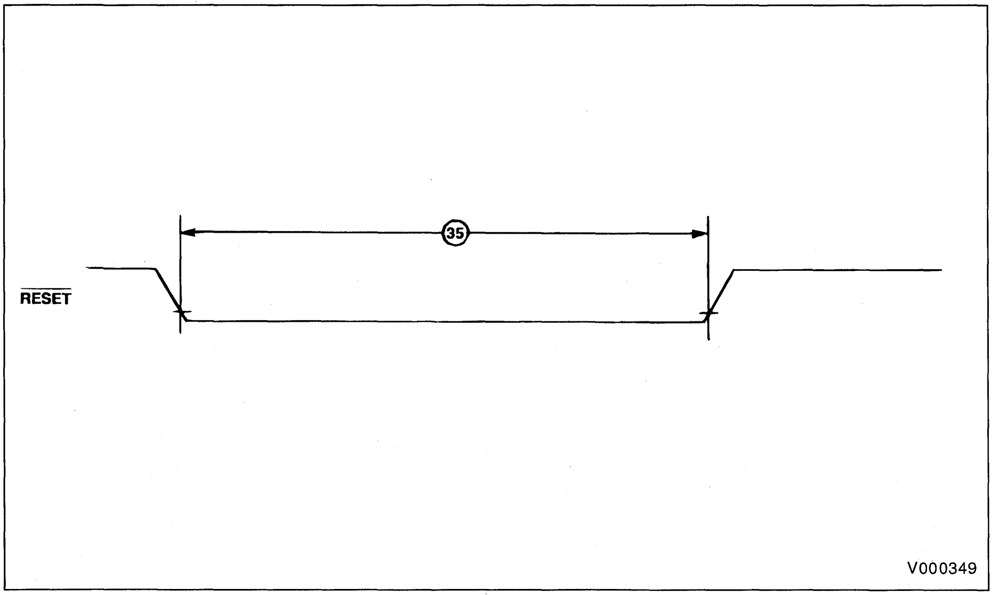


Figure 31 : Typical Output.

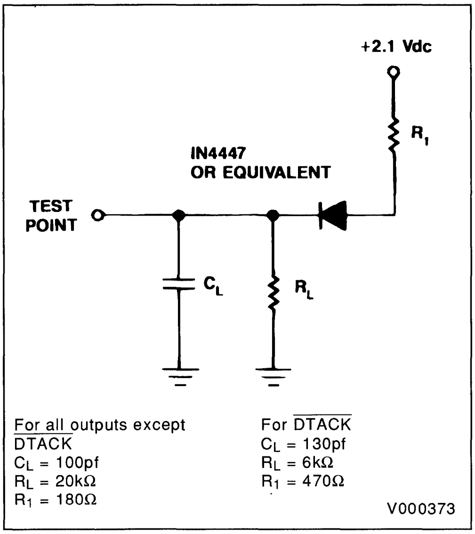


Figure 32 : INTR Test Load.

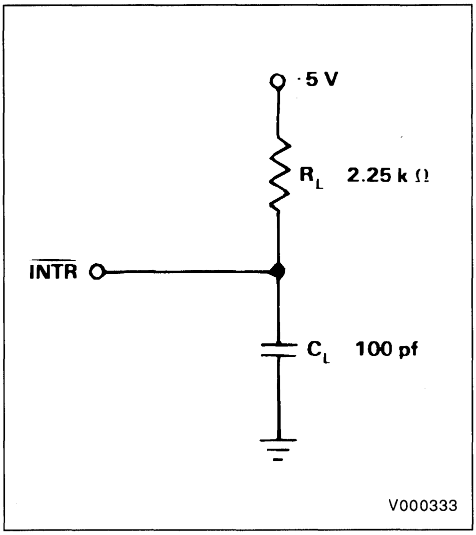
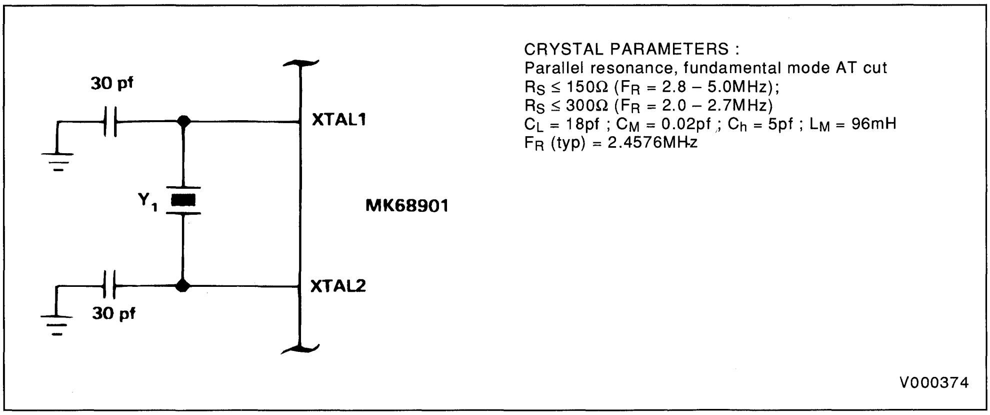


Figure 33 : MK68901 MFP External Oscillator Components.

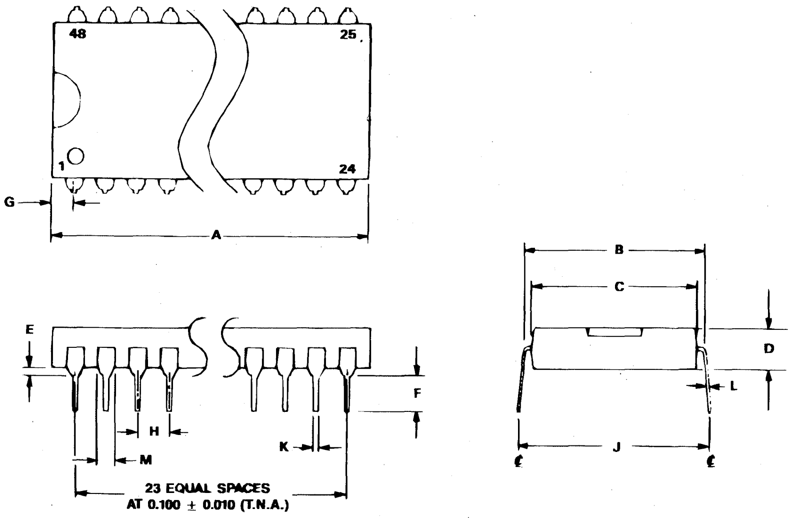


CRYSTAL PARAMETERS :
Parallel resonance, fundamental mode AT cut
$R_S \leq 150\Omega$ (F_R = 2.8 – 5.0MHz);
$R_S \leq 300\Omega$ (F_R = 2.0 – 2.7MHz)
C_L = 18pf ; C_M = 0.02pf ; C_h = 5pf ; L_M = 96mH
F_R (typ) = 2.4576MHz

MK68901 ORDERING INFORMATION

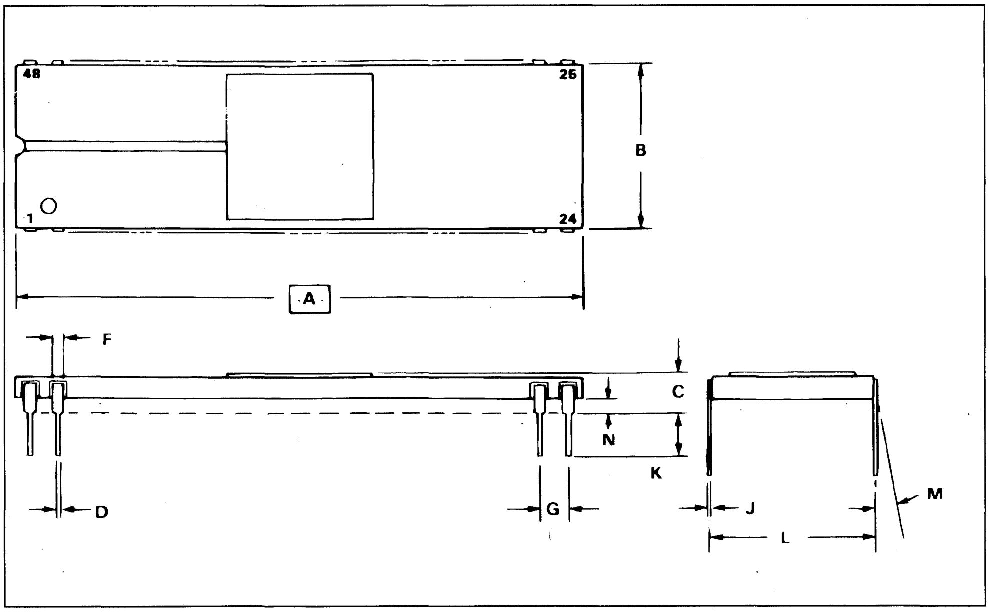
Part Number	Package Type	Max. Clock Frequency	Temperature Range
68901P04	Ceramic DIP	4.0MHz	0° to 70°C
68901P05	Ceramic DIP	5.0MHz	0° to 70°C
68901N04	Plastic DIP	4.0MHz	0° to 70°C
68901N05	Plastic DIP	5.0MHz	0° to 70°C
68901Q04	Plastic PLCC	4.0MHz	0° to 70°C
68901Q05	Plastic PLCC	5.0MHz	0° to 70°C

MK68901 48–PIN PLASTIC DUAL–IN–LINE PACKAGE (N)



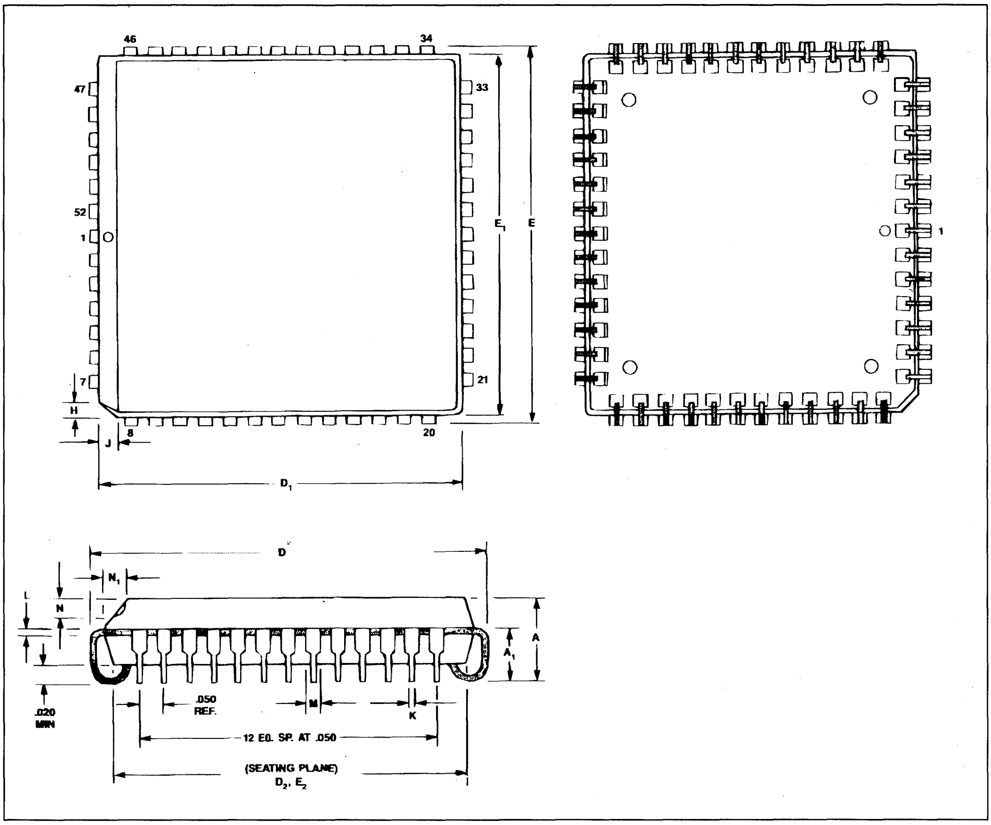
Dim	Millimeters		Inches	
	Min.	Max.	Min.	Max.
A	61.468	62.738	2.420	2.470
B	14.986	16.256	.590	.640
C	13.462	13.97	.530	.550
D	3.556	4064	.140	.160
E	0.381	1.524	.015	.060
F	3048	3.81	.120	.150
G	1.524	2.286	.060	.090
H	1.186	1.794	.090	.110
J	15.24	17.78	.600	.700
K	0.381	0.533	.015	.021
L	0.203	0.305	.008	.012
M	1.143	1.778	.045	.070

MK68901 48–PIN CERAMIC DUAL–IN–LINE PACKAGE (P)



Dim	Inches	
	Min.	Max.
A	2.376	2.424
B	0.576	0.604
C	0.120	0.160
D	0.015	0.021
F	0.030	0.055
G	0.100 BSC	
J	0.008	0.013
K	0.100	0.165
L	0.590	0.616
M	0°	10°
N	0.040	0.060

MK68901 52–PIN PLASTIC LEADED CHIP CARRIER (Q)



Dim	Inches	
	Min.	Max.
A	.165	.180
A_1	.090	.130
D	.785	.795
D_1	.750	.756
D_2	.690	.730
E	.785	.795
E_1	.750	.756
E_2	.690	.730
H	.042	.048
J	.042	.048
K	.013	.024
L	.008	.014
M	.026	.032
N/N_1	.043	.048

MK68901 PIN CONNECTIONS

PLCC	DIP	FUNC.	PLCC	DIP	FUNC.	PLCC	DIP	FUNC.
1	–	NC	19	17	XTAL1	37	33	$\overline{IEO}$
2	1	R/$\overline{W}$	20	18	XTAL2	38	34	$\overline{IEI}$
3	2	A1	21	–	NC	39	35	CLK
4	3	A2	22	19	TAI	40	36	GND
5	4	A3	23	20	TBI	41	37	D0
6	5	A4	24	21	$\overline{RESET}$	42	38	D1
7	6	A5	25	22	IO	43	39	D2
8	7	TC	26	23	I1	44	40	D3
9	8	SO	27	24	I2	45	41	D4
10	9	SI	28	25	I3	46	42	D5
11	10	RC	29	26	I4	47	43	D6
12	11	V_{CC}	30	27	I5	48	44	D7
13	–	NC	31	28	I6	49	45	$\overline{IACK}$
14	12	NC	32	29	I7	50	46	$\overline{DTACK}$
15	13	TAO	33	–	NC	51	47	$\overline{DS}$
16	14	TBO	34	30	$\overline{TR}$	52	48	$\overline{CS}$
17	15	TCO	35	31	$\overline{RR}$			
18	16	TDO	36	32	$\overline{INTR}$			

Note : NC – No Connection

SGS-THOMSON
MICROELECTRONICS

TS68HC901

HCMOS MULTI FUNCTION PERIPHERAL

PRELIMINARY INFORMATION

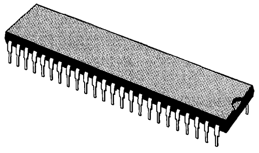
The TS68HC901 multi-function peripheral (CMFP) is a member of the 68000 Family of peripherals and the CMOS version of the MK68901. The CMFP directly interfaces to the 68000 processor via an asynchronous bus structure and can also support both multiplexed and non multiplexed buses. Both vectored, non vectored and polled interrupt schemes are supported, with the CMFP providing unique vector number generation for each of its 16 interrupt sources. Additionally, handshake lines are provided to facilitate DMAC interfacing.

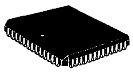
The TS68HC901 performs many of the functions common to most microprocessor-based systems. The resources available to the user include :

- Eight Individually Programmable I/O Pins with Interrupt Capability
- 16-Source Interrupt Controller with Individual Source Enabling and Masking
- Four Timers, Two of which are Multi-Mode Timers
- Timers may be used as Baud Rate Generators for the Serial Channel
- Single-Channel Full-Duplex Universal Synchronous / Asynchronous Receiver-Transmitter (USART) that Supports Asynchronous and with the Addition of a Polynominal Generator Checker Supports Byte Synchronous Formats.

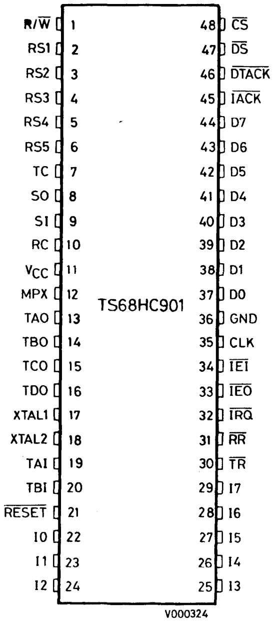
By incorporating multiple functions within the CMFP, the system designer retains flexibility while minimizing device' count.

The CMOS technology used for the TS68HC901 reduces also the power consumption of the system.


P
(Plastic Package)


FN
(PLCC52)

PIN CONNECTIONS



SECTION 1

INTRODUCTION

The TS68HC901 multi-function peripheral (CMFP) is a member of the 68000 peripherals. The CMFP directly interfaces to the 68000 processor via an asynchronous bus structure. Both vectored and polled interrupt schemes are supported, with the CMFP providing unique vector number generation for each of its 16 interrupt sources. Additionally, handshake lines are provided to facilitate DMAC interfacing. Refer to block diagram of the TS68HC901.

The TS68HC901 performs many of the functions common to most microprocessor-based systems.

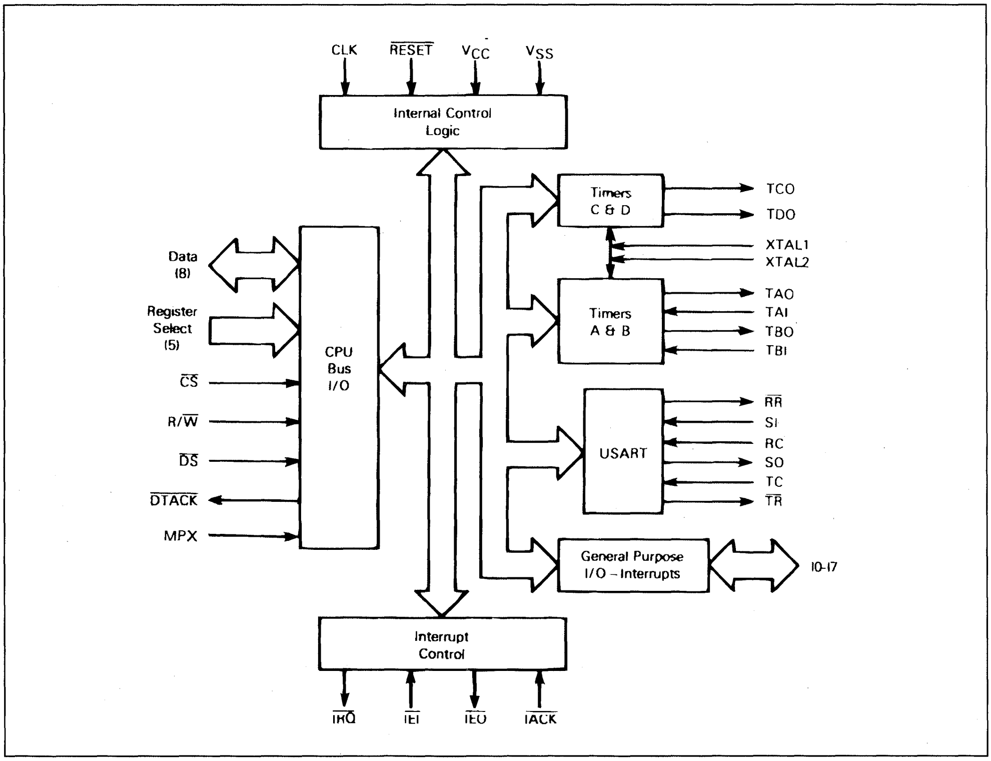
The resources available to the user include :

- Eight Individually Programmable I/O Pins with Interrupt Capability
- 16-Source Interrupt Controller with Individual Source Enabling and Masking
- Four Timers, Two of which are Multi-Mode Timers
- Timers May Be Used as Baud Rate Generators for the Serial Channel
- Single-Channel Full-Duplex Universal Synchronous / Asynchronous Receiver-Transmitter (USART) that Supports Asynchronous and with the Addition of a Polynomial Generator Checker Supports Byte Synchronous Formats

By incorporating multiple functions within the CMFP, the system designer retains flexibility while minimizing device count.

From a programmer's point of view, the versatility of

Figure 1.1 : Block Diagram.



the CMFP may be attributed to its register set. The registers are well organized and allow the CMFP to be easily tailored to a variety of applications. All of the 24 registers are also directly addressable which simplifies programming. The register map is shown in table 1.1.

Table 1.1 : CMFP Register Map.

Address						Abbreviation	Register Name
Hex	Binary						
	RS5	RS4	RS3	RS2	RS1		
01	0	0	0	0	0	GPIP	General Purpose I/O Register
03	0	0	0	0	1	AER	Active Edge Register
05	0	0	0	1	0	DDR	Data Direction Register
07	0	0	0	1	1	IERA	Interrupt Enable Register A
09	0	0	1	0	0	IERB	Interrupt Enable Register B
0B	0	0	1	0	1	IPRA	Interrupt Pending Register A
0D	0	0	1	1	0	IPRB	Interrupt Pending Register B
0F	0	0	1	1	1	ISRA	Interrupt In-service Register A
11	0	1	0	0	0	ISRB	Interrupt In-service Register B
13	0	1	0	0	1	IMRA	Interrupt Mask Register A
15	0	1	0	1	0	IMRB	Interrput Mask Register B
17	0	1	0	1	1	VR	Vector Register
19	0	1	1	0	0	TACR	Timer A Control Register
1B	0	1	1	0	1	TBCR	Timer B Control Register
1D	0	1	1	1	0	TCDCR	Timers C And D Control Register
1F	0	1	1	1	1	TADR	Timer A Data Register
21	1	0	0	0	0	TBDR	Timer B Data Register
23	1	0	0	0	1	TCDR	Timer C Data Register
25	1	0	0	1	0	TDDR	Timer D Data Register
27	1	0	0	1	1	SCR	Synchronous Character Register
29	1	0	1	0	0	UCR	USART Control Register
2B	1	0	1	0	1	RSR	Receiver Status Register
2D	1	0	1	1	0	TSR	Transmitter Status Register
2F	1	0	1	1	1	UDR	USART Data Register

Note : Hex addresses assume that RS1 connects with A1, RS2 connects with A2, etc... and that DS is connected to LDS on the 68000 or DS is connected to DS on the 68008.

SECTION 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 operations is also presented.

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.

2.1. SIGNAL DESCRIPTION

The input and output signals can be functionally organized into the groups shown in figure 2.1. The following paragraphs provide a brief description of the signal and a reference (if applicable) to other sections that contain more detail about its function.

2.1.1. V_{CC} AND GND. These inputs supply power to the CMFP. The V_{CC} is power at + 5 volts and GND is the ground connection.

2.1.2. CLOCK (CLK). The clock input is a single-phase TTL-compatible signal used for internal timing. This input should not be gated off at any time and must conform to minimum and maximum pulse width times. The clock is not necessarily the system clock in frequency nor phase. When the bus is multiplexed (MPX = 1), an address strobe signal is connected to this pin. In the non multiplexed mode (MPX = 0), this input is connected to the system clock when used with a 68000 processor type or to V_{SS} (0 V_{DC}) when used with a 6800 processor type.

2.1.3. ASYNCHRONOUS BUS CONTROL. Asynchronous data transfers are controlled by chip select, data strobe, read/write, and data transfer acknowledge. The low order register select lines, RS1-RS5, select an internal CMFP register for a read or write operation. The reset line initializes the CMFP registers and the internal control signals.

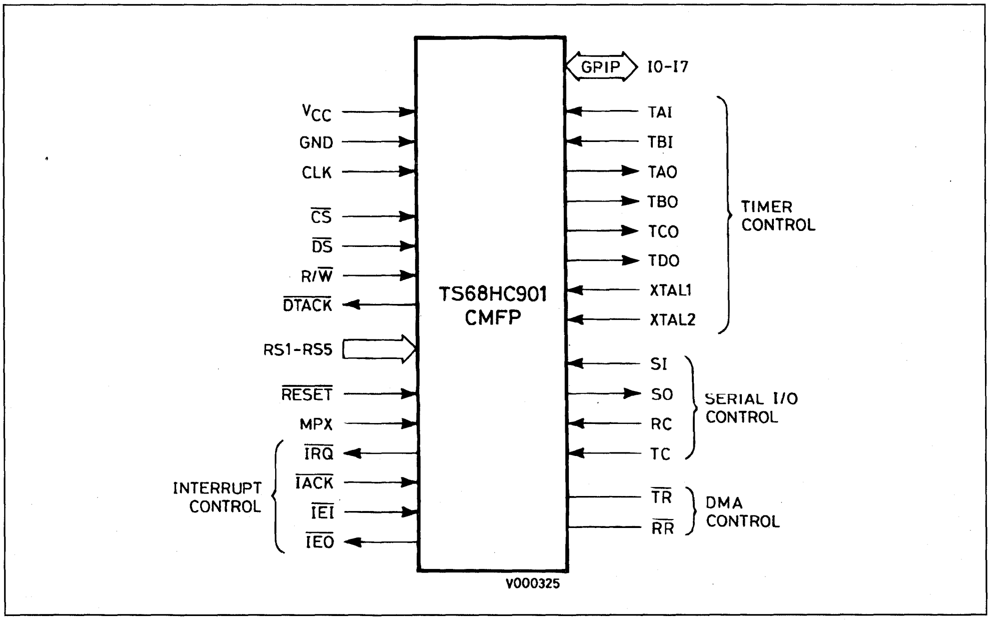
2.1.3.1. Chip Select ($\overline{CS}$).

This input activates the CMFP for internal register access.

2.1.3.2. Data Strobe ($\overline{DS}$).

This input is part of the internal chip select and interrupt acknowledge functions. The CMFP must be

Figure 2-1 : Input and Output Signals.



SGS-THOMSON MICROELECTRONICS

located on the lower portion of the 16-bit data bus so that the vector number passed to the processor during an interrupt acknowledge cycle will be located in the low byte of the data word. As a result, $\overline{\mathrm{DS}}$ must be connected to the processor's lower data strobe if vectored interrupts are to be used. Note that this forces all registers to be located at odd addresses and latches data on the rising edge for writes. This signal is used as $\overline{\mathrm{RD}}$ with an Intel processor type.

2.1.3.3. Read/Write (R/$\overline{\mathrm{W}}$).

This input defines a data transfer as a read (high) or a write (low) cycle. This signal is used as $\overline{\mathrm{WR}}$ with an Intel processor type.

2.1.3.4. Data Transfer Acknowledge ($\overline{\mathrm{DTACK}}$).

This output signals the completion of the operation phase of a bus cycle to the processor. If the bus cycle is a processor read, the CMFP asserts DTACK to indicate that the information on the data bus is valid. If the bus cycle is a processor to the CMFP, DTACK acknowledges the acceptance of the data by the CMFP. $\overline{\mathrm{DTACK}}$ will be asserted only by an CMFP that has $\overline{\mathrm{CS}}$ or $\overline{\mathrm{IACK}}$ (and $\overline{\mathrm{IEI}}$) asserted. This signal is not used with a 6800 processor type.

2.1.3.5. Register Select Bus (RS1 Through RS5).

The lower five bits if the register select bus select an internal CMFP register during a read or write operation.

2.1.3.6. Data Bus (D0 Through D7).

This bidirectional bus is used to receive data from or transmit data to the CMFP's internal registers during a processor read or write cycle. During an interrupt acknowledge cycle, the data bus is used to pass a vector number to the processor. Since the CMFP is an 8-bit peripheral, the CMFP could be located on either the upper or lower portion of the 16-bit data bus (even or odd address). However, during an interrupt acknowledge cycle, the vector number passed to the processor must be located in the low byte of the data word. As a result, D0-D7 of the CMFP must be connected to the low order eight bits of the processor data bus, placing CMFP registers at odd addresses if vectored interrupts are to be used.

2.1.3.7. Reset ($\overline{\mathrm{RESET}}$).

This input will initialize the CMFP during power up or in response to a total system reset. Refer to 2.2.3. for further information.

2.1.3.8. MPX.

This input selects the data bus mode :
MPX = 0 : non multiplexed mode
MPX = 1 : multiplexed mode. The register select lines RS1-RS5 and the data bus D0-D7 are multiplexed. An address strobe must be connected to the CLK pin.

2.1.4. INTERRUPT CONTROL. The interrupt request and interrupt acknowledge signals are handshake lines for a vectored interrupt scheme. Interrupt enable in and the interrupt enable out implement a daisy-chained interrupt structure.

2.1.4.1. Interrrupt Request ($\overline{\mathrm{IRQ}}$).

This output signals the processor that an interrupt is pending from the CMFP. These are 16 interrupt channels that can generate an interrupt request. Clearing the interrupt pending registers (IPRA and IPRB) or clearing the interrupt mask registers (IMRA and IMRB) will cause $\overline{\mathrm{IRQ}}$ to be negated. $\overline{\mathrm{IRQ}}$ will also be negated as the result of an interrupt acknowledge cycle, unless additional interrupts are pending in the CMFP. Refer to **SECTION 3** for further information.

2.1.4.2. Interrupt Acknowledge ($\overline{\mathrm{IACK}}$).

If both $\overline{\mathrm{IRQ}}$ and $\overline{\mathrm{IEI}}$ are active, the CMFP will begin an interrupt acknowledge cycle when $\overline{\mathrm{IACK}}$ and $\overline{\mathrm{DS}}$ are asserted. The CMFP will supply a unique vector number to the processor which corresponds to the interrupt handler for the particular channel requiring interrupt service. In a daisy-chained interrupt structure, all devices in the chain must have a common $\overline{\mathrm{IACK}}$. Refer to **2.2.2**. and **3.1.2**. for additional information.

2.1.4.3. Interrupt Enable In ($\overline{\mathrm{IEI}}$).

This input, together with the $\overline{\mathrm{IEO}}$ signal, provides a daisy-chained interrupt structure for a vectored interrupt scheme. $\overline{\mathrm{IEI}}$ indicates that no higher priority device is requesting interrupt service. So, the highest priority device in the chain should have its $\overline{\mathrm{IEI}}$ pin tied low. During an interrupt acknowledge cycle, an CMFP with a pending interrupt is not allowed to pass a vector number to the processor until its $\overline{\mathrm{IEI}}$ pin is asserted. When the daisy-chain option is not implemented, all CMFPs should have their $\overline{\mathrm{IEI}}$ pin tied low. Refer to **3.2**. for additional information.

2.1.4.4. Interrupt Enable Out ($\overline{\mathrm{IEO}}$).

This output, together with the $\overline{\mathrm{IEI}}$ signal, provides a daisy-chained interrupt structure for a vectored interrupt scheme. The $\overline{\mathrm{IEO}}$ of a particular CMFP signals lower priority devices that neither the CMFP nor any other higher-priority device is requesting interrupt service. When a daisy-chain is implemented, $\overline{\mathrm{IEO}}$ is tied to the next lower priority device's $\overline{\mathrm{IEI}}$ input. the lowest priority device's $\overline{\mathrm{IEO}}$ is not connected. When the daisy-chain option is not implemented, $\overline{\mathrm{IEO}}$ is not connected. Refer to 3.2 for additional information.

2.1.5. GENERAL PURPOSE I/O INTERRUPT LINES (I0 THROUGH I7). This is an 8-bit pin-programmable I/O port with interrupt capability. The data direction register (DDR) individually defines each line as either a high-impedance input or a TTL-compatible output. As an input, each line can generate an interrupt on the user selected transition of the input signal. Refer to **SECTION 4** for further information.

2.1.6. TIMER CONTROL. These lines provide internal timing and auxiliary timer control inputs required for certain operating modes. Additionally, the timer outputs are included in this group.

2.1.6.1. Timer Clock (XTAL1 AND XTAL2).

This input provides the timing signal for the four timers. A crystal can be connected between the timer clock inputs, XTAL1 and XTAL2, or XTAL2 can be driven with a CMOS-level clock while XTAL1 is grounded. The following crystal parameters are suggested :

a) Parallel resonance, fundamental mode AT-cut
b) Frequency tolerance measured with 18 picofarads load (0.1% accuracy) - drive level 10 microwatts
c) Shunt capacitance equals 7 picofarads maximum
d) Series resistance :
   $2.0 < f < 2.7MHz$; $R_S \leq 300\Omega$
   $2.8 < f < 4.0MHz$; $R_S \leq 150\Omega$

2.1.6.2. Timer Inputs (TAI AND TBI).

These inputs are control signals for timers A and B in the pulse width measurement mode and event count mode. These signals generate interrupts at the same priority level as the general purpose I/O interrupt lines I4 and I3, respectively. While I4 and I3 do not have interrupt capability when the timers are operated in the pulse width measurement mode or the event count mode, I4 and I3 may still be used for I/O. Refer to **5.1.2** and **5.1.3** for further information.

2.1.6.3. Timer Outputs (TAO, TBO, TCO, AND TDO).

Each timer has an associated output which toggles when its main counter counts through 01 (hexadecimal), regardless of which operational mode is selected. When in the delay mode, the timer output will be a square wave with a period equal to two timer cycles. This output signal may be used to supply the universal synchronous/asynchronous receiver-transmitter (USART) baud rate clocks. Timer outputs TAO and TBO may be cleared at any time by writing a one to the reset location in timer control registers A and B. Also, a device reset forces all timer outputs low. Refer to **5.2.2** for additional information.

2.1.7. SERIAL I/O CONTROL. The full duplex serial channel is implemented by a serial input and output line. The independent receive and transmit sections may be clocked by separate timing signals on the receiver clock input and the transmitter clock input.

2.1.7.1. Serial Input (SI).

This input line is the USART receiver data input. This input is not used in the USART loopback mode. Refer to **6.3.2** for additional information.

2.1.7.2. Serial Output (SO).

This output line is the USART transmitter data output. This output is driven high during a device reset.

2.1.7.3. Receiver Clock (RC).

This input controls the serial bit rate of the receiver. This signal may be supplied by the timer output lines or by any external TTL-level clock which meets the minimum and maximum cycle times. This clock is not used in the USART loopback mode. Refer to 6.3.2 for additional information.

2.1.7.4. Transmitter Clock (TC).

This input controls the serial bit rate of the transmitter. This signal may be supplied by the timer output lines or by an external TTL-level clock which meets the minimum and maximum cycle times.

2.1.8. DMA CONTROL. The USART supports DMA transfers through its receiver ready and transmitter ready status lines.

2.1.8.1. Receiver Ready ($\overline{RR}$).

This output reflects the receiver buffer full status for DMA operations.

2.1.8.2. Transmitter Ready ($\overline{TR}$).

This output reflects the transmitter buffer empty status for DMA operations.

2.1.9. SIGNAL SUMMARY. Table 2.1 is a summary of all the signals discussed in the previous paragraphs.

Table 2.1 : Signal Summary.

Signal Name	Mnemonic	I/O	Active
Power Input	V_{CC}	Input	High
Ground	GND	Input	Low
Clock	CLK	Input	N/A
Chip Select	$\overline{CS}$	Input	Low
Data Strobe	$\overline{DS}$	Input	Low
Read/Write	R/$\overline{W}$	Input	Read-high, Write-low
Data Transfer Acknowledge	$\overline{DTACK}$	Output	Low
Register Select Bus	RS1-RS5	Input	N/A
Data Bus	D0-D7	I/O	N/A
Reset	$\overline{RESET}$	Input	Low
Interrupt Request	$\overline{IRQ}$	Output	Low
Interrupt Acknowledge	$\overline{IACK}$	Input	Low
Interrupt Enable In	$\overline{IEI}$	Input	Low
Interrupt Enable Out	$\overline{IEO}$	Output	Low
General Purpose I/O - Interrupt Lines	I0-I7	I/O	N/A
Timer Clock	XTAL1, XTAL2	Input	High
Timer Inputs	TAI, TBI	Input	N/A
Timer Outputs	TAO, TBO, TCO, TDO	Output	N/A
Serial Input	SI	Input	N/A
Serial Output	SO	Output	N/A
Receiver Clock	RC	Input	N/A
Transmitter Clock	TC	Input	N/A
Receiver Ready	$\overline{RR}$	Output	Low
Transmitter Ready	$\overline{TR}$	Output	Low
MPX	$\overline{MPX}$	Input	N/A

2.2. BUS OPERATION

The following paragraphs explain the control signals and bus operation during data transfer operations and reset.

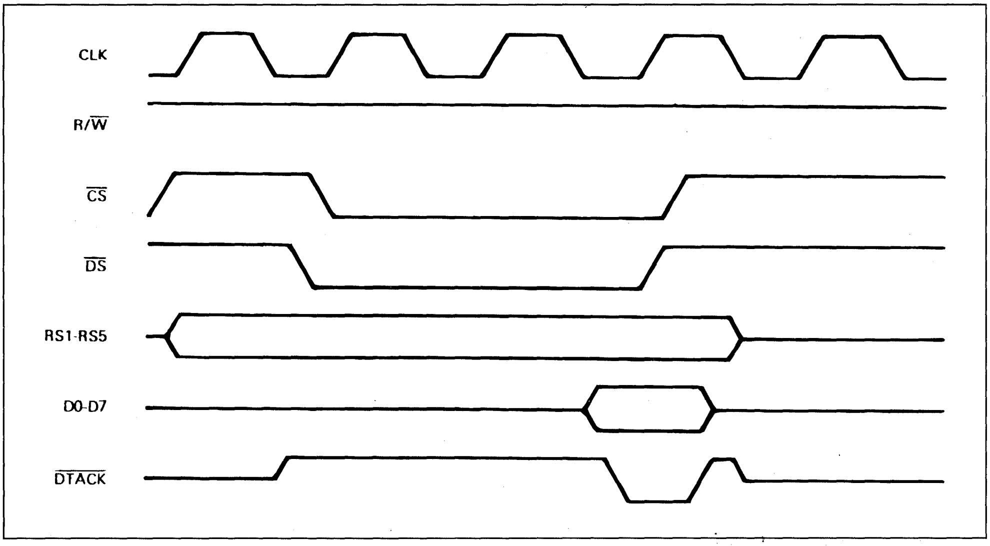
2.2.1. DATA TRANSFER OPERATIONS. Transfer of data between devices involves the following pins :
Register Select Bus - RS1 through RS5
Data Bus - D0 through D7
Control Signals
The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cycles, the bus master assumes responsibility for deskewing all signals it issues at both the start and end of a cycle. Additionally, the bus master is responsible for deskewing the acknowledge and data signals from the peripheral devices.

2.2.1.1. Read Cycle.

To read an CMFP register, $\overline{CS}$ and $\overline{DS}$ must be asserted, and R/$\overline{W}$ must be high. The CMFP will place the contents of the register which is selected by the register select bus (RS1 through RS5) on the data bus (D0 through D7) and then assert $\overline{DTACK}$. The register addresses are shown in table 1.1.

After the processor has latched the data, $\overline{DS}$ is negated. The negation of either $\overline{CS}$ or $\overline{DS}$ will terminate the read operation. The CMFP will drive $\overline{DTACK}$ high and place it in the high-impedance state. Also, the data bus will be in the high-impedance state. The timing for a read cycle is shown in figure 2.2. Refer to **7.7** for actual timing numbers.

Figure 2.2 : Read Cycle Timing.

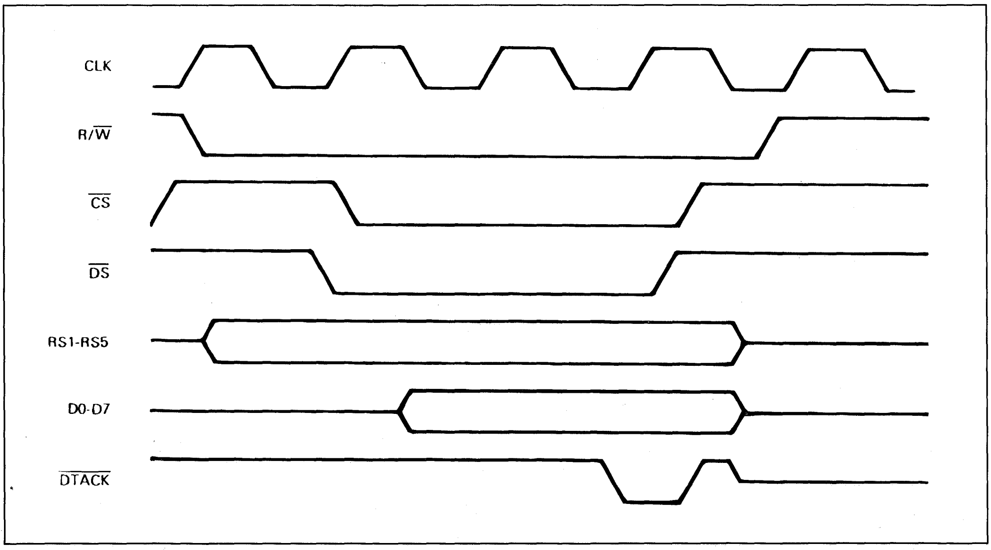


2.2.1.2. Write Cycle.

To write a register, $\overline{CS}$ and $\overline{DS}$ must be asserted, and R/W must be low. The CMFP will decode the address bus to determine which register is selected (the register map is shown in table 1.1). Then the register will be loaded with the contents of the data bus and DTACK will be asserted.

When the processor recognizes $\overline{DTACK}$, $\overline{DS}$ will be negated. The write cycle is terminated when either CS or DS is negated. The CMFP will drive DTACK high and place it in the high-impedance state. The timing for a write cycle is shown in figure 2.3. Refer to 7.7 for actual numbers.

Figure 2.3 : Write Cycle Timing.



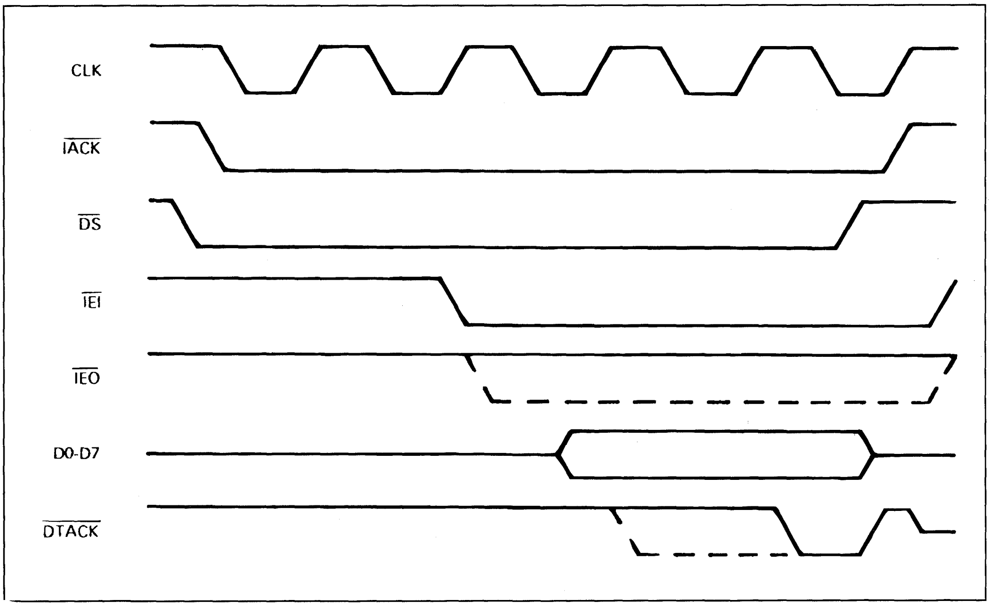
2.2.2. INTERRUPT ACKNOWLEDGE OPERATION. The CMFP has 16 interrupt sources, eight internal sources, and eight external sources. When an interrupt request is pending, the CMFP will assert IRQ. In a vectored interrupt scheme, the processor will acknowledge the interrupt request by performing an interrupt acknowledge cycle. IACK and DS will be asserted. The CMFP responds to the IACK signal by placing a vector number on the lower eight bits of the data bus. This vector number corresponds to the IRQ handler for the particular interrupt requesting service. The format of this vector number is given in figure 3.1.

When the CMFP asserts DTACK to indicate that valid data is on the bus, the processor will latch the data and terminate the bus cycle by negating DS. When either DS or IACK are negated, the CMFP will terminate the interrupt acknowledge operation by driving DTACK high and placing it in the high-impedance state. Also, the data bus will be placed in the high-impedance state. IRQ will be negated as a result of the IACK cycle unless additional interrupts are pending.

The CMFP can be part of a daisy-chain interrupt structure which allows multiple CMFPs to be placed at the same interrupt level by sharing a common IACK signal. A daisy-chain priority scheme is implemented with signals IEI and IEO. IEI indicates that no higher priority device is requesting interrupt service. IEO signals lower priority devices that neither this device nor any higher priority device is requesting service. To daisy-chain CMFPs, the highest priority CMFP has its IEI tied low and successive CMFPs have their IEI connected to the next higher priority device's IEO. Note that when the daisy-chain interrupt structure is not implemented, the IEI of all CMFPs must be tied low. Refer to **3.2** for additional information.

When the processor initiates an interrupt acknowledge cycle by driving IACK and DS, the CMFP whose IEI is low may respond with a vector number if an interrupt is pending. If this device does not have a pending interrupt, IEO is asserted which allows the next lower priority device to respond to the interrupt acknowledge. When an CMFP propagates IEO, it will not drive the data bus nor DTACK during the interrupt acknowledge cycle. The timing for an IACK cycle is shown in figure 2.4. Refer to **7.6** for further information.

Figure 2.4 : IACK Cycle Timing.



2.2.3. RESET OPERATION. The reset operation will initialize the CMFP to a known state. The reset operation requires that the $\overline{\text{RESET}}$ input be asserted for a minimum of two microseconds. During a device reset condition, all internal CMFP registers are cleared except for the timer data registers (TADR, TBDR, TCDR, and TDDR), the USART data register (UDR), the transmitter status register (TSR) and the interrupt vector register. All timers are stopped and the USART receiver and transmitter are disabled. The interrupt channels are also disabled and any pending interrupts are cleared. In addition, the general purpose interrupt I/O lines are placed in the high-impedance input mode and the timer outputs are driven low. External CMFP signals are negated. The interrupt vector register is initialized to a $0F.

2.2.4. NON MULTIPLEXED MODE. In this mode the MPX input must be set to zero, and the TS68HC901 can be used with a 68000 processor type or a 6800 processor type. Refer to figure 7.4, 7.5, 7.8 for the electrical characteristics.

With a 6800 processor type the $\overline{\text{DS}}$ pin is connected to the E signal of the processor, the $\overline{\text{DTACK}}$ signal is not used and the CLK must be zeroed.

2.2.5. MULTIPLEXED MODE. The CMFP can be used either on a MOTOROLA or INTEL bus type. In this case the MPX pin is connected to V_{CC}. The following table gives the signification of the different signals used. A dummy access to the TS68HC901 has to be done before any valid access in order to set up the internal logic of sampling.

Pin	MOTOROLA 6800 Type	MOTOROLA Multiplexed	INTEL
48	$\overline{\text{CS}}$	$\overline{\text{CS}}$	$\overline{\text{CS}}$
47	E	$\overline{\text{DS}}$	$\overline{\text{RD}}$
1	R/$\overline{\text{W}}$	R/$\overline{\text{W}}$	$\overline{\text{WR}}$
35	V_{SS}	AS	ALE

SECTION 3

INTERRUPT STRUCTURE

In a 68000 system, the CMFP will be assigned to one of the seven possible interrupt levels. All interrupt service requests from the CMFP's 16 interrupt channels will be presented at this level. Although, as an interrupt controller, the CMFP will internally prioritize its 16 interrupt sources. Additional interrupt sources may be placed at the same interrupt level by daisy-chaining multiple CMFPs. The CMFPs will be prioritized by their position in the chain.

3.1. INTERRUPT PROCESSING

Each CMFP provides individual interrupt capability for its various functions. When an interrupt is received on one of the external interrupt channels or from one of the eight internal sources, the CMFP will request interrupt service. The 16 interrupt channels are assigned a fixed priority so that multiple pending interrupts are serviced according to their relative importance. Since the CMFP can internally generate 16 vector numbers, the unique vector number which corresponds to the highest priority channel that has a pending interrupt is presented to the processor during an interrupt acknowledge cycle. This unique vector number allows the processor to immediately begin execution of the interrupt handler for the interrupting source, decreasing interrupt latency time.

3.1.1. INTERRUPT CHANNEL PRIORITIZATION. The 16 interrupt channels are prioritized as shown in table 3.1. General purpose interrupt 7 (I7) is the highest priority interrupt channel and I0 is the lowest priority channel. Pending interrupts are presented to the CPU in order of priority unless they have been masked off. By selectively masking interrupts, the channels are in effect re-prioritized.

3.1.2. INTERRUPT VECTOR NUMBER FORMAT. During an interrupt acknowledge cycle, a unique 8-bit vector number is presented to the system which corresponds to the specific interrupt source which is requesting service. The format of the vector is shown in figure 3.1. The most significant four bits of the interrupt vector number are user programmable. These bits are set by writing the upper four bits of the vector register which is shown in figure 3-2. The low order bits are generated internally by the TS68HC901. Note that the binary channel number shown in table 3.1 corresponds to the low order bits of the vector number associated with each channel.

Table 3.1 : Interrupt Channel Prioritization.

Priority	Channel	Description
Highest	1111	General Purpose Interrupt 7 (I7)
	1110	General Purpose Interrupt 6 (I6)
	1101	Timer A
	1100	Receiver Buffer Full
	1011	Receive Error
	1010	Transmitt Buffer Empty
	1001	Transmit Error
	1000	Timer B
	0111	General Purpose Interrupt 5 (I5)
	0110	General Purpose Interrupt 4 (I4)
	0101	Timer C
	0100	Timer D
	0011	General Purpose Interrupt 3 (I3)
	0010	General Purpose Interrupt 2 (I2)
	0001	General Purpose Interrupt 1 (I1)
Lowest	0000	General Purpose Interrupt 0 (I0)

Figure 3.1 : Interrupt Vector Format.

7	6	5	4	3	2	1	0
V7	V6	V5	V4	IV3	IV2	IV1	IV0

V7-V4	The four most significant bits are copied from the vector register
IV3-IV0	These bits are supplied by the CMFP. They are the binary channel number of the highest priority channel that is requesting interrupt service.

Figure 3.2 : Vector Register Format (VR).

	7	6	5	4	3	2	1	0
Address 17 (Hex)	V7	V6	V5	V4	S	*	*	*

*Unused bits are read as zero.

V7-V4 The upper four bits of the vector register are written by the user. These bits become the most significant four bits of the interrupt vector number.

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

S In-Service Register Enable. When the S bit is zero, the CMFP is in the automatic end-of-interrupt mode and the in-service register bits are forced low. When the S bit is a one, the CMFP is in the software end-of-interrupt mode and the in-Service register bits are enabled. Refer to **3.4.2** and **3.4.3** for additional information.

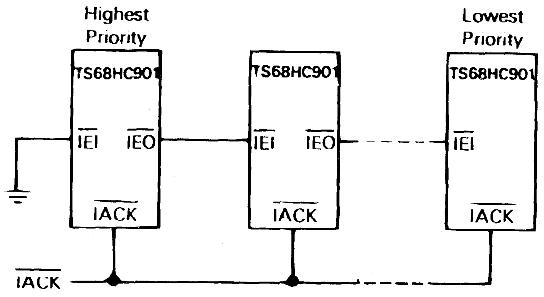
SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

3.2. DAISY-CHAINING CMFPs

As an interrupt controller, the TS68HC901 CMFP will support eight external interrupt sources in addition to its eight internal interrupt sources. When a system requires more than eight external interrupt sources to be placed at the same interrupt level, sources may be added to the prioritized structure by daisy-chaining CMFPs. Interrupt sources are prioritized internally within each CMFP and the CMFPs are prioritized by their position in the chain. Unique vector numbers are provided for each interrupt source.

The $\overline{\text{IEI}}$ and $\overline{\text{IEO}}$ signals implement the daisy-chained interrupt structure. The $\overline{\text{IEI}}$ of the highest priority CMFP is tied low and the $\overline{\text{IEO}}$ output of this device is tied to the next highest priority CMFP's $\overline{\text{IEI}}$. The $\overline{\text{IEI}}$ and $\overline{\text{IEO}}$ signals are daisy-chained in this manner for all CMFPs in the chain, with the lowest priority CMFP's $\overline{\text{IEO}}$ left unconnected. A diagram of an interrupt daisy-chain is shown in figure 3.3.

Figure 3.3 : Daisy-Chained Interrupt Structure.



SGS-THOMSON MICROELECTRONICS

Daisy-chaining requires that all parts in the chain have a common $\overline{\text{IACK}}$. When the common $\overline{\text{IACK}}$ is asserted during an interrupt acknowledge cycle, all parts will prioritize interrupts in parallel. When the $\overline{\text{IEI}}$ signal to a CMFP is asserted, the part may respond to the $\overline{\text{IACK}}$ cycle if it requires interrupt service. Otherwise, the part will assert $\overline{\text{IEO}}$ to the next lower priority device. Thus, priority is passed down the chain via $\overline{\text{IEI}}$ and $\overline{\text{IEO}}$ until a part which has a pending interrupt is reached. The part with the pending interrupt passes a vector number to the processor and does not propagate $\overline{\text{IEO}}$.

3.3. INTERRUPT CONTROL REGISTERS

CMFP interrupt processing is managed by the interrupt enable registers A and B, interrupt pending registers A and B, and interrupt mask registers A and B. These registers allow the programmer to enable or disable individual interrupt channels, mask individual interrupt channels, and access pending interrupt status information. In-service registers A and B allow interrupts to be nested as described in **3.4**. The interrupt control registers are shown in figure 3.4.

3.3.1. INTERRUPT ENABLE REGISTERS.

The interrupt channels are individually enabled or disabled by writing a one or zero, respectively, to the appropriate bit of interrupt enable register A (IERA) or interrupt enable register B (IERB). The processor may read these registers at any time.

When a channel is enabled, interrupts received on the channel will be recognized by the CMFP and $\overline{\text{IRQ}}$ will be asserted to the processor, indicating that interrupt service is required. On the other hand, a disabled channel is completely inactive ; interrupts received on the channel are ignored by the CMFP.

Writing a zero to a bit of interrupt enable register A or B will cause the corresponding bit of interrupt pending register A or B to be cleared. This will terminate all interrupt service requests for the channel and also negate $\overline{\text{IRQ}}$, unless interrupts are pending from other sources. Disabling a channel, however, does not affect the corresponding bit in interrupt in-service registers A or B. So, if the CMFP is in the software end-of-interrupt mode (see **3.4.3**) and an interrupt is in service when a channel is disabled, the in-service status bit for that channel will remain set until cleared by software.

Figure 3.4 : Interrupt Control Registers.

(a) Interrupt Enable Registers (IERA and IERB).

	7	6	5	4	3	2	1	0
Address 07 (Hex)	GPIP7	GPIP6	Timer A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	Timer B

	7	6	5	4	3	2	1	0
Address 09 (Hex)	GPIP5	GPIP4	Timer C	Timer D	GPIP3	GPIP2	GPIP1	GPIP0

When a bit is a zero, the associated interrupt channel is disabled. When a bit is a one, the associated interrupt channel is enabled.

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

Figure 3.4 : Interrupt Control Registers (continued).

(b) Interrupt Pending Registers (IPRA and IPRB).

	7	6	5	4	3	2	1	0
Address 0B (Hex)	GPIP7	GPIP6	Timer A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	Timer B

	7	6	5	4	3	2	1	0
Address 0D (Hex)	GPIP5	GPIP4	Timer C	Timer D	GPIP3	GPIP2	GPIP1	GPIP0

When a bit is a zero, no interrupt is pending on the associated interrupt channel. When a bit is a one, an interrupt is pending on the associated interrupt channel.

SET	a) Interrupt is received on an enabled interrupt channel
CLEARED	a) Interrupt vector for the associated interrupt channel is passed during an $\overline{\text{IACK}}$ cycle
	b) Associated interrupt channel is disabled
	c) MPU writes a zero
	d) Reset

(c) Interrupt In-Service Registers (ISRA and ISRB).

	7	6	5	4	3	2	1	0
Address 0F (Hex)	GPIP7	GPIP6	Timer A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	Timer B

	7	6	5	4	3	2	1	0
Address 11 (Hex)	GPIP5	GPIP4	Timer C	Timer D	GPIP3	GPIP2	GPIP1	GPIP0

When a bit is a zero, no interrupt processing is in progress for the associated interrupt channel. When a bit is a one, interrupt processing is in progress for the associated interrupt channel.

SET	a) Interrupt vector number for the associated interrupt channel is passed during an $\overline{\text{IACK}}$ cycle and the S bit of the vector register is set.
CLEARED	a) Interrupt service is completed for the associated interrupt channel
	b) The S bit of the vector register is a zero
	c) MPU writes a zero
	d) Reset

(d) Interrupt Mask Registers (IMRA and IMRB).

	7	6	5	4	3	2	1	0
Address 13 (Hex)	GPIP7	GPIP6	Timer A	RCV Buffer Full	RCV Error	XMIT Buffer Empty	XMIT Error	Timer B

	7	6	5	4	3	2	1	0
Address 15 (Hex)	GPIP5	GPIP4	Timer C	Timer D	GPIP3	GPIP2	GPIP1	GPIP0

When a bit is zero, interrupts are masked for the associated interrupt channel. When a bit is a one, interrupts are not masked for the associated interrupt channel.

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

3.3.2. INTERRUPT PENDING REGISTERS. When an interrupt is received on an enabled channel, the corresponding interrupt pending bit is set in interrupt pending register A or B (IPRA or IPRB). In a vectored interrupt scheme, this bit will be cleared when the processor acknowledges the interrupting channel and the CMFP responds with a vector number. In a polled interrupt system, the interrupt pending registers must be read to determine the interrupting channel and then the Interrupting pending bit is cleared by the interrupt handling routine without performing an interrupt acknowledge sequence.

A single bit of the interrupt pending registers is cleared in software by writing ones to all bit positions except the bit to be cleared. Note that writing ones to IPRA and IPRB has no effect on the contents of the register. A single bit of the interrupt pending registers is also cleared when the corresponding channel is disabled by writing a zero to the appropriate bit of IERA or IERB.

3.3.3. INTERRUPT MASK REGISTERS. Interrupts are masked for a channel by clearing the appropriate bit in interrupt mask register A or B (IMRA or IMRB). Even though an enabled channel is masked, the channel will recognize subsequent interrupts and set its interrupt pending bit. However, the channel is prevented from requesting interrupt service ($\overline{IRQ}$ to the processor) as long as the mask bit for that channel is cleared.

If a channel is requesting interrupt service at the time that its corresponding bit in IMRA or IMRB is cleared, the request will cease and $\overline{IRQ}$ will be negated, unless another channel is requesting interrupt service. Later, when the mask bit is set, any pending interrupt on the channel will be processed according to the channel's assigned priority. IMRA and IMRB may be read at any time.

3.4. NESTING CMFP INTERRUPTS

In a 68000 vectored interrupt system, the CMFP is assigned to one of seven possible interrupt levels. When an interrupt is received from the CMFP, an interrupt acknowledge for that level is initiated. Once an interrupt is recognized at a particular level, interrupts at that same level or below are masked by 68000. As long as the processor's interrupt mask is unchanged, the 68000 interrupt structure will prohibit the nesting of interrupts at the same interrupt level. However, additional interrupt requests from the CMFP can be recognized before a previous channel's interrupt service routine is completed by lowering the processor's interrupt mask to the next lower interrupt level within the interrupt handler.

When nesting CMFP interrupts, it may be desirable to permit interrupts on any CMFP channel, regardless of its priority, to preempt or delay interrupt processing of an earlier channel's interrupt service request. Or, it may be desirable to only allow subsequent higher priority channel interrupt requests to supercede previously recognized lower priority interrupt requests. The CMFP interrupt structure provides this flexibility by offering two end-of-interrupt options for vectored interrupt schemes. Note that the end-of-interrupt modes are not active in a polled interrupt scheme.

3.4.1. SELECTING THE END-OF-INTERRUPT MODE. In a vectored interrupt scheme, the CMFP may be programmed to operate in either the automatic end-of-interrupt mode or the software end-of-interrupt mode. The mode is selected by writing the S bit of the vector register (see figure 3.2). When the S bit is programmed to a one, the CMFP is placed in the software end-of-structure mode and when the S bit is a zero, all channels operate in the automatic end-of-interrupt mode.

3.4.2. AUTOMATIC END-OF-INTERRUPT. When an interrupt vector number is passed to the processor during an interrupt acknowledge cycle, the corresponding channel's interrupt pending bit is cleared. In the automatic end-of-interrupt mode, no further history of the interrupt remains in the CMFP. The in-service bits of the interrupt in-service registers (ISRA and ISRB) are forced low. Subsequent interrupts which are received on any CMFP channel will generate an interrupt request to the processor, even if the current interrupt's service routine has not been completed.

3.4.3. SOFTWARE END-OF-INTERRUPT. In the software end-of-interrupt mode, the channel's associated interrupt pending bit is cleared and in addition, the channel's in-service bit of in-service register A or B is set when its vector number is passed to the processor during an $\overline{IACK}$ cycle. A higher priority channel may subsequently request interrupt service and be acknowledged, but as long as the channel's in-service bit is set, no lower priority channel may request interrupt service nor pass its vector during an interrupt acknowledge sequence.

While only higher priority channels may request interrupt service, any channel can receive an interrupt and set its interrupt pending bit. Even the channel whose in-service bit is set can receive a second interrupt. However, no interrupt service request is made until its in-service bit is cleared.

The in-service bit for a particular channel can be cleared by writing a zero to its corresponding bit in ISRA or ISRB and ones to all other bit positions. Since bits in the in-service registers can only be cleared in software and not set, writing ones to the registers does not alter their contents. ISRA and ISRB may be read at any time.

SECTION 4

GENERAL PURPOSE INPUT/OUTPUT INTERRUPT PORT

The general purpose interrupt input/output (I/O) port (GPIP) provides eight I/O lines (I0 through I7) that may be operated as either inputs or outputs under software control. In addition, these lines may optionally generate an interrupt on either a positive transition or a negative transition of the input signal. The flexibility of the GPIP allows it to be configured as an 8-bit I/O port or for bit I/O. Since interrupts are enabled on a bit-by-bit basis, a subset of the GPIP could be programmed as handshake lines or the port could be connected to as many as eight external interrupt sources, which would be prioritized by the CMFP interrupt controller for interrupt service.

4.1. 6800 INTERRUPT CONTROLLER

The CMFP interrupt controller is particularly useful in a system which has many 6800-type devices. Typically, in a vectored 68000 system, 6800-type peripherals use the autovector which corresponds to their assigned interrupt level since they do not provide a vector number in response to an $\overline{\text{IACK}}$ cycle. The autovector interrupt handler must then poll all 6800-type devices at that interrupt level to determine which device is requesting service. However, by tying the $\overline{\text{IRQ}}$ output from a 6800-type device to the general purpose I/O interrupt port (GPIP) of a CMFP, a unique vector number will be provided to the processor during an interrupt acknowledge cycle. This interrupt structure will significantly reduce interrupt latency for 6800-type devices and other peripheral devices which do not support vector-by-device.

4.2. GPIP CONTROL REGISTERS

The GPIP is programmed via three control registers shown in figure 4.1. These registers control the data direction, provide user access to the port, and specify the active edge for each bit of the GPIP which will produce an interrupt. These registers are described in detail in the following paragraphs.

4.2.1. GPIP DATA REGISTER. The general purpose I/O data register is used to input or output data to the port. When data is written to the GPIP data register, those pins which are defined as inputs will remain in the high-impedance state. Pins which are defined as outputs will assume the state (high or low) of their corresponding bit in the data register. When the GPIP is read, data will be passed directly from the bits of the data register for pins which are defined as outputs. Data from pins defined as inputs will come from the input buffers.

4.2.2. ACTIVE EDGE REGISTER. The active edge register (AER) allows each of the GPIP lines to produce an interrupt on either a one-to-zero or a zero-to-one transition. Writing a zero to the appropriate edge bit of the active edge register causes the associated input to generate an interrupt on the one-to-zero transition. Writing a one to the edge bit will produce an interrupt on the zero-to-one transition of the corresponding GPIP line.

Figure 4.1 : GPIP Control Registers.

(a) GPIP Data Register (GPIP).

	7	6	5	4	3	2	1	0
Address 01 (Hex)	GPIP7	GPIP6	GPIP5	GPIP4	GPIP3	GPIP2	GPIP1	GPIP0

SET a) MPU writes a one

(b) Active Edge Register (AER)

	7	6	5	4	3	2	1	0
Address 03 (Hex)	GPIP7	GPIP6	GPIP5	GPIP4	GPIP3	GPIP2	GPIP1	GPIP0

CLEARED a) MPU writes a zero

When a bit is a zero, interrupts will be generated on the falling edge of the associated input signal. When a bit is a one, interrupts will be generated on the rising edge of the associated input signal

SET a) MPU writes a one

CLEARED a) MPU writes a zero

Figure 4.1 : GPIP Control Registers (continued).

(c) Data Direction Register (DDR).

	7	6	5	4	3	2	1	0
Address 05 (Hex)	GPIP7	GPIP6	GPIP5	GPIP4	GPIP3	GPIP2	GPIP1	GPIP0

When a bit is a zero, the associated I/O line is defined to be an input. When a bit is a one, the associated I/O line is defined to be an output

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

Note : The transition detector is an exclusive-OR gate whose inputs are the edge bit and the input buffer. As a result, writing the AER may cause an interrupt-producing transition, depending upon the state of the input. So, the AER should be configured before enabling interrupts via the interrupt enable registers (IERA and IERB). Also, changing the edge bit while interrupts are enabled may cause an interrupt on the corresponding channel.

4.2.3. DATA DIRECTION REGISTER. The data direction register (DDR) allows the programmer to define I0 through I7 as inputs or outputs by writing the corresponding bit. When a bit of the data direction register is written as a zero, the corresponding interrupt I/O pin will be a high-impedance input. Writing a one to any bit of the data direction register will cause the corresponding pin to be configured as a push-pull output.

SECTION 5

TIMERS

The CMFP contains four 8-bit timers which provide many functions typically required in microprocessor systems. The timers can supply the baud rate clocks for the on-chip serial I/O channel, generate periodic interrupts, measure elapsed time, and count signal transitions. In addition, two timers have waveform generation capability.

All timers are prescaler/counter timers with a common independent clock input (XTAL1 or XTAL2) and are not required to be operated from the system clock. Each timer's output signal toggles when the timer's main counter times out. Additionally, timers A and B have auxiliary control signals which are used in two of the operation modes. An interrupt channel is assigned to each timer and when the auxiliary control signals are used, a separate interrupt channel will respond to transitions on these inputs.

5.1. OPERATION MODES

Timers A and B are full function timers which, in addition to the delay mode, operate in the pulse width measurement mode and the event count mode. Timers C and D are delay timers only. A brief discussion of each of the timer modes follows.

5.1.1. DELAY MODE OPERATION. All timers may operate in the delay mode. In this mode, the prescaler is always active. The prescaler specifies the number of timer clock cycles which must elapse before a count pulse is applied to the main counter. A count pulse causes the main counter to decrement by one. When the timer has decremented down to 01 (hexadecimal), the next count pulse will cause the main counter to be reloaded from the timer data register and a time out pulse will be produced. This time out pulse is coupled to the timer's interrupt channel and, if the channel is enabled, an interrupt will occur. The time out pulse also causes the timer output pin to toggle. The output will remain in this new state until the next time out pulse occurs.

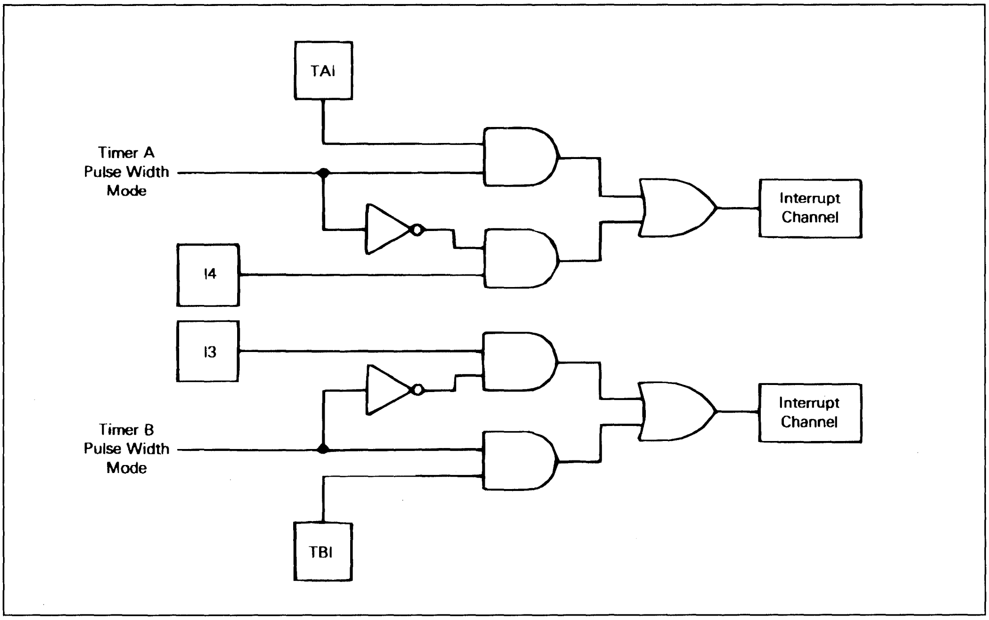
For example, if delay mode with a divide-by-10 prescaler is selected and the timer data register is loaded with 100 (decimal), the main counter will decrement once every 10 timer clock cycles. After 1,000 timer clocks, a time out pulse will be produced. This time out pulse will generate an interrupt if the channel is enabled (IERA, IERB) and in addition, the timer's output line will toggle. The output line will complete one full period every 2,000 cycles of the timer clock.

If the prescaler value is changed while the timer is enabled, the first time out pulse will occur at an indeterminate time no less than one nor more than 200 timer clock cycles. Subsequent time out pulses will then occur at the correct interval.

If the main counter is loaded with 01 (hexadecimal), a time out pulse will occur every time the prescaler presents a count pulse to the main counter. If the main counter is loaded with 00, a time out pulse will occur every 256 count pulses.

5.1.2. PULSE WIDTH MEASUREMENT OPERATION. Besides the delay mode, timers A and B may be programmed to operate in the pulse width measurement mode. In this mode an auxiliary control input is required ; timers A and B auxiliary input lines are TAI and TBI. Also, in the pulse width measurement mode, interrupt channels normally associated with I4 and I3 will respond to transitions on TAI and TBI, respectively. General purpose lines I3 and I4 may still be used for I/O. A conceptual circuit of the timers in the pulse width measurement mode is shown in figure 5.1.

Figure 5.1 : Conceptual Circuit of Timers A and B in Pulse Width Measurement Mode.



The pulse width measurement mode functions similarly to the delay mode, with the auxiliary control signal acting as an enable to the timer. When the control signal is active, the prescaler and main counter are allowed to operate. When the control signal is negated, the timer is stopped. So, the width of the active pulse on TAI or TBI is measured by the number of timer counts which occur while the timer is allowed to operate.

The active state of the auxiliary input line is defined by the associated interrupt channel's edge bit in the active edge register (AER). GPIP4 of the AER is the edge bit associated with TAI and GPIP3 is associated with TBI. When the edge bit is a one, the auxiliary input will be active high, enabling the timer while the input signal is at a high level. If the edge bit is low, the auxiliary input will be active low and the timer will operate while the input signal is at a low level.

The state of the active edge bit also specifies whether a zero-to-one transition or a one-to-zero transition of the auxiliary input pin will produce an interrupt when the interrupt channel is enabled. In normal operation, programming the active edge bit to a one will produce an interrupt on the zero-to-one transition of the associated input signal. Alternately, programming the edge bit to a zero will produce an interrupt on the one-to-zero transition of the input signal. However, in the pulse width measurement mode, the interrupt generated by a transition on TAI or TBI will occur on the opposite transition as that normally defined by the edge bit.

For example, in the pulse width measurement mode, if the edge bit is a one, the timer will be allowed to run while the auxiliary input TAI is high. When TAI transitions from high to low, the timer will stop and, if the interrupt channel is enabled, an interrupt will occur. By having the interrupt occur on the one-to-zero transition instead of the zero-to-one transition, the processor will be interrupted when the pulse being measured has terminated and the width of the pulse is available from the timer. Therefore, the timers act like a divide-by-prescaler that can be programmed by the timer data register and the timer's A and B control register.

After reading the contents of the timer, the main counter must be reinitialized by writing to the timer data register to allow consecutive pulses to be measured. If the timer is written after the auxiliary input signal is active, the timer will count from the previous contents of the timer data register until it counts through 01 (hexadecimal). At that time, the main counter is loaded with the new value from the timer data register, a time out pulse is generated which will toggle the timer output, and an interrupt may be optionally generated on the timer interrupt channel.

Note that the pulse width measured will include counts from before the main counter was reloaded. If the timer data register is written while the pulse is transitioning to the active state, an indeterminate value may be written into the main counter.

Once the timer is reprogrammed for another mode, interrupts will again occur as normally defined by the edge bit. Note that an interrupt may be generated as the result of placing the timer into the pulse width measurement mode or by reprogramming the timer for another mode. Also, an interrupt may be generated by changing the state of the edge bit while in the pulse width measurement mode.

5.1.3. EVENT COUNT MODE OPERATION. In addition to the delay mode and the pulse width measurement mode, timers A and B may be programmed to operate in the event count mode. Like the pulse width measurement mode, the event count mode also requires an auxiliary input signal, TAI or TBI, and the interrupt channels normally associated with I4 and I3 will respond to transitions on TAI and TBI, respectively. General purpose lines I3 and I4 still function normally.

In the event count mode the prescaler is disabled, allowing each active transition on TAI and TBI to produce a count pulse. The count pulse causes the main counter to decrement by one. When the timer counts through 01 (hexadecimal), a time out pulse is generated which will cause the output signal to toggle and may optionally produce an interrupt via the associated timer interrupt channel. The timer's main counter is also reloaded from the timer data register. To count transitions reliably, the input signal may only transition once every four timer clock periods. For this reason, the input signal must have a maximum frequency equal to one-fourth that of the timer clock.

The active edge of the auxiliary input signal is defined by the associated interrupt channel's edge bit. GPIP4 of the AER specifies the active edge for TAI and GPIP3 defines the active edge for TBI. When the edge bit is programmed to a one, a count pulse will be generated on the zero-to-one transition of the auxiliary input signal. When the edge bit is programmed to a zero, a count pulse will be generated on the one-to-zero transition. Also, note that changing the state of the edge bit while the timer is in the event count mode may produce a count pulse.

Besides generating a count pulse, the active transition of the auxiliary input signal will also produce an interrupt on the I3 or I4 interrupt channel, if the interrupt channel is enabled. Typically, in the event count mode, these channels are not enabled since the timer is automatically counting transitions on the input signal. If the interrupt channel were enabled, the number of transitions could be counted in the interrupt routine without requiring the use of the timer.

5.2. TIMER REGISTERS

The four timers are programmed via three control registers and four timer data registers. Control registers TACR and TBCR and timer data registers TADR and TBDR (refer to figure 5-1) are associated with timers A and B respectively. Timers C and D are controlled by the control register TCDCR and the data registers TCDR and TDDR (refer to figure 5.2).

Figure 5.2 : Timer Data Registers.

(a) Timer A Data Register (TADR).

	7	6	5	4	3	2	1	0
Address 1F (Hex)	D7	D6	D5	D4	D3	D2	D1	D0

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero

(b) Timer B Data Register (TBDR).

	7	6	5	4	3	2	1	0
Address 21 (Hex)	D7	D6	D5	D4	D3	D2	D1	D0

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero

Figure 5.2 : Timer Data Registers (continued).

(c) Timer C Data Register (TCDR).

	7	6	5	4	3	2	1	0
Address 23 (Hex)	D7	D6	D5	D4	D3	D2	D1	D0

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero

(d) Timer D Data Register (TDDR).

	7	6	5	4	3	2	1	0
Address 25 (Hex)	D7	D6	D5	D4	D3	D2	D1	D0

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero

5.2.1. TIMER DATA REGISTERS. Each timer's main counter is an 8-bit binary down counter. The value of the main counter may be read at any time by reading the timer's data register. The information read is the value of the counter which was captured on the last low-to-high transition of the $\overline{DS}$ pin.

The main counter is initialized by writing to the timer's data register. If the timer is stopped, data is loaded simultaneously into both the timer data register and the main counter. If the timer data register is written while the timer is enabled, the value is not loaded into the timer until the timer counts through 01 (hexadecimal). Writing the timer data register while the timer is counting through 01 (hexadecimal) will cause an indeterminate value to be loaded into the timer's main counter. The four data registers are shown in figure 5.2.

5.2.2. TIMER CONTROL REGISTERS. Bits in the timer control registers select the operation mode, select the prescaler value, and disable the timers. Timer control registers TACR and TBCR also have bits which allow the programmer to reset output lines TAO and TBO. These control registers are shown in figure 5.3.

Figure 5.3 : Timer Control Registers.

(a) Timer A Control Register (TACR).

	7	6	5	4	3	2	1	0
Address 19 (Hex)	*	*	*	Reset TA0	AC3	AC2	AC1	AC0

* Unused bits read as zero

(b) Timer B Control Register (TBCR).

	7	6	5	4	3	2	1	0
Address 1B (Hex)	*	*	*	Reset TB0	BC3	BC2	BC1	BC0

* Unused bits read as zero

Reset TAO/TBO Timer's A and B output lines (TAO and TBO) may be forced low at any time by writing a one to the reset location in TACR and TBCR, respectively. The output will be held low only during the write operation ; at the conclusion of the operation, the output will be allowed to toggle in response to a time-out pulse. When resetting TAO and TBO, the remaining bits in the control register must be written with their previous value to avoid altering the operation mode.

SET	a) End of write cycle which clears the bit
CLEARED	a) MPU writes a zero
	b) Reset

AC3-AC0, BC3-BC0 These bits are decoded to determine the timer operatio$ mode.

AC3 BC3	AC2 BC2	AC1 BC1	AC0 BC0	Operation Mode
0	0	0	0	Timer Stopped *
0	0	0	1	Delay Mode, ÷ 4 Prescaler
0	0	1	0	Delay Mode, ÷ 10 Prescaler
0	0	1	1	Delay Mode, ÷ 16 Prescaler
0	1	0	0	Delay Mode, ÷ 50 Prescaler
0	1	0	1	Delay Mode, ÷ 64 Prescaler
0	1	1	0	Delay Mode, ÷ 100 Prescaler
0	1	1	1	Delay Mode, ÷ 200 Prescaler
1	0	0	0	Event Count Mode
1	0	0	1	Pulse Width Mode, ÷ 4 Prescaler
1	0	1	0	Pulse Width Mode, ÷ 10 Prescaler
1	0	1	1	Pulse Width Mode, ÷ 16 Prescaler
1	1	0	0	Pulse Width Mode, ÷ 50 Prescaler
1	1	0	1	Pulse Width Mode, ÷ 64 Prescaler
1	1	1	0	Pulse Width Mode, ÷ 100 Prescaler
1	1	1	1	Pulse Width Mode, ÷ 200 Prescaler

* Regardless of the operation mode, counting is inhibited when the timer is stopped. The contents of the timer's main counter is not affected, although any residual count in the prescaler is lost.

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

Figure 5.3 : Timer Control Registers (continued).

(c) Timers C and D Control Register (TCDCR).

	7	6	5	4	3	2	1	0
Address 1D (Hex)	*	CC2	CC1	CC0	*	DC2	DC1	DC0

* Unused bits read as zero

CC2-CC0, DC2-DC0 The bits are decoded to determine the timer operation mode.

CC2 DC2	CC1 CC0	DC0 DC1	Operation Mode
0	0	0	Timer Stopped *
0	0	1	Delay Mode, ÷ 4 Prescaler
0	1	0	Delay Mode, ÷ 10 Prescaler
0	1	1	Delay Mode, ÷ 16 Prescaler
1	0	0	Delay Mode, ÷ 50 Prescaler
1	0	1	Delay Mode, ÷ 64 Prescaler
1	1	0	Delay Mode, ÷ 100 Prescaler
1	1	1	Delay Mode, ÷ 200 Prescaler

* When the timer is stopped, counting is inhibited. The contents of the timer's main counter is not affected, although any residual count in the prescaler is lost.

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

SECTION 6

UNIVERSAL SYNCHRONOUS/ASYNCHRONOUS RECEIVER-TRANSMITTER

The universal synchronous/asynchronous receiver-transmitter (USART) is a single full-duplex serial channel with a double-buffered receiver and transmitter. There are separate receive and transmit clocks and separate receive and transmit status and data bytes. The receive and transmit sections are also assigned separate interrupt channels. Each section has both a normal condition interrupt channel and an error condition interrupt channel. These channels can be optionally disabled from interrupting the processor and instead, DMA transfers can be performed using the receiver ready and transmitter ready external CMFP signals.

6.1. CHARACTER PROTOCOLS

The CMFP USART supports asynchronous and with the aid of a polynomial generator checker (PGC) supports byte synchronous character formats. These formats are selected independently of the divide-by-one and divide-by-16 clock modes.

When the divide-by-one clock mode is selected, synchronization must be accomplished externally. The receiver will sample the serial data on the rising edge of the receiver clock. In the divide-by-16 clock mode, the data is sampled at mid-bit time to increase transient noise rejection.

Also, when the divide-by-16 clock mode is selected, the USART resynchronization logic is enabled. This logic increases the channel's clock skew tolerance. When a valid transition is detected, an internal counter is reset to state zero. Transition checking is then inhibited until state four. Then at state eight, the previous state of the transition checking logic is clocked into the receive shift register.

6.1.1. ASYNCHRONOUS FORMAT. Variable word length and start/stop bit configurations are available under software control for asynchronous operation. The word length can be five to eight bits and one, one and one-half, or two stop bits can be selected. The user can also select odd, even, or no parity. For character lengths of less than eight bits, the assembled character will consist of the required number of data bits followed by zeros in the unused bit positions and a parity bit, if parity is enabled.

In the asynchronous format, start bit detection is always enabled. New data is not shifted into the receive shift register until a zero bit is received. When the divide-by-16 clock mode is selected, the false start bit logic is also active. Any transition must be stable for three positive receive clock edges to be considered valid. Then a valid zero-to-one transition must not occur for at least eight additional positive clock edges.

6.1.2. SYNCHRONOUS FORMAT. When the synchronous character format is selected, the 8-bit synchronous character loaded into the synchronous character register is compared to received serial data until a match is found. Once synchronization is established, incoming data is clocked into the receiver. The synchronous word will be continuously transmitted during an underrun condition. All synchronous characters can be optionally stripped from the receive buffer. Figure 6.1 shows the synchronous character register.

The synchronous character is typically written after the data word length is selected, since unused bits in the synchronous character register are zeroed out. When parity is enabled, synchronous word length is the data word length plus one. The CMFP will compute and append the parity bit for the synchronous word when a word length of eight is selected. However, if the word length is less than eight, the user must determine the synchronous word parity and write it into synchronous character. The CMFP will then transmit the extra bit in the synchronous word as a parity bit.

Figure 6.1 : Synchronous Character Register (SCR).

	7	6	5	4	3	2	1	0
Address 27 (Hex)	D7	D6	D5	D4	D3	D2	D1	D0

6.1.3. USART CONTROL REGISTER. The USART control register (UCR) selects the clock mode and the character format for the receive and transmit sections. This register is shown in figure 6-2.

6.2. RECEIVER

As data is received on the serial input line (SI), it is clocked into an internal 8-bit shift register until the specified number of data bits have been assembled. This character will then be transferred to the receive buffer, assuming that the last word in the receiver buffer has been read. This transfer produces a buffer full interrupt to the processor.

Reading the receive buffer satisfies the buffer full condition and allows a new data word to be transferred to the receive buffer when it is assembled. The receive buffer is accessed by reading the USART data register (UDR). The UDR is simply an 8-bit data register used when transferring data from the CMFP and the CPU.

Each time a word is transferred to the receive buffer, its status information is latched into the receiver status register (RSR). The RSR is not updated again until the data word in the receive buffer has been read. When a buffer full condition exists, the RSR should always be read before the receive buffer (UDR) to maintain the correct correspondance between data and flags. Otherwise, it is possible that after reading the UDR and prior to reading the RSR, a new word could be received and transferred to the receive buffer. Its associated flags would be latched into the RSR, over-writing the flags for the previous data word. Then when the RSR were read to access the status information for the first data word, the flags for the new word would be retrieved.

Figure 6.2 : USART Control Register (UCR).

	7	6	5	4	3	2	1	0
Address 29 (Hex)	CLK	WL1	WL0	ST1	ST0	PE	E/O	WU

CLK — Clock Mode. When this bit is zero, data will be clocked into and out of the receiver and transmitter at the frequency of their respective clocks. When this bit is a one, data will be clocked into and out of the receiver and transmitter at one sixteenth the frequency of their respective clocks. Also, the receiver data transition resynchronization logic will be enabled.

SET - ÷ 16	a) MPU writes a one
CLEARED - ÷ 1	a) MPU writes a zero
	b) Reset

WL0, WL1 — Word Length. These two bits specify the length of the data word exclusive of start bits, stop bits, and parity.

WL1	WL0	Word Length
0	0	8 Bits
0	1	7 Bits
1	0	6 Bits
1	1	5 Bits

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

ST0, ST1 — Start/Stop Bit and Format Control. These two bits select the number of start and stop bits and also specify the character format.

ST1	ST0	Start Bits	Stop Bits	Format
0	0	0	0	Synchronous
0	1	1	1	Asynchronous
1	0	1	1 1/2	Asynchronous*
1	1	1	2	Asynchronous

* Only used with divide-by-16 clock mode

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

Figure 6.2 : USART Control Register (UCR) (continued).

PE — Parity Enable. When this bit is zero, no parity check will be made and no parity bit will be computed for transmission. When this bit is a one, parity will be checked by the receiver and parity will be calculated and inserted during data transmission. Note that parity is not automatically appended to the synchronous character for word lengths of less than eight bits. In this case, the parity should be written into the synchronous character register along with the synchronous word.

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

E/O — Even/Odd Parity. When this bit is zero, odd parity is selected. When this bit is a one, even parity is selected.

SET	a) MPU writes a one
CLEARED	a) MPU writes a zero
	b) Reset

WU — Bit 0 Reserved. Must be mantained at 0.

6.2.1. RECEIVER INTERRUPT CHANNELS. The USART receive section is assigned two interrupt channels. One indicates the buffer full condition, while the other channel indicates an error condition. Error conditions include overrun, parity error, synchronous found, and break. These interrupting conditions correspond to the BF, OE, PE, and F/S or B bits of the receiver status register. These flags will function as described in 6.2.2. whether the receiver interrupt channels are enabled or disabled.

While only one interrupt is generated per character received, two dedicated interrupt channels allow separate vector numbers to be assigned for normal and abnormal receiver conditions. When a received word has an error associated with it and the error interrupt channel is enabled, an interrupt will be generated on the error channel only. However, if the error channel is disabled, an interrupt for an error condition will be generated on the buffer full interrupt channel along with interrupts produced by the buffer full condition. The receiver status register must always be read to determine which error condition produced the interrupt.

6.2.2. RECEIVER STATUS REGISTER. The receiver status register contains the receive buffer full flag, the synchronous strip enable, the receiver enable, and various status information associated with the data word in the receive buffer. The RSR is latched each time a data word is transferred to the receive buffer. RSR flags cannot change again until the data word has been read. The exception is the character in progress flag which monitors when a new word is being assembled in the asynchronous character format. The receiver status register is shown in figure 6.3.

Figure 6.3 : Receiver Status Register (RSR).

Address 2B	BF	OE	PE	FE	F/S or B	M/CIP	SS	RE

BF — Buffer Full. This bit is set when a received word is transferred to the receive buffer. This bit is cleared when the receive buffer is read by accessing the USART data register (UDR). This bit is read only.

SET	a) Received word transferred to buffer
CLEARED	a) Receive buffer read
	b) Reset

OE — Overrun Error. An overrun error occurs when a received word is due to be transferred to the receive buffer, but the receive buffer is full. Neither the receive buffer nor the RSR is overwritten. The OE bit is set after the receive buffer full condition is satisfied by reading the UDR. This error condition will generate an interrupt to the processor. The OE bit is cleared by reading the RSR. New data words will not be assembled until the RSR is read.

SET	a) Incoming word received and receive buffer full
CLEARED	a) Receiver status register read
	b) Reset

PE — Parity Error. This bit is set when the word transferred to the receive buffer has a parity error. This bit is cleared when the word transferred to the receive buffer does not have a parity error.

SET	a) Word in receive buffer has a parity error
CLEARED	a) Word in receive buffer does not have a parity error
	b) Reset

Figure 6.3 : Receiver Status Register (RSR) (continued).

FE	Frame Error. A frame error exists when a non-zero data word is not followed by a stop bit in the asynchronous character format. The FE bit is set when the word transferred to the receive buffer has a frame error. The FE bit is cleared when the word transferred to the receive buffer does not have a frame error.		
	SET	a)	Word in receive buffer has a frame error
	CLEARED	a)	Word in receive buffer does not have a frame error
		b)	Reset
F/S or B	Found/Search or Break Detect. In the synchronous character format this bit can be set or cleared in software. When the bit is a zero, the USART receiver is placed in the search mode. The incoming data is compared to the synchronous character register (SCR) and the word length counter is disabled. The F/S bit will automatically be set when a match is found and the word length counter will be enabled. An interrupt will also be produced on the receive error channel.		
	SET	a)	Incoming word matches synchronous character
	CLEARED	a)	MPU writes a zero
		b)	Incoming word does not match synchronous character
		c)	Reset
	In the asynchronous character format, this flag indicates a break condition. A break is detected when an all zero data word with no stop bit is received. The break condition continues until a non-zero data bit is received. The 8-bit is set when the word transferred to the receive buffer is a break indication. A break condition generates an interrupt to the processor. This bit is cleared when a non-zero data bit is received and the break condition has been acknowledged by reading the RSR at least once An end of break interrupt will be generated when the bit is cleared.		
	SET	a)	Word in receive buffer is a break
	CLEARED	a)	Break terminates and receiver status register read since beginning of break condition
		b)	Reset
M or CIP	Match/Character in Progress In the synchronous format, this flag indicates that a synchronous character has been received. The M bit is set when the word transferred to the receive buffer matches the synchronous character register. The M bit is cleared when the word transferred to the receive buffer does not match the synchronous character register.		
	SET	a)	Word transferred to receive buffer matches the synchronous character
	CLEARED	a)	Word transferred to receive buffer does not match synchronous character
		b)	Reset
	In the asynchronous character format, this flag indicates that a word is being assembled. The CIP bit is set when a start bit is detected. The CIP bit is cleared when the final stop bit has been received.		
	SET	a)	Start bit is detected
	CLEARED	a)	End of word detected
		b)	Reset
SS	Synchronous Strip Enable. When this bit is a one, data words that match the synchronous character register will not be loaded into the receive buffer and no buffer full condition will be produced. When this bit is a zero, data words that match the synchronous character register will be transferred to the receive buffer and a buffer full condition will be produced.		
	SET	a)	MPU writes a one
	CLEARED	a)	MPU writes a zero
		b)	Reset
RE	Receiver Enable. When this bit is a zero, the receiver will be immediately disabled. All flags will be cleared. When this bit is a one, normal receiver operation is enabled. This bit should not be set to a one until the receiver clock is active.		
	SET	a)	MPU writes a one
		b)	Transmitter is disabled in auto-turnaround mode
	CLEARED	a)	MPU writes a zero
		b)	Reset

6.2.3. SPECIAL RECEIVE CONSIDERATIONS. Certain receive conditions relating to the overrun error flag and the break detect flag require further explanation. Consider the following examples :

1) A break is received while the receive buffer is full. This does not produce an overrun condition. Only the B flag will be set after the receiver buffer is read.

2) A new word is received and the receive buffer is full. A break is received before the receive buffer is read.
   Both the B and OE flags will be set when the buffer full condition is satisfied.

6.3. TRANSMITTER

The transmit buffer is loaded by writing to the USART data register (UDR). The data word will be transferred to an internal 8-bit shift register when the last word in the shift register has been transmitted. This will produce a buffer empty condition. If the transmitter completes the transmission of word in the shift register before a new word is written to the transmit buffer, an underrun error will occur. In the asynchronous character format, the transmitter will send a mark until the transmit buffer is written. In the synchronous character format, the transmitter will continuously send the synchronous character.

The transmit buffer can be loaded prior to enabling the transmitter. After the transmitter is enabled, there is a delay before the first bit is output. The serial output line (SO) should be programmed to be high, low, or high impedance when the transmitter is enabled to force the output line to the desired state until the first bit is shifted out. Note that a one bit will always be transmitted prior to the word in the transmit shift register when the transmitter is first enabled.

When the transmitter is disabled, any word currently being transmitted will continue to completion. However, any word in the transmit buffer will not be transmitted and will remain in the buffer. So, no buffer empty condition will occur. If the buffer is empty when the transmitter is disabled, the buffer empty condition will remain, but no underrun condition will be generated when the word in transmission is completed. If no word is being transmitted when the transmitter is disabled, the transmitter will stop at the next rising edge of the internal shift clock.

In the asynchronous character format, the transmitter can be programmed to send a break. The break will be transmitted once the word currently in the shift register has been sent. If the shift register is empty, the break command will be effective immediately. An END interrupt will be generated at every normal character boundary to aid in timing the break transmission. The break will continue until the break command is cleared.

Any character in the transmit buffer at the start of a break will be transmitted when the break is terminated. If the transmit buffer is empty at the start of a break, it may be written at any time during the break. If the buffer is still empty at the end of the break, an underrun condition will exist.

Disabling the transmitter during a break condition causes the transmitter to cease transmission of the break character at the end of the current character. No end of break stop bit will be transmitted. Even if the transmit buffer is empty, no buffer empty condition will occur nor will an underrun condition occur. Also, any word in the transmit buffer will remain.

6.3.1. TRANSMITTER INTERRUPT CHANNELS.

The USART transmit section is assigned two interrupt channels. One channel indicates a buffer empty condition and the other channel indicates an underrun or end condition. These interrupting condition correspond to the BE, UE, and END flag bits of the transmitter status register (TSR). The flag bits will function as described in 6.3.2 whether their associated interrupt channel is enabled or disabled.

6.3.2. TRANSMITTER STATUS REGISTER.

The transmitter status register contains various transmitter error flags and transmitter control bits for selecting auto-turnaroundand loopback mode. The TSR is shown in figure 6.4.

Figure 6.4 : Transmitter Status Register (TSR).

Address 2D	BE	UE	AT	END	B	H	L	TE

BE	Buffer Empty. This bit is set when the word in the transmit buffer is transferred to the transmit shift register. This bit is cleared when the transmit buffer is reloaded by writing to the USART data register (UDR). SET a) Transmit buffer contents transferred to transmit shift register CLEARED a) Transmit buffer written
UE	Underrun Error. This bit is set when the word in the transmit shift register has been transmitted before a new word is loaded into the transmit buffer. This bit is cleared by reading the TSR or by disabling the transmitter. This bit does not need to be cleared before writing to the UDR. SET a) Transmit shift register contents transmitted before transmit buffer written CLEARED a) Transmitter status register read b) Transmitter disabled
AT	Auto-Turnaround. When this bit is set, the receiver will be enabled automatically after the transmitter has been disabled and the last character being transmitted is completed. SET a) MPU writes a one CLEARED a) Transmitter disabled
END	End of Transmission. When the transmitter is disabled while a character is being transmitted, the END will be set after the character transmission is complete. If no word is being transmitted when the transmitter is disabled, the END bit will be set immediately. The END bit is cleared by reenabling the transmitter. SET a) Transmitter disabled CLEARED a) Transmitter enabled
B	Break. This bit has no function in the synchronous character format. In the asynchronous character format, when this bit is set to a one, a break will be transmitted upon the completion of the transmission of any word in the transmit shift register. A break consists of an all zero data word with no stop bit. When this bit is cleared by software, the break indication will cease and normal transmission will resume. Note that when B is set, BE cannot be set. SET a) MPU writes a one CLEARED a) MPU writes a zero
H, L	High and Low. These control bits configure the transmitter output (SO) when the transmitter is disabled. These bits also force the transmitter output after the transmitter is enabled until END is cleared.

H	L	Output State
0	0	High Impedance
0	1	Low
1	0	High
1	1	Loopback Mode

Loopback mode internally connects the transmitter output to the receiver input and the transmitter clock to the receiver clock internally. The receiver clock (RC) and the serial input (SI) are not used. When the transmitter is disabled, SO is forced high.
SET a) MPU writes a one
CLEARED a) MPU writes a zero

TE	Transmitter Enable. When this bit is cleared, the transmitter is disabled. The UE bit will be cleared and the END bit will be set. When this bit is set, the transmitter is enabled. The transmitter output will be driven according to the H and L bits until transmission begins. A one bit will be transmitted before the transmission of the word in the transmit shift register is begun. SET a) MPU writes a one CLEARED a) MPU writes a zero b) Reset

6.4. DMA OPERATION

USART error conditions are only valid for each character boundary. When the USART performs block data transfers by using the DMA handshake lines RR (receiver ready) and TR (transmitter ready), errors must be saved and checked at the end of a block. This is accomplished by enabling the error channel for the receiver or transmitter and by masking interrupts for this channel. Once the transfer is complete, interrupt pending register A is read. Any pending receiver or transmitter error indicates an error in the data transfer.

SECTION 7

ELECTRICAL CHARACTERISTICS

This section contains the electrical specifications and associated timing information for the TS68HC901 multi-function peripheral.

7.1. MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	– 0.3 to + 7.0	V
V_{IN}	Input Voltage	– 0.3 to + 7.0	V
T_A	Operating Temperature Range TS68HC901C TS68HC901V	T_L to T_H 0 to + 70 – 40 to + 85	°C
T_{stg}	Storage Temperature	– 65 to + 150	°C

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_{CC} or GND).

7.2. THERMAL DATA

θ_{JA}	Thermal Resistance Plastic	50	°C/W

7.3. POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from :

$$T_J = T_A + (P_D \bullet \theta_{JA}) \qquad (1)$$

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 :

$$P_D = K + (T_J + 273°C) \qquad (2)$$

Solving equations 1 and 2 for K gives :

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} P_D^2 \qquad (3)$$

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.

7.4. DC ELECTRICAL CHARACTERISTICS
($T_A = T_L$ to T_H $V_{CC} = +5V \pm 5\%$, unless otherwise noted)

Symbol	Parameter	Min.	Max.	Unit
V_{IH}	Input HIgh Voltage Except XTAL1, XTAL2	2.0	$V_{DD} + 0.3$	V
V_{IH}	Input High Voltage XTAL1, XTAL2	$V_{DD} - 1.5$	$V_{DD} + 0.3$	V
V_{IL}	Input Low Voltage	– 0.3	0.8	V
V_{OH}	Output High Voltage, Except $\overline{DTACK}$ (I_{OH} = - 120 µA)	4.1		V
V_{OL}	Output Low Voltage, Except $\overline{DTACK}$ (I_{OL} = 2.0 mA)		0.5	V
I_{CC}	Power Supply Current (outputs open)		6	mA
I_{LI}	Input Leakage Current (V_{in} = 0 to V_{CC})		± 10	µA
I_{LOH}	Hi-Z Output Leakage Current in Float (V_{out} = 2.4 to V_{CC})		10	µA
I_{LOL}	Hi-Z Output Leakage Current in Float (V_{out} = 0.5 V)		– 10	µA
I_{OH}	$\overline{DTACK}$ Output Source Current (V_{out} = 2.4 V)		– 400	µA
I_{OL}	$\overline{DTACK}$ Output Sink Current (V_{out} = 0.5 V)		5.3	mA
P_D	Power Dissipation		32	mW

7.5. CAPACITANCE (T_A = 25°C, f = 1MHz, unmeasured pins returned to ground)

Symbol	Parameter	Min.	Max.	Unit
C_{IN}	Input Capacitance		10	pF
C_{OUT}	Hi-Z Output Capacitance		10	pF

Figure 7.1 : $\overline{IRQ}$ Test Load.

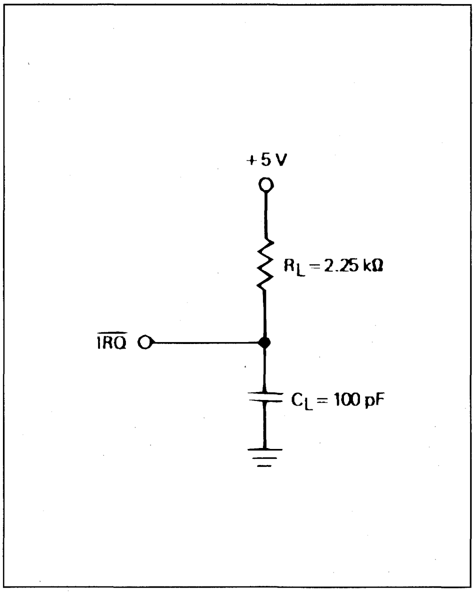
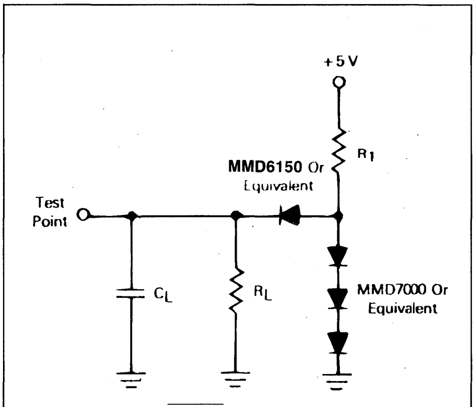


Figure 7.2 : Typical Test Load.



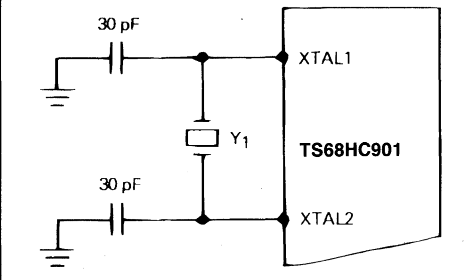
for all outputs except $\overline{DTACK}$
C_L = 100pF
R_L = 20kΩ
R_1 = 1.90kΩ

for $\overline{DTACK}$
C_L = 130pF
R_L = 6kΩ
R_1 = 740Ω

7.6. CLOCK TIMING

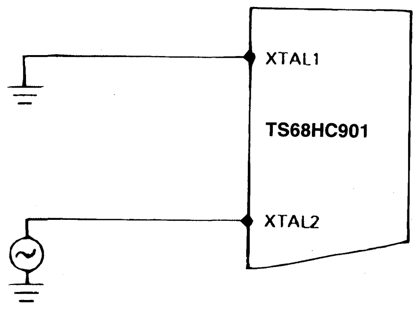
Symbol	Parameter	4 MHz		5 MHz		8 MHz		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
f	Frequency of Operation	1.0	4.0	1.0	5.0	1.0	8.0	MHz
t_{cyc}	Cycle Time	250	1000	200	1000	125	1000	ns
t_{CL}, t_{CH}	Clock Pulse Width	110	480	90	480	55	480	ns
t_{Cr}, t_{Cf}	Rise and Fall Times	–	15	–	10	–	10	ns

Figure 7.3 : CMFP External Oscillator Components.



Crystal Parameters
Parallel resonance fundamental mode AT cut
$R_S \leq 150\Omega$ (f = 2.8 - 4.0MHz)
$R_S \leq 300\Omega$ (f = 2.0 - 2.7MHz)
C_L = 18pF, C_M = 0.02pF, C_R = 5pF, L_M = 96MHz
f (typical) = 2.4576MHz

Figure 7.3.1 : CMFP External Clock Connection.



Other possible configuration :
XTAL1 driven with a CMOS clock and XTAL2 not connected.

7.7. AC ELECTRICAL CHARACTERISTICS T_{amb} = 0°C to 70°C, V_{CC} = 5.0 V_{DC} ± 5%, V_{SS} = 0 V_{DC} (unless otherwise specified). See figures 7-4 through 7-10.

N°	Parameter	4 MHz		5 MHz		8 MHz		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
1	$\overline{CS}$, $\overline{DS}$ Width High	50		35		25		
2	R/$\overline{W}$, A1-A5 Valid to Falling $\overline{CS}$ (setup)	30		25		20		ns
3	Data Valid Prior to Rising $\overline{DS}$	280		150		100		ns
4(3)	$\overline{CS}$, $\overline{IACK}$ Valid to Falling Clock (setup)	50		50		50		ns
4a (4)	Falling Clock to Next $\overline{CS}$ Low	100		80		50		ns
5	CLK Low to $\overline{DTACK}$ Low		220		180		90	ns
6	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to $\overline{DTACK}$ High		60		55		50	ns
7	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to $\overline{DTACK}$ Tri-state		100		100		100	ns
8	$\overline{DTACK}$ Low to Data Invalid (hold time)	0		0		0		ns
9	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to Data Tri-state		50		50		50	ns
10	$\overline{CS}$ or $\overline{DS}$ High to R/$\overline{W}$, A1-A5 Invalid (hold time)	0		0		0		ns

7.7. AC ELECTRICAL CHARACTERISTICS (continued)

N°.	Parameter	4 MHz		5 MHz		8 MHz		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
11[3,5]	Data Valid from $\overline{CS}$ Low		310		260		200	ns
12	Read Data Valid to $\overline{DTACK}$ Low (setup time)	50		50		20		ns
13	$\overline{DTACK}$ Low to $\overline{DS}$, $\overline{CS}$ or $\overline{IACK}$ High (hold time)	0		0		30		ns
14	$\overline{IEI}$ Low to Falling $\overline{CLK}$ (setup)	50		50		50		ns
15[1]	$\overline{IEO}$ Valid from Clock Low (delay)		180		180		120	ns
16	Data Valid from Clock Low (delay)		300		300		180	ns
17	$\overline{IEO}$ Invalid from $\overline{IACK}$ High (delay)		150		150		100	ns
18	$\overline{DTACK}$ Low from Clock High (delay)		180		165		100	ns
19[1]	$\overline{IEO}$ Valid from $\overline{IEI}$ Low (delay)		100		100		100	ns
20	Data Valid from $\overline{IEI}$ Low (delay)		220		220		195	ns
21	Clock Cycle Time	250	1000	200	1000	125	1000	ns
22	Clock Width Low	110		90		55		ns
23	Clock Width High	110		90		55		ns
24[4]	$\overline{DS}$ Inactive to Rising Clock (setup)	100		80		50		ns
25	I/O Minimun Active Pulse Width	100		100		100		ns
26	$\overline{IACK}$ Width High/Minimun Delay between two Pulses	2		2		2		CLK
27	I/O Data Valid from Rising $\overline{CS}$ or $\overline{DS}$		450		450		350	ns
28	Receiver Ready Delay from Falling RC		600		600		200	ns
29	Transmitter Ready Delay from Falling TC		600		600		200	ns
30[6]	Timer Ouput Low from Rising Edge of $\overline{CS}$ or $\overline{DS}$ (A & B) (reset T_{OUT})		450		450		200	ns
31[2]	T_{OUT} Valid from Internal Timeout		2 t_{CLK} + 300		2 t_{CLK} + 300		2 t_{CLK} + 300	ns
32	Timer Clock Low Time	110		90		55		ns
33	Timer Clock High Time	110		90		55		ns
34	Timer Clock Cycle Time	250	1000	200	1000	125	1000	ns
35	$\overline{RESET}$ Low Time	2		1.8		1.5		µs
36	Delay to Falling $\overline{INTR}$ from External Interrupt Active Transition		380		380		250	ns
37	Transmitter Internal Interrupt Delay from Falling Edge of TC		550		550		350	ns
39	Receiver Buffer Full Interrupt Transition Delay from Rising Edge of RC		800		800		400	ns
38	Receiver Error Interrupt Transition Delay from Falling Edge of RC		800		800		400	ns

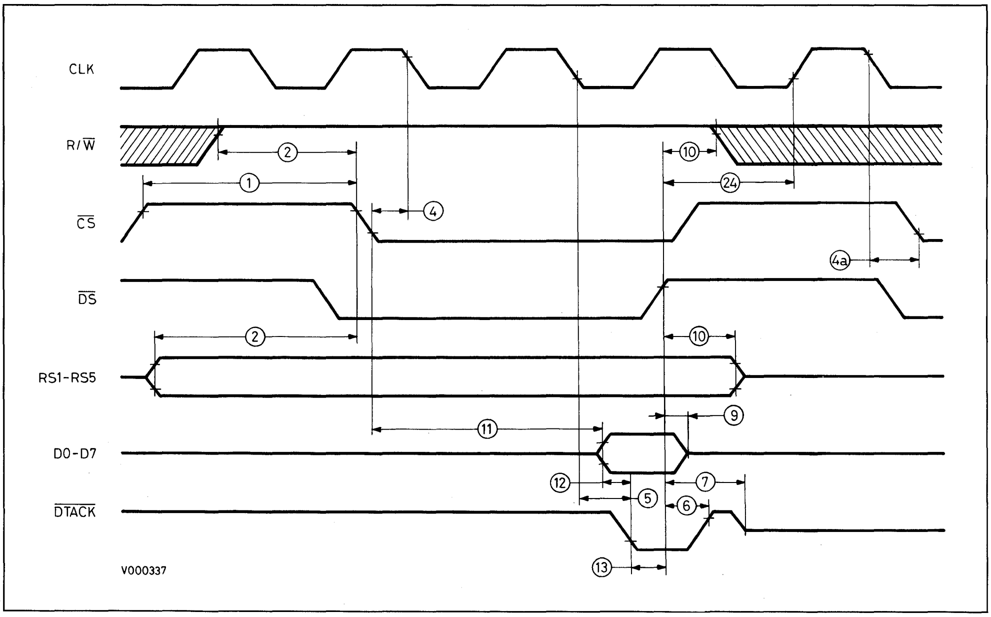
Notes :
1. $\overline{IEO}$ only goes low if no acknowledgeable interrupt is pending. If $\overline{IEO}$ goes low, $\overline{DTACK}$ and the data bus remain tri-stated.
2. T_{CLK} refers to the clock applied to the CMFP CLK input pin. t_{CLK} refers to the timer clock signal regardless of whether that signal comes from the XTAL1/XTAL2 crystal clock inputs or the TAI or TBI timer inputs.
3. If the setup time is not met, $\overline{CS}$ or $\overline{IACK}$ will not be recognized until the next falling CLK.
4. If the setup time is met (for consecutive cycles), the minimum hold-off time of one clock cycle will be obtained. If not met, the hold-off will be two clock cycles.
5. Althrough $\overline{CS}$ and $\overline{DTACK}$ are synchronized with the clock, the data out during a read cycle is asynchronous to the clock, relying only on $\overline{CS}$ for timing.
6. Spec. 30 applies to timer outputs TAO and TBO only.

7.7. AC ELECTRICAL CHARACTERISTICS (continued)

N°.	Parameter	4 MHz		5 MHz		8 MHz		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
40	Serial in Set Up Time to Rising Edge of RC (divide by one only)	80		70		50		ns
41	Data Hold Time from Rising Edge of RC (divide by one only)	350		325		100		ns
42	Serial Output Data Valid from Falling Edge of TC (÷ 1)		440		420		200	ns
43	Transmitter Clock Low Time	500		450		250		ns
44	Transmitter Clock High Time	500		450		250		ns
45	Transmitter Clock Cycle Time	1.05		0.95		0.55		µs
46	Receiver Clock Low Time	500		450		250		ns
47	Receiver Clock High Time	500		450		250		ns
48	Receiver Clock Cycle Time	1.05		0.95		0.55		µs
49(2)	$\overline{CS}$, $\overline{IACK}$, $\overline{DS}$ Width Low		80		80		80	T_{CLK}
50	Serial Output Data Valid from Falling Edge of TC (÷ 16)		490		370		250	ns

Note : 2. T_{CLK} refers to the clock applied to the CMFP CLK input pin. t_{CLK} refers to the timer clock signal regardless of whether that signal comes from the XTAL1/XTAL2 crystal clock inputs or the TAI or TBI timer inputs.

Figure 7.4 : Read Cycle Timing.



7.7.1. AC ELECTRICAL CHARACTERISTICS - READ CYCLES
(V_{CC} = 5.0 V_{DC} ± 5%, V_{SS} = 0 V_{DC}, T_A = T_L to T_H unless otherwise noted)

N°.	Parameter	4 MHZ		5 MHZ		8 MHz		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
1	$\overline{CS}$, $\overline{DS}$ Width High	50		35		25		
2	R/$\overline{W}$, A1-A5 Valid to Falling $\overline{CS}$ (setup)	30		25		20		ns
4(3)	$\overline{CS}$, $\overline{IACK}$ Valid to Falling Clock (setup)	50		50		50		ns
4a(4)	Falling Clock to Next $\overline{CS}$ Low	100		80		50		ns
5	CLK Low to $\overline{DTACK}$ Low		220		180		90	ns
6	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to $\overline{DTACK}$ High		60		55		50	ns
7	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to $\overline{DTACK}$ Tri-state		100		100		100	ns
9	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to Data Tri-state		50		50		50	ns
10	$\overline{CS}$ or $\overline{DS}$ High to R/$\overline{W}$, A1-A5 Invalid (hold time)	0		0		0		ns
11(3,5)	Data Valid from $\overline{CS}$ Low		310		260		200	ns
12	Read Data Valid to $\overline{DTACK}$ Low (setup time)	50		50		20		ns
13	$\overline{DTACK}$ Low to $\overline{DS}$, $\overline{CS}$ or $\overline{IACK}$ High (hold time)	0		0		0		ns
24(4)	$\overline{DS}$ Inactive to Rising Clock (setup)	100		80		50		ns

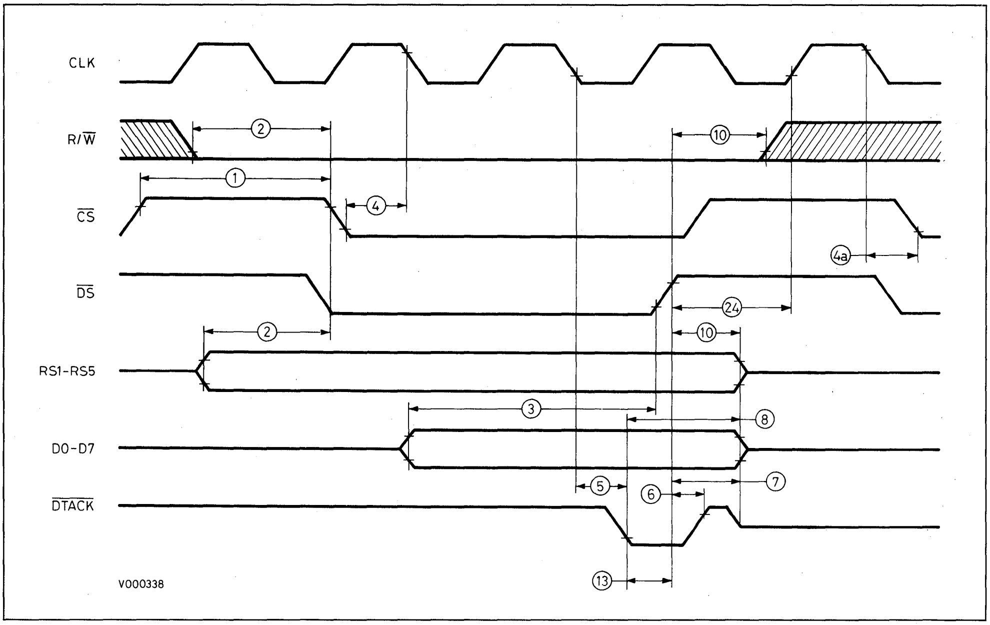
Notes : 3. If the setup time is not met, $\overline{CS}$ or $\overline{IACK}$ will not be recognized until the next falling CLK.
4. If the setup time is met (for consecutive cycles), the minimum hold-off time of one clock cycle will be obtained. If not met, the hold-off will be two clock cycles.
5. Althrough CS and DTACK are synchronized with the clock, the data out during a read cycle is asynchronous to the clock, relying only on CS for timing.

7.7.2. AC ELECTRICAL CHARACTERISTICS - WRITE CYCLES
(V_{CC} = 5.0 V_{DC} ± 5%, V_{SS} = 0 V_{DC}, T_A = T_L to T_H unless otherwise noted)

N°.	Parameter	4 MHZ		5 MHZ		8 MHz		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
1	$\overline{CS}$, $\overline{DS}$ Width High	50		35		25		
2	R/$\overline{W}$, A1-A5 Valid to Falling $\overline{CS}$ (setup)	30		25		20		ns
3	Data Valid Prior to Rising $\overline{DS}$	280		150		100		ns
4(3)	$\overline{CS}$, $\overline{IACK}$ Valid to Falling Clock (setup)	50		50		50		ns
4a(4)	Falling Clock to Next $\overline{CS}$ Low	100		80		50		ns
5	CLK Low to $\overline{DTACK}$ Low		220		180		90	ns
6	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to $\overline{DTACK}$ High		60		55		50	ns
7	$\overline{CS}$, $\overline{DS}$ or $\overline{IACK}$ High to $\overline{DTACK}$ Tri-state		100		100		100	ns
8	$\overline{DTACK}$ Low to Data Invalid (hold time)	0		0		0		ns
10	$\overline{CS}$ or $\overline{DS}$ High to R/$\overline{W}$, A1-A5 Invalid (hold time)	0		0		0		ns
13	$\overline{DTACK}$ Low to $\overline{DS}$, $\overline{CS}$ or $\overline{IACK}$ High (hold time)	0		0		0		ns
24(4)	$\overline{DS}$ Inactive to Rising Clock (setup)	100		80		50		n

Notes : 3. If the setup time is not met, $\overline{CS}$ or $\overline{IACK}$ will not be recognized until the next falling CLK.
4. If the setup time is met (for consecutive cycles), the minimum hold-off time of one clock cycle will be obtained. If not met, the hold-off will be two clock cycles.

Figure 7.5 : Write Cycle Timing.



7.7.3. AC ELECTRICAL CHARACTERISTICS - INTERRUPT ACKNOWLEDGE CYCLES
(V_{CC} = 5.0 V_{DC} ± 5%, V_{SS} = 0 V_{DC}, T_A = T_L to T_H - C unless otherwise noted)
See Figures 7.6 and 7.7.

N°.	Parameter	4 MHz		5 MHz		8 MHz		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
4(3)	CS, IACK Valid to Falling Clock (setup)	50		50		50		ns
5	CLK Low to DTACK Low		220		180		90	ns
6	CS, DS or IACK High to DTACK High		60		55		50	ns
7	CS, DS or IACK High to DTACK Tri-state		100		100		100	ns
9	CS, DS or IACK High to Data Tri-state		50		50		50	ns
13	DTACK Low to DS, CS or IACK High (hold time)	0		0		0		ns
14	IEI Low to Falling CLK (setup)	50		50		50		ns
15(1)	IEO Valid from Clock Low (delay)		180		180		120	ns
16	Data Valid from Clock Low (delay)		300		300		180	ns
17	IEO Invalid from IACK High (delay)		150		150		100	ns
18	DTACK Low from Clock High (delay)		180		165		100	ns
19(1)	IEO Valid from IEI Low (delay)		100		100		100	ns
20	Data Valid from IEI Low (delay)		220		200		195	ns
21	Clock Cycle Time	250	1000	200	1000	125	1000	ns
22	Clock Width Low	110		90		55		ns
23	Clock Width High	110		90		55		ns
24(4)	DS, Inactive to Rising Clock (setup)	100		80		50		ns
25	I/O Minimum Active Pulse Width	100		100		100		ns
26	IACK Width High/minimun Delay between two Pulses	2		2		2		CLK

Notes : 1. IEO only goes low if no acknowledgeable interrupt is pending. If IEO goes low, DTACK and the data bus remain tri-stated.
3. If the setup time is not met, CS or IACK will not be recognized until the next falling CLK.
4. If the setup time is met (for consecutive cycles), the minimum hold-off time of one clock cycle will be obtained. If not met, the hold-off will be two clock cycles.

Figure 7.6 : Interrupt Acknowledge Cycle ($\overline{\text{IEI}}$ Low).

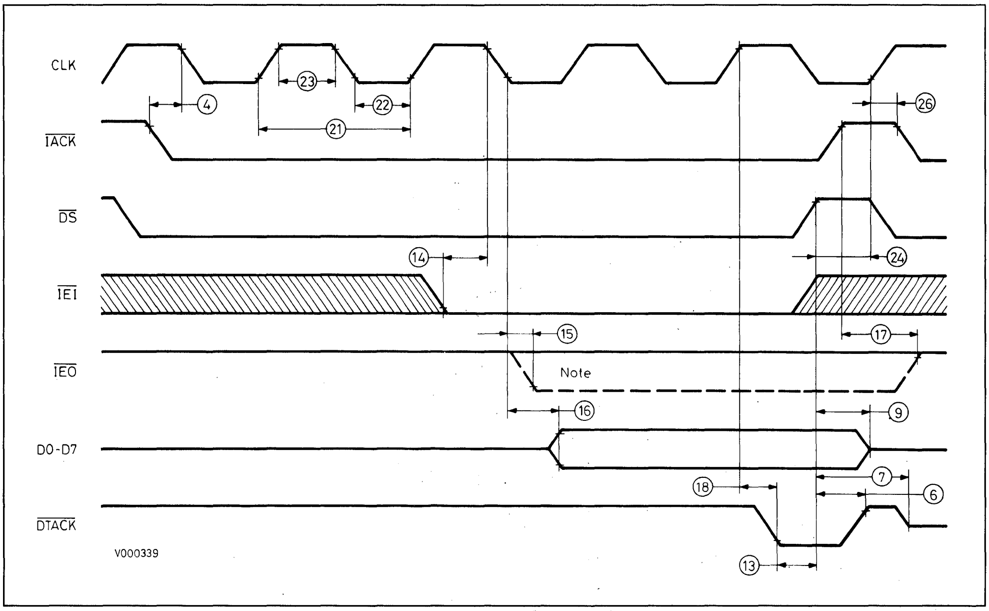
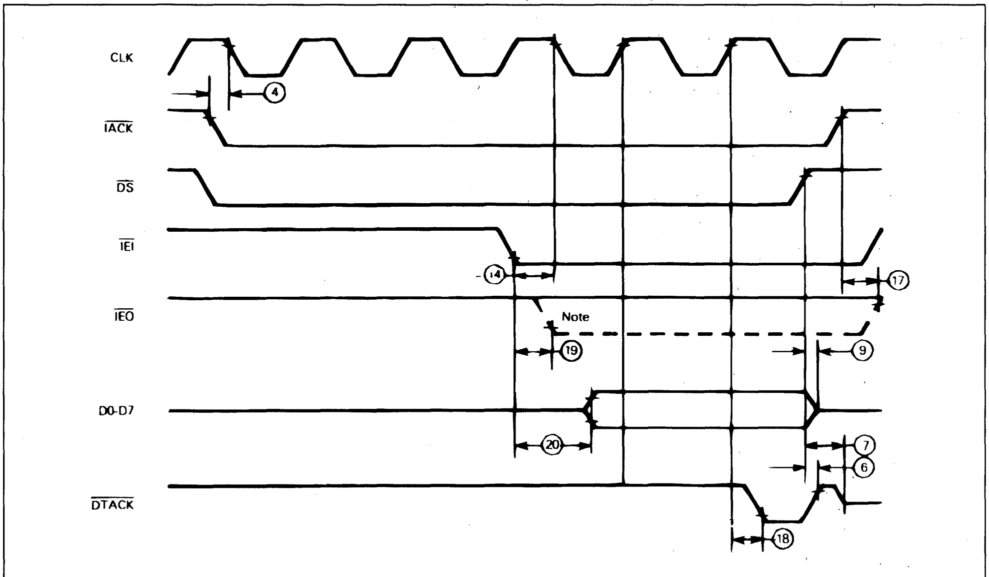


Figure 7.7 : Interrupt Acknowledge Cycle ($\overline{\text{IEI}}$ High)

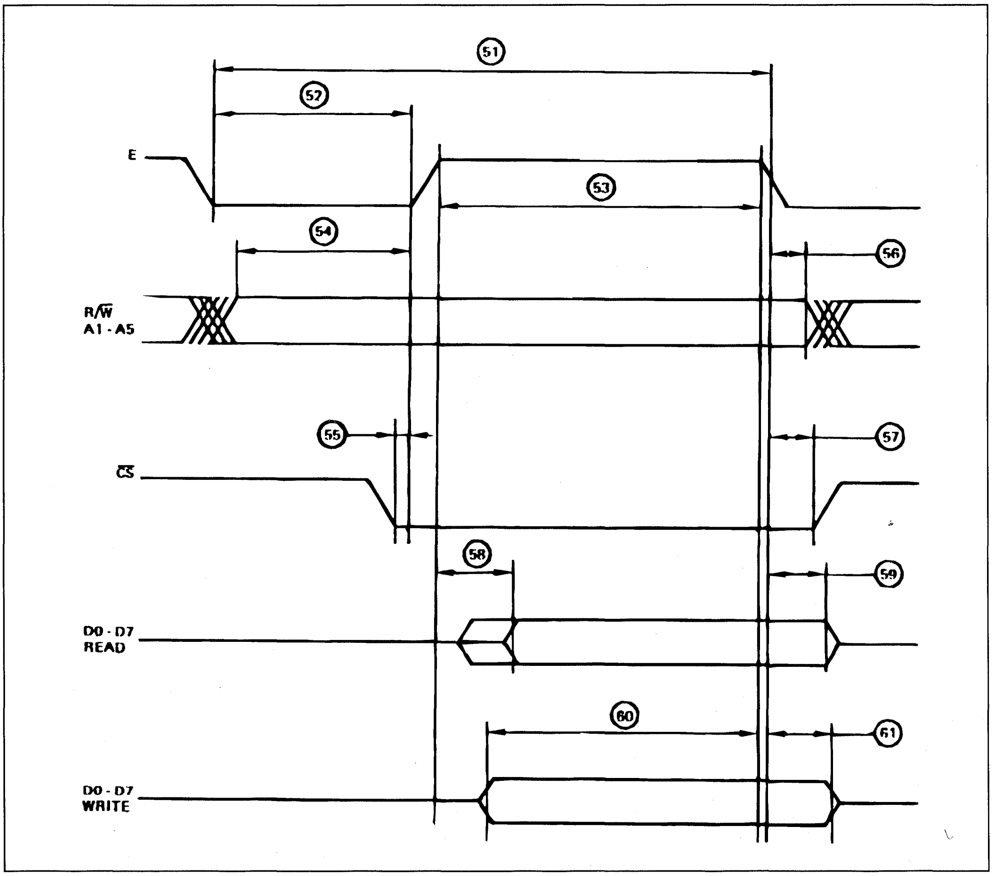


Note : $\overline{\text{CS}}$ and $\overline{\text{IACK}}$ must be a function of $\overline{\text{DS}}$.

7.7.4. AC ELECTRICAL CHARACTERISTICS - 6800 INTERFACE TIMING (V_{CC} = 5.0 V_{DC} ± 5%, V_{SS} = 0 V_{DC}, T_A = 0˚C to 70˚C unless otherwise noted). See figure 7.8.

N°.	Parameter	Min.	Max.	Unit
51	Cycle Time	1000		ns
52	Pulse Width, E High	430		ns
53	Pulse Width, E Low	450		ns
54	Address, R/$\overline{W}$ Setup Time Before E	80		ns
55	$\overline{CS}$ Setup Time Before E	80		ns
56	Address Hold Time	10		ns
57	$\overline{CS}$ Hold Time	10		ns
58	Output Data Delay Time (read)		250	ns
59	Data Hold Time (read)	0	100	ns
60	Input Data Setup Time (write)	280		ns
61	Data Hold Time (write)	20		ns

Figure 7.8 : 6800 Interfacing Timing.

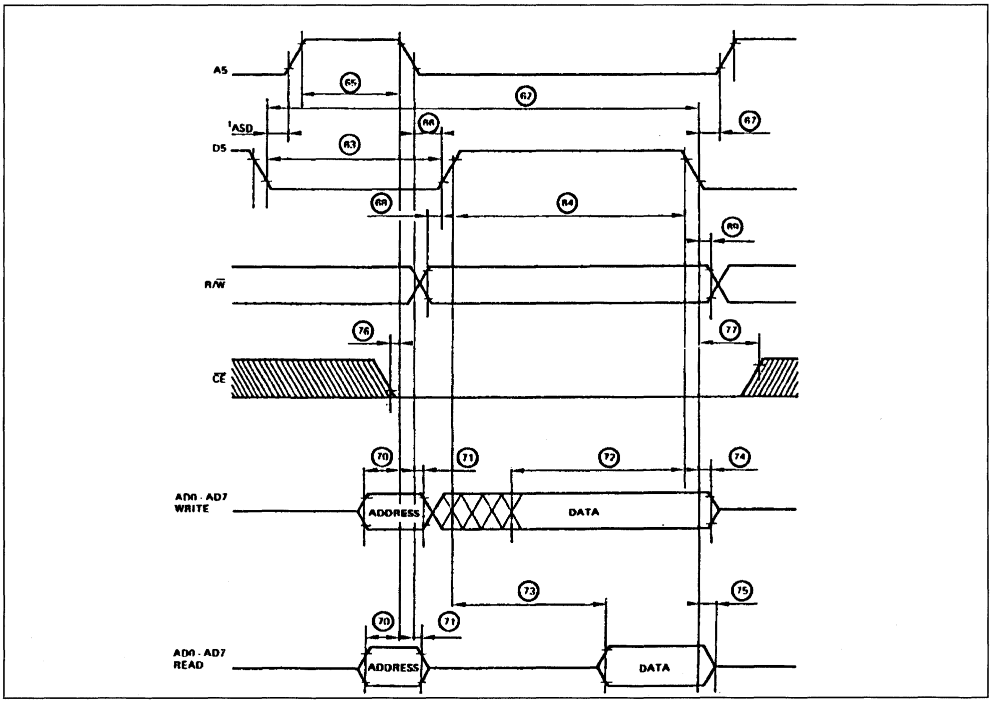


SGS-THOMSON MICROELECTRONICS

7.7.5. AC ELECTRICAL CHARACTERISTICS - MULTIPLEXED BUS TIMING (V_{CC} = 5.0 V_{DC} ± 5%, V_{SS} = 0 V_{DC}, T_A = 0°C to 70°C unless otherwise noted). See figures 7.9, 7.10.

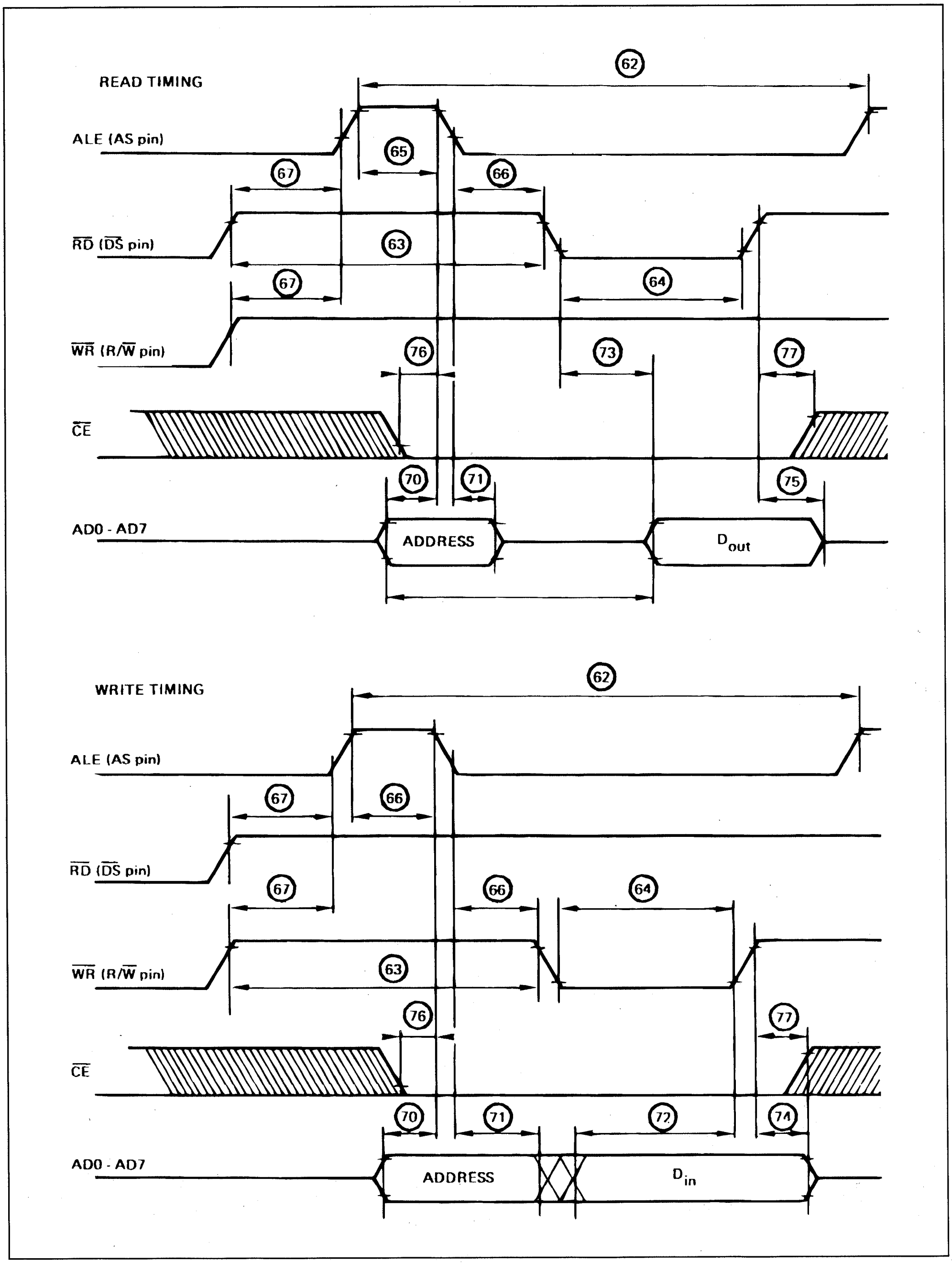
N°.	Parameter	Min.	Max.	Unit
62	Cycle Time	800		ns
63	Pulse Width $\overline{DS}$ Low or RD/$\overline{WR}$ High	350		ns
64	Pulse Width $\overline{DS}$ High or RD/$\overline{WR}$ Low	340		ns
65	Pulse Width AS/ALE High	100		ns
66	Delay $\underline{AS}$ Fall to $\overline{DS}$ Rise or ALE Fall to RD/WR Fall	30		ns
67	Delay $\overline{DS}$ or RD/$\overline{WR}$ Rise to AS/ALE Rise	30		ns
68	R/$\overline{W}$ Setup Time to $\overline{DS}$	100		ns
69	R/$\overline{W}$ Hold Time to $\overline{DS}$	10		ns
70	Address Setup Time to AS/ALE	20		ns
71	Address Hold Time to AS/ALE	20		ns
72	Data Setup Time to $\overline{DS}$ or $\overline{WR}$ (write)	280		ns
73	Delay Data to $\overline{DS}$ or RD (read)		250	ns
74	Data Hold Time to $\overline{DS}$ or WR (write)	20		ns
75	Data Hold Time to $\overline{DS}$ or RD (read)	0	100	ns
76	CE Setup TIme to AS/ALE Fall	20		ns
77	CE Hold Time to $\overline{DS}$, RD or $\overline{WR}$	20		ns

Figure 7.9 : Multiplexed Bus Timing Motorola Type.



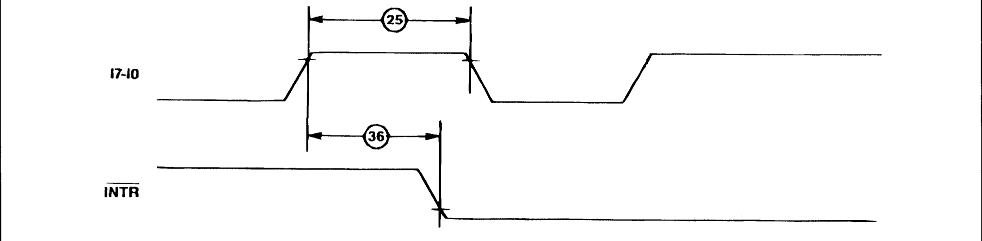
SGS-THOMSON MICROELECTRONICS

Figure 7.10 : Multiplexed Bus Timing - Intel Type.



SGS-THOMSON MICROELECTRONICS

Figure 7.11 : Interrupt Timing.



Note : Active edge is assumed to be the rising edge.

Figure 7.12 : Port Timing.

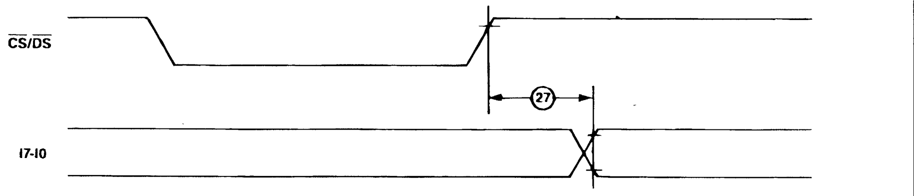


Figure 7.13 : Receiver Timing.

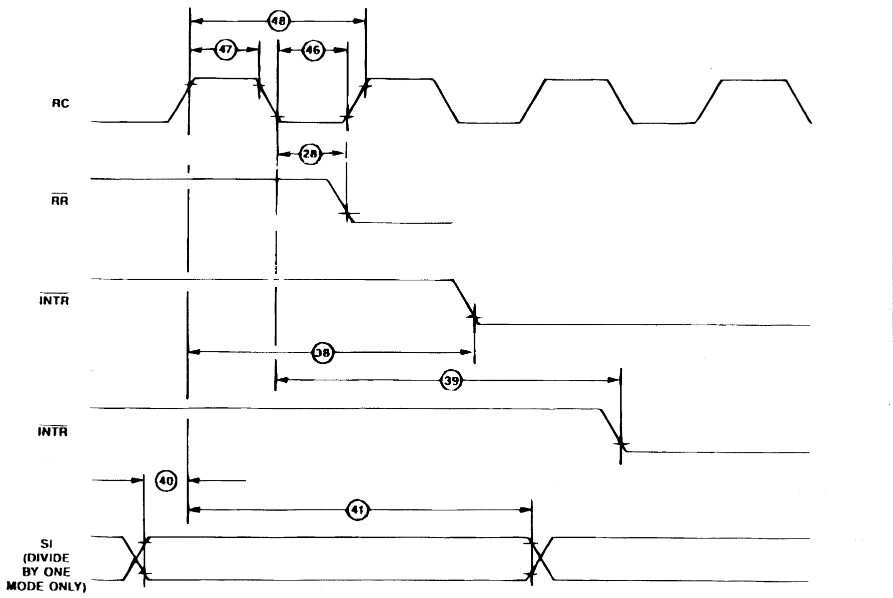


Figure 7.14 : Transmitter Timing.

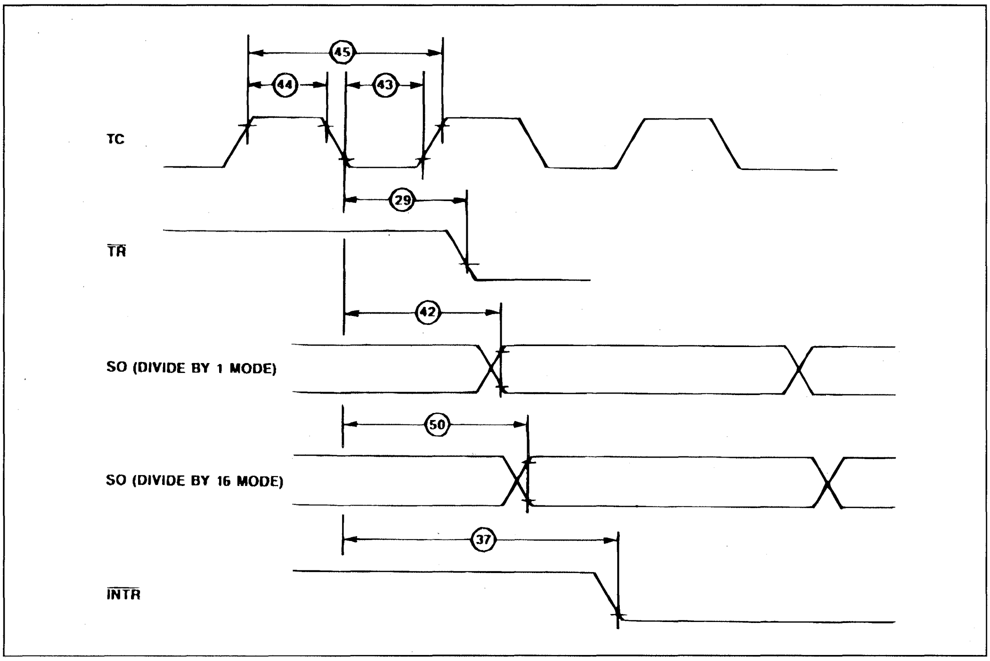


Figure 7.15 : Timer Timing.

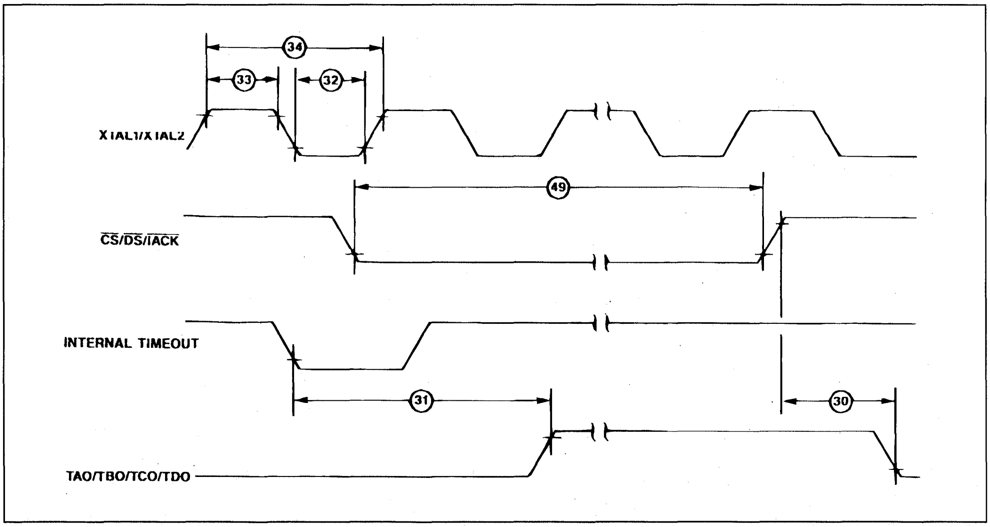
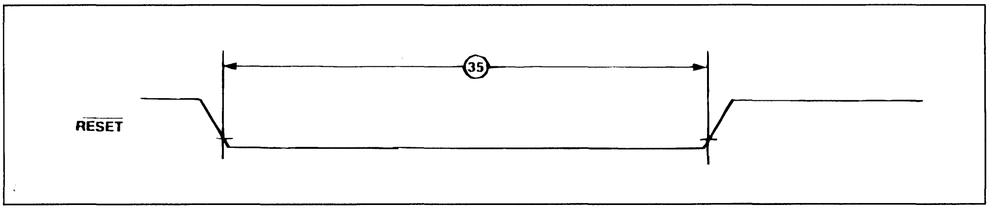


Figure 7.16 : Reset Timing.



7.8. TIMER AC CHARACTERISTICS

DEFINITION

Parameter
Error = Indicated Time Value - Actual Time Value
$t_{psc} = t_{CLK}$ x Prescale Value

INTERNAL TIMER MODE

Parameter	Value
Single Interval Error (free running) (see note 2)	± 100ns
Cumulative Internal Error	0
Error between Two Timer Reads	$\pm (t_{psc} - 4\ t_{CLK})$
Start Timer to Stop Timer Error	$2\ t_{CLK}$ + 100ns to $-(t_{psc} + 6\ t_{CLK}$ + 100ns)
Start Timer to Read Timer Error	0 to $-(t_{psc} + 6\ t_{CLK}$ + 400ns)
Start Timer to Interrupt Request Error (see note 3)	$-2\ t_{CLK}$ to $-(4\ t_{CLK}$ + 800 ns)

PULSE WIDTH MEASUREMENT MODE

Parameter	Value
Measurement Accuracy (see note 1)	$2\ t_{CLK}$ to $-(t_{psc} + 4\ t_{CLK})$
Minimum Pulse Width	$4\ t_{CLK}$

EVENT COUNTER MODE

Parameter	Value
Minimum Active Time of TAI and TBI	$4\ t_{CLK}$
Minimum Inactive Time of TAI and TBI	$4\ t_{CLK}$

Notes : 1. Error may be cumulative if repetitively performed.
2. Error with respect to t_{out} or IRQ if note 3 is true.
3. Assuming it is possible for the timer to make an interrupt request immediately.

7.9. FREQUENCY RANGE SUMMARY

The following table shows the maximum operating frequency of the TS68HC901 internal peripherals, according to the type used.

Type	8 MHz	5 MHz	4 MHz	Unit
Timer	8	5	4	MHz
USART	2	1.1	1	MHz
68000 Interface	8	5	4	MHz
6800 Interface	1	1	1	MHz
Multiplexed Interface	1.25	1.25	1.25	MHz

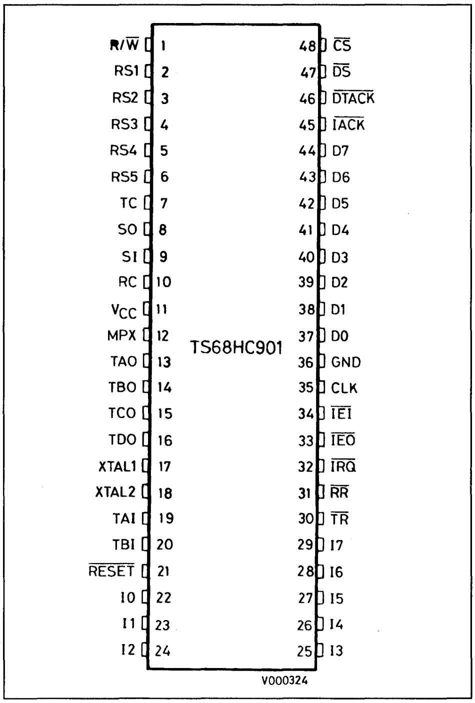
SECTION 8

MECHANICAL DATA AND ORDERING INFORMATION

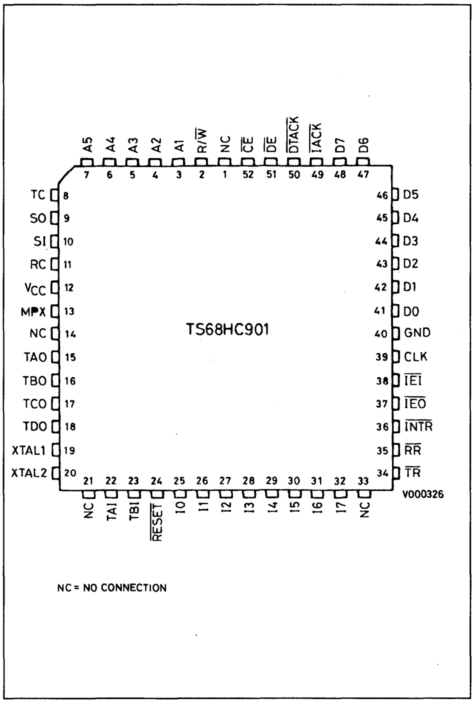
This section contains the pin assignments, package dimensions, and ordering information for the TS68HC901.

8.1. PIN ASSIGNMENTS

48-Pin Dual in-line.



52-Pin Quad Pack (PLCC).

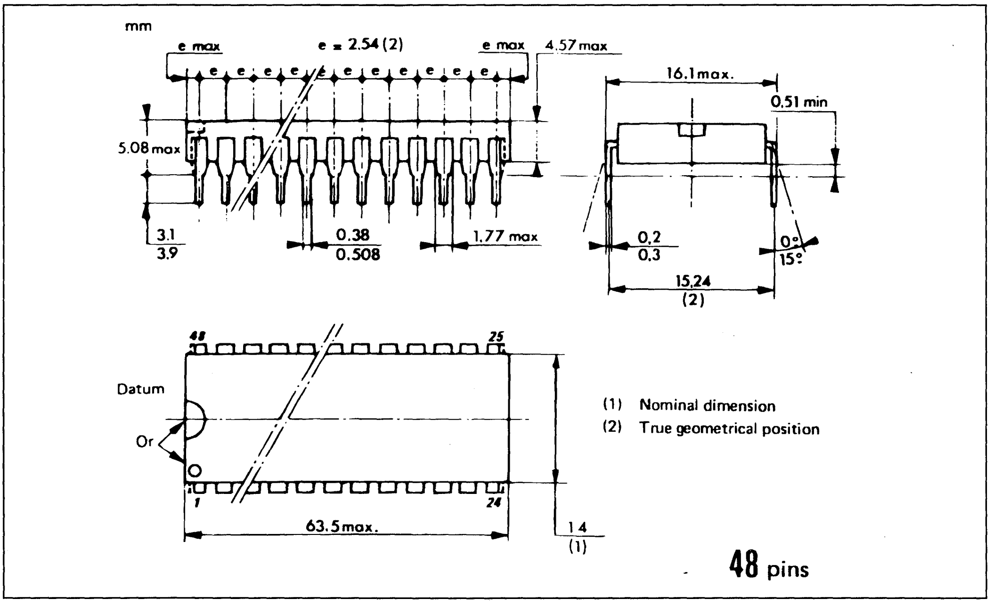


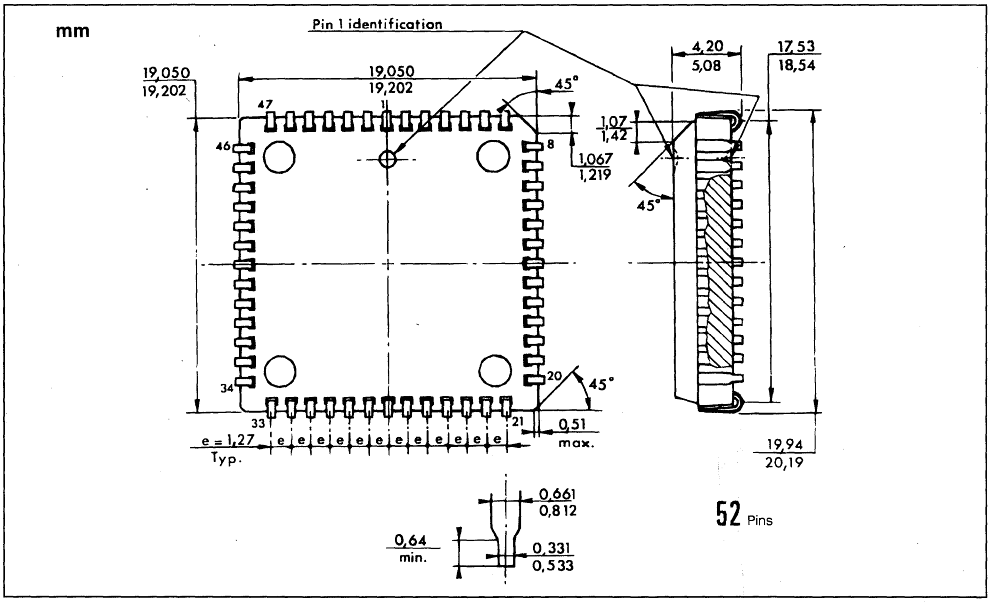
ORDERING INFORMATION

STANDARD VERSIONS

Part Number	Frequency (MHz)	Temperature Range	Package Type
TS68HC901CP4 TS68HC901CP5 TS68HC901CP8	4.0 5.0 8.0	0 °C to + 70 °C 0 °C to + 70 °C 0 °C to + 70 °C	Plastic DIL P Suffix
TS68HC901FN4 TS68HC901FN5 TS68HC901FN8	4.0 5.0 8.0	0 °C to + 70 °C 0 °C to + 70 °C 0 °C to + 70 °C	PLCC FN Suffix

8.2. PACKAGE MECHANICAL DATA





Z8500 UNIVERSAL PERIPHERALS

SGS-THOMSON
MICROELECTRONICS

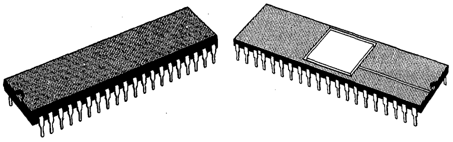
Z8038

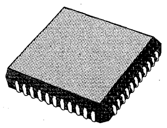
Z-FIO/FIFO INPUT/OUTPUT INTERFACE UNIT

DESCRIPTION

The Z8038 FIO (FIFO Input/Output Interface Unit) is a 128-byte buffer that interfaces two CPUs or a CPU and a peripheral device. Multiple FIOs can be used to create a 16-bit or wider data path, or two can be connected to form a 256-byte FIFO RAM buffer.

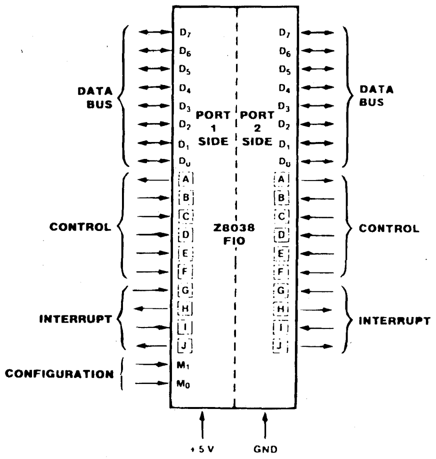
The FIO manages data transactions by assuming one of 12 operating modes, using one or more of the following signal configurations : Z-BUS™ high-byte microprocessor, Z-BUS low-byte microprocessor, non-Z-BUS microprocessor, interlocked 2-wire handshake I/O, and 3-wire handshake I/O. These configurations interface dissimilar CPUs or CPUs and peripheral devices running at different speeds or under different protocols. This allows asynchronous data transactions and byte-per-cycle DMA operation and cuts I/O overhead by as much as two orders of magnitude. Figure 1 and 2 illustrate logic functions and pin configuration.


PDIP-40 CDIP-40


PLCC44

(Ordering Information at the end of the datasheet)

Figure 1 : FIO Logic Functions.



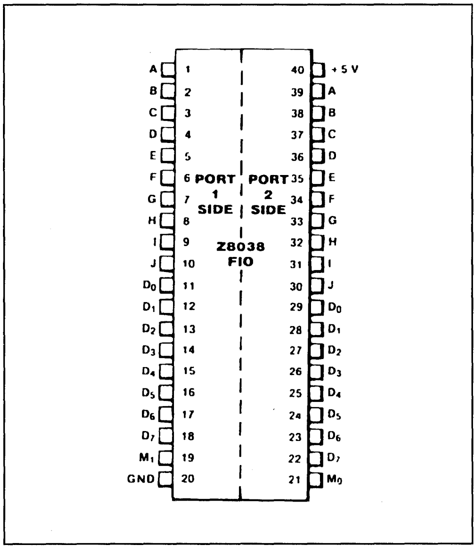
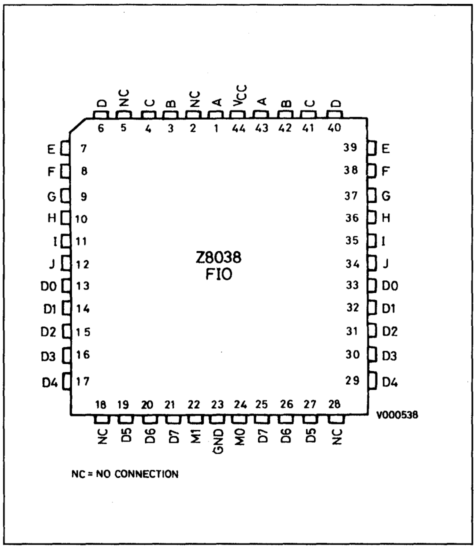


Figure 2 : Dual in Line Pin Connection.



D NC C B NC A VCC A B C D
E F G H I J D0 D1 D2 D3 D4
E F G H I J D0 D1 D2 D3 D4
NC D5 D6 D7 M1 GND M0 D7 D6 D5 NC
Z8038
FIO
V000538
NC = NO CONNECTION

Figure 2a: Chip Carrier Pin Connection.

ARCHITECTURAL DESCRIPTION

The FIO provides an asynchronous, 128-byte FIFO buffer between two CPUs or between a CPU and a peripheral device. Figure 3 shows the general architecture of the FIO. Two or more FIOs can be used in parallel to create a 16-bit or larger interface. Figure 4 shows a 32-bit interface constructed from four FIOs. Two FIOs can be combined to create 256 bytes of buffer space, and additional buffer space can be provided by adding one or more Z8060 FIFO buffers. Figure 5 shows a 512-byte buffer constructed from two FIOs and two FIFOs.

The two ports (1 and 2) are controlled through 16 programmable, directly accessible registers. These registers specify the operating mode of the FIO and provide a message register for CPU-to-CPU communication without involving the FIFO buffer.

The FIO supports DMA operation by facilitating variably-sized block transfers and data transactions to or from memory each machine cycle. Since devices can write to or read from either side of the FIO buffer asynchronously, as well as enable interrupts upon specified conditions, system I/O overhead can be significantly reduced.

ARCHITECTURAL DESCRIPTION (continued)

Figure 3 : FIO Functional Block Diagram.

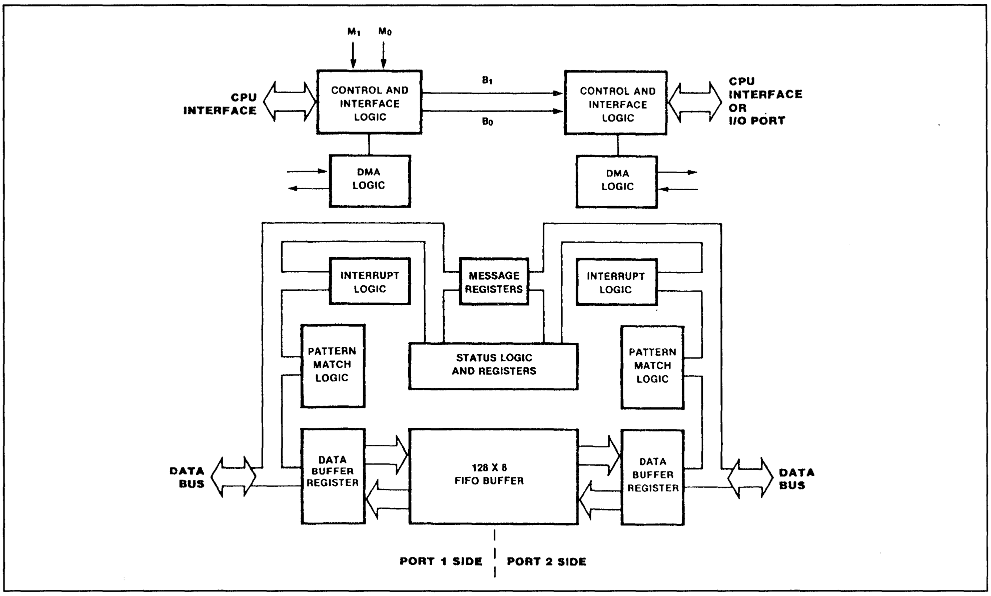
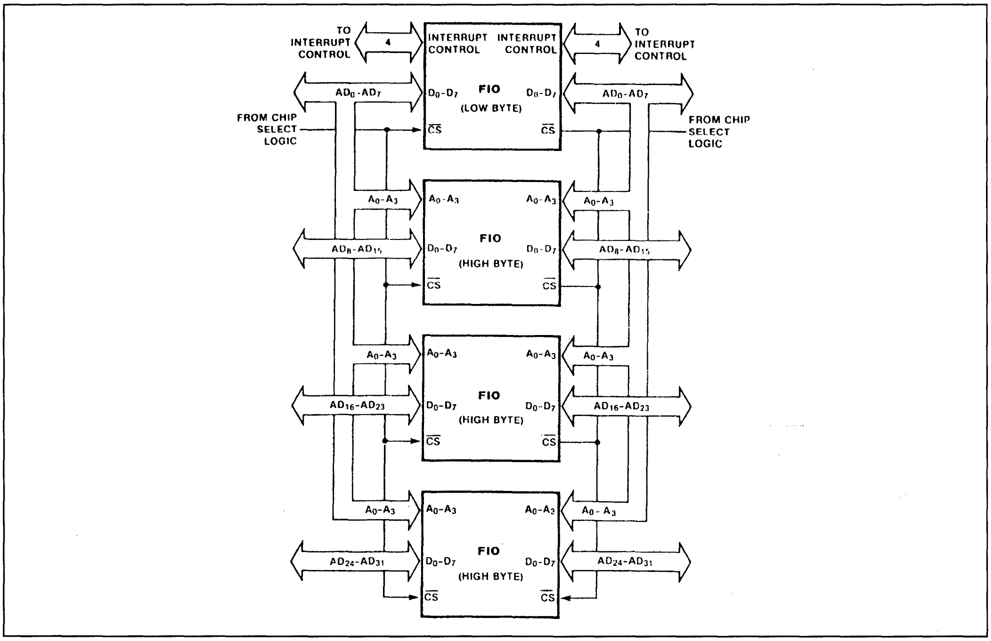
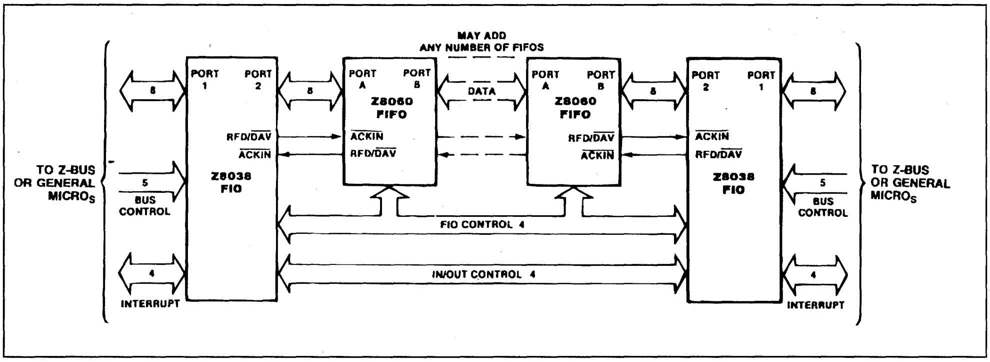


Figure 4 : FIO 32-Bit Width Expansion.



ARCHITECTURAL DESCRIPTION (continued)

Figure 5 : FIO-FIFO 512-Byte Buffer Expansion.



FUNCTIONAL DESCRIPTION

The FIO manages data transfers by assuming one of the 12 operating modes listed in Table 1. These modes are various combinations of the five interfacing protocols.

The five interfacing protocols are Z-BUS high-byte microprocessor, Z-BUS low-byte microprocessor, non-Z-BUS microprocessor (a generalized microprocessor interface), 2-wire interlocked handshake, and 3-wire handshake. Pins A through J carry the signals used in these interfaces ; Table 1-2 shows the pin assignments for each of the interfaces. Also see Appendix A for all 12 pin assignments.

Sixteen programmable registers control each port. Once the operating mode is specified, the internal registers are programmed to specify the direction of data transfer, the transfer of inter-CPU message bytes, external interrupt control, and various internal interrupt conditions.

The FIO improves I/O transaction efficiency by providing seven sources of interrupt generation. The FIO also provides an Interrupt vector that can be programmed to identify the reason for the interrupt.

Table 1 : Operating Modes.

Mode	M1	M0	B1	B0	Port 1	Port2
0	0	0	0	0	Z-BUS Low Byte	Z-BUS Low Byte
1	0	0	0	1	Z-BUS Low Byte	Non Z-BUS
2	0	0	1	0	Z-BUS Low Byte	3-Wie HS
3	0	0	1	1	Z-BUS Low Byte	2-Wire HS
4	0	1	0	0	Z-BUS Low Byte	Z-BUS High Byte
5	0	1	0	1	Z-BUS High Byte	Non Z-BUS
6	0	1	1	0	Z-BUS High Byte	3-Wire HS
7	0	1	1	1	Z-BUS High Byte	2-Wire HS
8	1	0	0	0	Non Z-BUS	Z-BUS Low Byte
9	1	0	0	1	Non Z-BUS	Non Z-BUS
10	1	0	1	0	Non Z-BUS	3-Wire HS
11	1	0	1	1	Non Z-BUS	2-Wire HS

FUNCTIONAL DESCRIPTION (continued)

Table 2 : Pin Assignments.

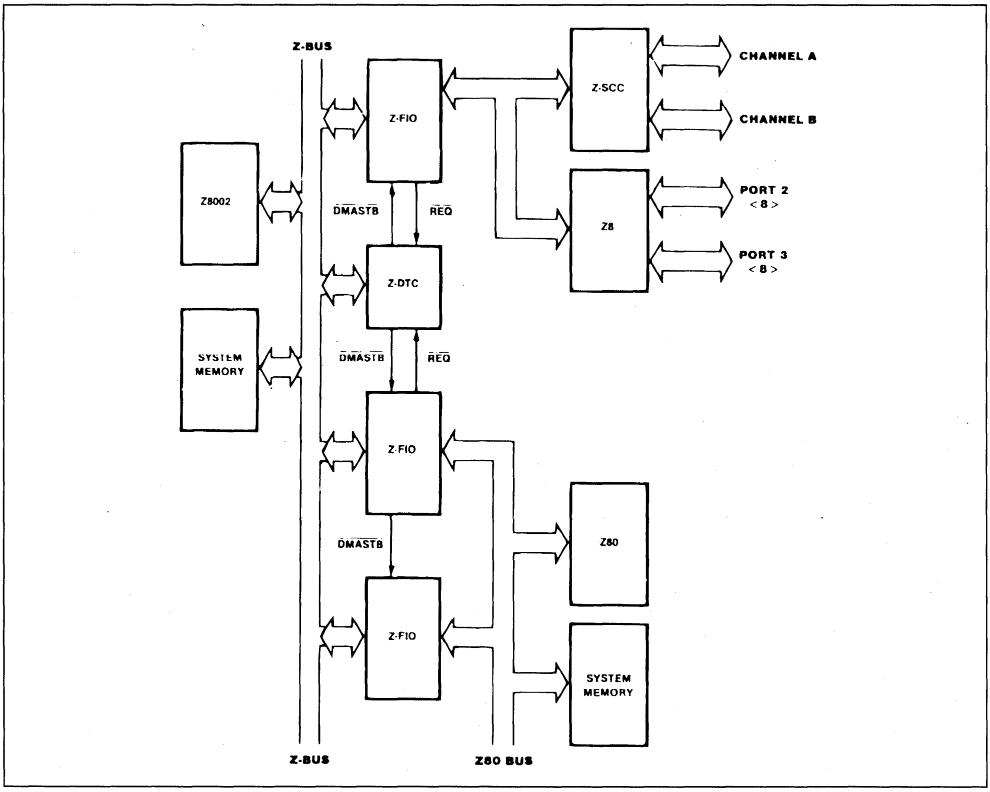
	Z-BUS Low Byte	Z-BUS High Byte	Non Z-BUS	Interlocked HS Port *	3-Wire HS Port *
A	$\overline{\text{REQ}}/\overline{\text{WAIT}}$	$\overline{\text{REQ}}/\overline{\text{WAIT}}$	$\overline{\text{REQ}}/\overline{\text{WAIT}}$	RFD/$\overline{\text{DAV}}$	RFD/$\overline{\text{DAV}}$
B	$\overline{\text{DMASTB}}$	$\overline{\text{DMASTB}}$	$\overline{\text{DACK}}$	$\overline{\text{ACKIN}}$	$\overline{\text{DAV}}$/DAC
C	$\overline{\text{DS}}$	$\overline{\text{DS}}$	$\overline{\text{RD}}$	FULL	DAC/RFD
D	R/$\overline{\text{W}}$	R/$\overline{\text{W}}$	$\overline{\text{WR}}$	EMPTY	EMPTY
E	$\overline{\text{CS}}$	$\overline{\text{CS}}$	$\overline{\text{CE}}$	$\overline{\text{CLEAR}}$	$\overline{\text{CLEAR}}$
F	$\overline{\text{AS}}$	$\overline{\text{AS}}$	C/$\overline{\text{D}}$	DATA DIR	DATA DIR
G	$\overline{\text{INTACK}}$	A_0	$\overline{\text{INTACK}}$	IN_0	IN_0
H	IEO	A_1	IEO	OUT_1	OUT_1
I	IEI	A_2	IEI	$\overline{\text{OE}}$	$\overline{\text{OE}}$
J	$\overline{\text{INT}}$	A_3	$\overline{\text{INT}}$	OUT_3	OUT_3

FIO IN DISTRIBUTED NETWORKS

In typical interfacing applications, such as the distributed processing system shown in Figure 6, the Port 1 CPU (Z8002) loads a series of bytes to the FIO's Port 1 registers. This block of control information characterizes the FIO's signal configuration for a specific transaction. The Port 2 CPU can then specify Port 2 operating conventions. The Port 1 CPU remains in control, dynamically reading or writing to the Port 1 registers, either to cause a single change or to replace an entire block of control bytes as specified.

FUNCTIONAL DESCRIPTION (continued)

Figure 6 : Using FIOs in a Distributed Network.



ARCHITECTURAL DESCRIPTION

As figure 3 shows, the FIO architecture is organized as two sets of interface logic and registers. Once programmed, these registers define the two groups of signal lines (A-J) that provide interfacing and timing data to control data flow into and out of the FIO. Table 3 shows how control signals are mapped to these pins in each of the five interfacing protocols.

This chapter also describes the 16 registers in each side of the FIO : the Control registers, Data Buffer register, Byte Count registers, Pattern Match registers, Message registers, and the Interrupt Status registers. All registers, are directly addressable, although in certain operating modes registers must be addressed indirectly.

REGISTER ARCHITECTURE

Thirty-two registers (16 for each port) control the operating modes and pin signals of the FIO. All registers are addressable and can be read from or written to, except the Byte Count and Message In registers, which are read only. All 16 registers are used by Port 1 ; Port 2 uses all 16 except for Control Register 2. These registers are organized into six groups that control similar functions or signals.

The control registers define the operating mode for both ports, as well as control device-level interrupts, some functional interrupts, addressing, Reset, Request/Wait, Data Direction, and Clear. The Data Buffer register buffers data into and out of the 128-byte FIFO. The Byte Count registers monitor the number of bytes left in the FIFO buffer and provides a compare byte for interrupt generation when a specified value is reached during data transfers. The Pattern Mathc registers hold a bit pattern that is compared with the byte in the data buffer. If the bit patterns match, the FIO requests an interrupt. They also provide a mask byte, which forces true comparisons for any bit set. The Message registers transfer byte messages between Port 1 and Port 2 CPUs when in the CPU-to-CPU interface modes. The Interrupt Status registers control generation of interrupt requests. The following sections describe these registers in detail.

Register Addressing Z-BUS High and Low Byte. The Right Justify Address (RJA) bit in Control Register 0 specifies the bits used in register addressing. In the Z-BUS low-byte configuration, when RJA is 0 (the default condition), Address/Data lines AD_1-AD_4 carry the register address. When RJA is 1, Address/Data lines AD_0-AD_3 carry the register address, allowing CPUs to place a 4-bit address on the lowest nibble of the Address/Data lines. Table 4 describes FIO register addressing.

The Z-BUS high-byte configuration is normally used with another FIO in the Z-BUS low-byte configuration, forming a 16-bit interface. In this configuration, Address/Data lines AD_0-AD_3 or AD_1-AD_4 are wired to pins G-J, as appropriate, and the RJA bit does not affect register addressing.

PINS COMMON TO BOTH SIDES

Pin Signals	Pin Names	Pin Numbers	Signal Description
M_0 M_1	M_0 M_1	21 19	M1 and M0 Program Port 1 Side CPU Interface
+ 5 V_{dc}	+ 5 V_{dc}	40	DC Power Source
GND	GND	20	DC Power Ground

REGISTER ARCHITECTURE (continued)

Table 3 : Pin Description.

Z-BUS LOW BYTE MODE

Pin Signals	Pin Names	Pin Numbers Port		Signal Description
		1	2	
AD_0-AD_7	D_0-D_7	11-18	29-22	Multiplexed Bidirectionnal Address/data Lines, Z-BUS Compatible
$\overline{REQ}/\overline{WAIT}$ (request/wait)	A	1	39	Output, Active Low, REQUEST (ready) Line for DMA Transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data Transfers
$\overline{DMASTB}$ (direct memory access strobe)	B	2	38	Input, Active Low. Strobes DMA Data to and from the FIFO Buffer.
DS (data strobe)	C	3	37	Input, Active Low. Provides timing for data transfer to or from FIO
R/$\overline{W}$ (read/write)	D	4	36	Input, active High signals CPU read from FIO ; Active Low signals CPU write to FIO.
$\overline{CS}$ (chip select)	E	5	35	Input, Active Low. Enables FIO. Latched on the Rising Edge of $\overline{AS}$.
AS (address strobe)	F	6	34	Input, Active Low. Addresses, $\overline{CS}$ and $\overline{INTACK}$ sampled while $\overline{AS}$ Low.
$\overline{INTACK}$ (interrupt acknowledge)	G	7	33	Input, Active Low. Acknowledge an Interrupt. Latched on the Rising Edge of $\overline{AS}$.
IEO (interrupt enable out)	H	8	32	Output, Active High. Sends interrupt enable to lower priority device IEI pin.
IEI (interrupt enable in)	I	9	31	Input, Active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (interrupt)	J	10	30	Output, Open Drain, Active Low. Signals FIO interrupt request to CPU.

REGISTER ARCHITECTURE (continued)

Z-BUS HIGH BYTE MODE

Pin Signals	Pin Names	Pin Numbers Port 1	2	Signal Description
AD_0-AD_7	D_0-D_7	11-18	29-22	Multiplexed Bidirectionnal Address/data Lines, Z-BUS Compatible
$\overline{REQ}/\overline{WAIT}$ (request/wait)	A	1	39	Output, Active Low, $\overline{REQUEST}$ (ready) Line for DMA Transfer ; $\overline{WAIT}$ Line (open-drain) Output for Synchronized CPU and FIO Data Transfers
$\overline{DMASTB}$ (direct memory access strobe)	B	2	38	Input, Active Low. Strobes DMA Data to and from the FIFO Buffer.
DS (data strobe)	C	3	37	Input, Active Low. Provides TIming for Transfer of Data to or from FIO.
$R/\overline{W}$ (read/write)	D	4	36	Input, Active High. Signals CPU read from FIO ; Active Low signals CPU write to FIO.
$\overline{CS}$ (chip select)	E	5	35	Input, Active Low. Enables FIO. Latched on the Rising Edge of $\overline{AS}$.
$\overline{AS}$ (address strobe)	F	6	34	Input, Active Low. Addresses, $\overline{CS}$ and $\overline{INTACK}$ are sampled while $\overline{AS}$ is Low.
A_0 (address bit 0)	G	7	33	Input, Active High.With A_1, A_2 and A_3, Addresses FIO Internal Registers.
A_1 (address bit 1)	H	8	32	Input, Active High. With A_0, A_2 and A_3, Addresses FIO Internal Registers.
A_2 (address bit 2)	I	9	31	Input, Active High. With A_0, A_1 and A_3, Addresses FIO Internal Registers.
A_3 (address bit 3)	J	10	30	Input, Active High. With A_0, A_1 and A_2, Addresses FIO Internal Registers.

REGISTER ARCHITECTURE (continued)

NON-Z-BUS MODE

Pin Signals	Pin Names	Pin Numbers Port		Signal Description
		1	2	
D_0-D_7	D_0-D_7	11-18	29-22	Bidirectional Data Bus
$\overline{REQ}/\overline{WT}$ (request/wait)	A	1	39	Output, Active Low, REQUEST (ready) Line for DMA Transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data Transfer.
$\overline{DACK}$ (DMA acknowledge)	B	2	38	Input, Active Low. DMA Acknowledge
$\overline{RD}$ (read)	C	3	37	Input, Active Low. Signals CPU read from FIO
$\overline{WR}$ (write)	D	4	36	Input, Active Low. Signals CPU write to FIO
$\overline{CE}$ (chip select)	E	5	35	Input, Active Low. Used to Select FIO
$C/\overline{D}$ (control/data)	F	6	34	Input, Active High. Identifies control byte on D0-D7 ; Active low indentifies Data byte on D0-D7.
$\overline{INTACK}$ (interrupt acknowledge)	G	7	33	Input, Active Low. Acknowledges an Interrupt
IEO (interrupt enable out)	H	8	32	Output, Active High. Sends interrupt enable to lower priority device IEI pin.
IEI (interrupt enable in)	I	9	31	Input, Active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (interrupt)	J	10	30	Output, Open Drain, Active Low. Signals FIO interrupt to CPU.

REGISTER ARCHITECTURE (continued)

PORT 2– I/O PORT MODE

Pin Signals	Pin Names	Pin Numbers	Mode	Signal Description
D_0-D_7	D_0-D_7	29-22	2-Wire HS* 3-Wire HS	Bidirectional Data Bus
RFD/$\overline{\text{DAV}}$ (ready for data/data available)	A	39	2-Wire HS 3-Wire HS	Output, RFD Active High. Signals peripherals that FIO is ready to receive data. $\overline{\text{DAV}}$ active low signals that FIO is ready to send data to peripherals.
$\overline{\text{ACKIN}}$ (acknowledge input)	B	38	2-Wire HS	Input, Active Low. Signals FIO that output data is received by peripherals or that input data is valid.
$\overline{\text{DAV}}$/DAC (data available/data accepted)	B	38	3-Wire HS	Input ; $\overline{\text{DAV}}$ (active low) signals that data is valid on bus. DAC (active high) signals that output data is accepted by peripherals.
FULL	C	37	2-Wire HS	Output, Open Drain, Active High. Signals that FIO buffer is full.
DAC/RFD (data accepted/ready for data)	C	37	3-Wire HS	Direction Controlled by Internal Programming. Both Active High. DAC (an output) signals that FIO has received data from peripheral ; RFD (an input) signals that the listeners are ready for data.
EMPTY	D	36	2-Wire HS 3-Wire HS	Output, Open Drain, Active High. Signals that FIO buffer is empty.
$\overline{\text{CLEAR}}$	E	35	2-Wire HS 3-Wire HS	Programmable Input or output, Active Low. Clears all Data from FIFO Buffer.
Data Direction	F	34	2-Wire HS 3-Wire HS	Programmable Input or output. Active High Signals Data Input to Port 2 ; Low Signals Data Output from Port 2.
IN_0	G	33	2-Wire HS 3-Wire HS	Input Line to D_0 of Control Register 3.
OUT_i	H	32	2-Wire HS 3-Wire HS	Output Line from D_1 of Control Register 3.
$\overline{\text{Output Enable}}$	I	31	2-Wire HS 3-Wire HS	Input Active Low. When Low, enables bus drivers. When High, floats bus drivers at high impedance.
OUT_3	J	30	2-Wire HS 3-Wire HS	Output Line from D_3 of Control Register 3.

* Handshake

REGISTER ARCHITECTURE (continued)

Table 4 : Register Addressing.

Non-Z-BUS	D_7-D_4	D_3	D_2	D_1	D_0	
Z-BUS High		A_3	A_2	A_1	A_0	
Z-BUS Low RJA = 0 / RJA = 1	AD_7-AD_5 / AD_7-AD_4	AD_4 / AD_3	AD_3 / AD_2	AD_2 / AD_1	AD_1 / AD_0	AD_0
Description						
Control Register 0	x	0	0	0	0	x
Control Register 1	x	0	0	0	1	x
Interrupt Status Register 0	x	0	0	1	0	x
Interrupt Status Register 1	x	0	0	1	1	x
Interrupt Status Register 2	x	0	1	0	0	x
Interrupt Status Register 3	x	0	1	0	1	x
Interrupt Vector Register	x	0	1	1	0	x
Byte Count Register	x	0	1	1	1	x
Byte Count Comparison Register	x	1	0	0	0	x
Control Register 2*	x	1	0	0	1	x
Control Register 3	x	1	0	1	0	x
Message Out Register	x	1	0	1	1	x
Message In Register	x	1	1	0	0	x
Pattern Match Register	x	1	1	0	1	x
Pattern Mask Register	x	1	1	1	0	x
Data Buffer Register	x	1	1	1	1	x

x = Don't Care
* Register is only on Port 1 side

Non-Z-BUS (Nonmultiplexed) Microprocessor. In the non-Z-BUS microprocessor configuration, the FIO connects to the system data bus via D_0-D_7 but not to the address bus. The CPU must provide address information on the data bus and the FIO must distinguish between control and data bytes. The $C/\overline{D}$ line (pin F) is pulled Low to indicate that the byte on Data lines D_0-D_7 should go into the Data Buffer register. When the $C/\overline{D}$ line is held High, incoming bytes are interpreted as control bytes. To write to any Control register, $C/\overline{D}$ is forced High ; the address of the destination register is written to the Register Pointer, then read or written to the target register.

Port 2 I/O Mode. When Port 2 is programmed for either I/O Port mode, none of the Port 2 side Control registers can be programmed. The only register available is Port 2's Data Buffer register.

Control Registers. Control Registers 0-3 define FIO operation according to their programming. The Port 1 Control registers must always be programmed ; the Port 2 Control registers are functional only if Port 2 is in a CPU interface mode, and not if Port 2 is in an I/O interface mode. Even when Port 2 interfaces to a CPU, the Port 2 interface is disabled unless the Port 1 CPU enables Port 2 by setting bit 0 in the Port 1 Control Register 2. Figure 7 shows the four Control registers.

Unless noted in the following descriptions, the register architectures of Port 1 and Port 2 are identical.

Control Register 0. Control Register 0 controls the Master Interrupt Enable, Disable Lower Daisy Chain, Nonvectored Interrupt, Vector Includes Status, Port 2 Operating Mode Select, Right Justify Address, and Reset functions. All bits except D_0 are forced to 0 by Reset. Each of these functions is programmed by a single bit, except for the Port 2 Operating Mode Select, which requires two bits to program one of four possible interfaces. When programming bits in this register, be careful not to accidentally change the Port 2 Operating Mode Control bits, B_1 and B_0.

If reset, bit 7 (Master Interrupt Enable) disables all interrupts from the FIO. If set, individual interrupts can be enabled by setting the appropriate IE (Interrupt Enable) bit in the Interrupt Status registers.

REGISTER ARCHITECTURE (continued)

Bit 6 (Disable Lower Daisy Chain), if set, disables interrupt generation by devices with a lower priority on the Z-BUS interrupt daisy chain by forcing IEO Low.

Bit 5 (No Vector On Interrupt), if set, floats the data bus outputs at high impedance during an active Interrupt Acknowledge signal. The interrupt source can be identified by polling the Interrupt Status registers' Interrupt Pending bits, or by testing to find which IUS bit was set. Do not set this bit when using the nonvectored interrupt on the Z8000.

Bit 4 (Vector Includes Status), if set, codes the reason for the interrupt request into the Interrupt vector, where it can be read by the CPU.

Bits 3 and 2 program the Port 2 operating mode, as described in Table 1. In this table, bit 3 is shown as B_1 and bit 2 is shown as B_0. (See also figure 7).

Bit 1 (Right Justify Address), if reset, specifies that bits 1-4 of the Address/Data bus address the internal FIO registers. This convention is compatible with the Z8001 and Z8002 byte I/O addressing modes and is the default state. When this bit is set, it specifies that bits 0-3 of the bus address the internal FIO registers. This convention supports other CPUs such as the Z8. Since the address of Control Register 0 is 00_H, the setting of RJA has no effect on addressing this register in any mode. In the non-Z-BUS configurations, it is forced to 1. This bit has no effect in Z-BUS High Byte mode, since the address information comes in on pins G, H, I, and J.

Bit 0 (Reset), if set, forces the FIO into a reset state, with all Control registers cleared and all Port 2 outputs at high impedance. This bit must be reset before initialization can take place.

Control Register 1. As shown in figure 7, bits in Control Register 1 freeze the Byte Count register value, indicate transmitted or received messages in the Message registers, stop request transfers upon pattern match or start request on Byte Count register true match, and enable Wait/Request signals. All bits in this register are cleared by reset.

Bit 7 of Control Register 1 is not used and must be programmed to 0. This bit always reads 0.

Bit 6 (Freeze Byte Count), if set, freezes the present value in the Byte Count register so the CPU can obtain a stable value to read. Since bytes can be transferred to and from FIO RAM while such a read is in progress, the frozen value in the Byte Count register might not reflect the ongoing byte count within FIO RAM. Bit 6 is reset upon completion of the CPU read of the Byte Count register. The ongoing count appears in the Byte Count register after the read.

Bit 5 (Message Register Out Full), if set, indicates that the CPU has placed a message in its Message Out register. This bit is reset when the receiving CPU reads the message in its Message In register. This bit is the other CPU's message IP bit and is a read-only bit.

Bit 4 (Message Register Interrupt Under Service), if set, indicates that the other CPU has received a message in its Message In register. This bit is the message IUS (Interrupt Under Service) bit of the other CPU and is a read-only bit.

Bit 3 (Stop Request On Pattern Match), if set, forces the request line High upon a true match between the current byte in the data buffer and the byte loaded into the Pattern Match register. DMA operation halts when the Request line is forced High.

Bit 2 (Start Request On Byte Count), if set, forces the Request line Low upon a true match between the current count in the Byte Count register and the value loaded into the Byte Count Compare register. DMA operation starts when the Request line is forced Low. See figures 27 and 28 for more details.

Bit 1 (Request/Wait), if set, selects the request function for use in high-speed data transfers, such as DMA transactions. If bit 1 is reset, the Wait function is selected (and the output is open drain) for use in lower-speed data transfers and CPU synchronization.

Bit 0 (Request/Wait Enable), if set, enables the Request/Wait signals (see bit 1) during CPU-to-CPU transactions. When reset, the Request/Wait pin is at high impedance.

Control Register 2. Only bits 0 and 1 are used in Control Register 2 (figure 7). Bits 7 through 2 are not used and should be programmed to 0. These bits always read 0. Bits 0 and 1 are cleared by reset. Only Port 1 has a Control Register 2. When the Port 2 CPU reads its Control Register 2, it will always read 00_H.

Bit 1 (Port 2 Side Handshake Enabled), if set, enables Port 2 I/O operation when bits 2 and 3 of Control Register 0 specify one of the I/O Handshake modes shown in Table 1. When this bit is reset, the handshake are disabled ; RFD/$\overline{\text{DAV}}$ is forced High.

Bit 0 (Port 2 Enabled), if set, allows the Port 2 CPU to program the Port 2 registers-when Port 2 is programmed as a CPU interface-and to assume control of Port 2 operation. The Port 2 side is at high impedance until this bit equals 1.

Control Register 3. Bits 7-4 in Control Register 3 enable and control the Clear and Data Direction si-

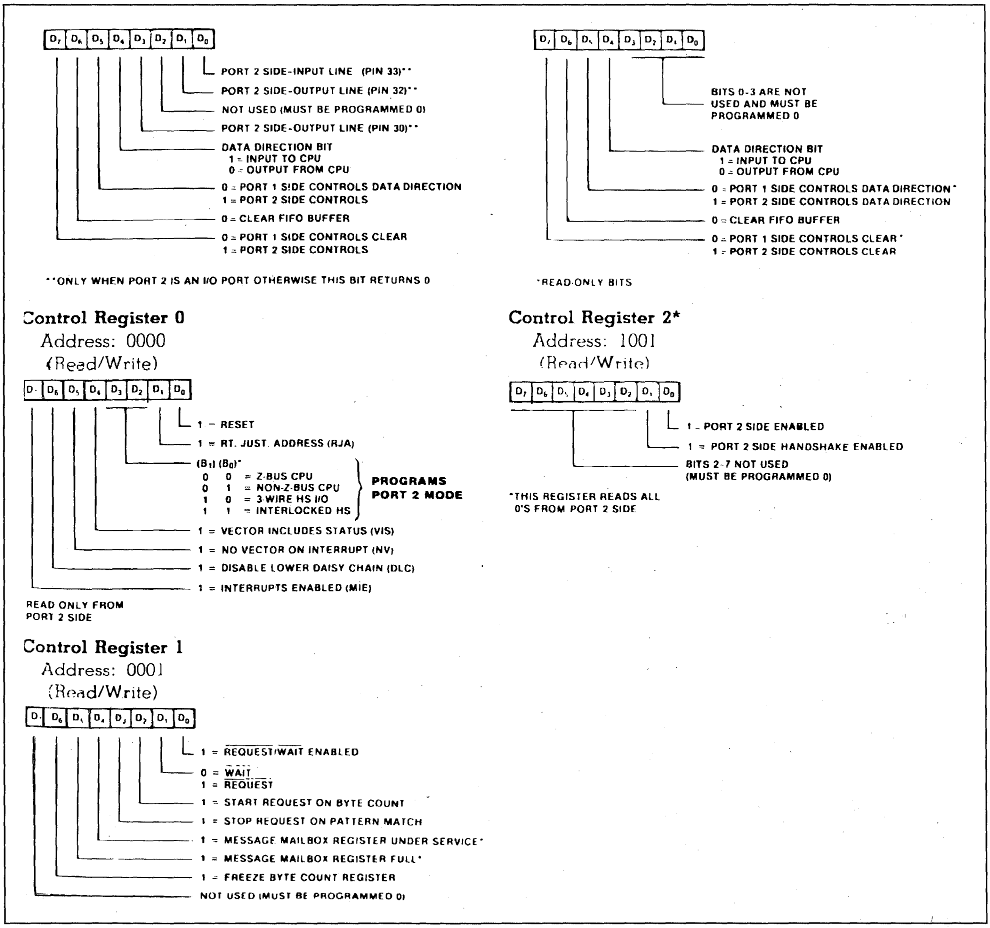
REGISTER ARCHITECTURE (continued)

gnals (figure 7) ; bits 3, 1, and 0 are simple I/O bits for the Port 1 CPU only. Bits 3-0 on the Port 2 side always read 0. Bit 2 is not used and must be programmed to 0 ; it will always read 0. All bits in this register are cleared to 0 by reset. Bits 7 and 5 can be programmed only by the Port 1 CPU. These bits are read-only in the Port 2 Control Register 3.

Bit 7 (Clear, Port 2/Port 1), for CPU-to-CPU operation if set, specifies that Port 2 CPU controls the Clear signal (bit 6). If bit 7 is reset, Port 1 CPU controls the Clear signal. If Port 2 is programmed as an I/O port, this bit programs pin 35 as either input or output (0 = output, 1 = input). This bit can be programmed only by the Port 1 CPU. This is a read-only bit for the Port 2 CPU.

Bit 6 (Clear FIFO Buffer), if reset, clears the FIO RAM buffer. Bit 7 controls which CPU issues Clear signals. (See preceding paragraph). This bit becomes a read-only bit by the CPU not in control of the Clear function.

Figure 7 : Control Registers 0-3.


D7 D6 D5 D4 D3 D2 D1 D0
PORT 2 SIDE-INPUT LINE (PIN 33)**
PORT 2 SIDE-OUTPUT LINE (PIN 32)**
NOT USED (MUST BE PROGRAMMED 0)
PORT 2 SIDE-OUTPUT LINE (PIN 30)**
DATA DIRECTION BIT 1 = INPUT TO CPU 0 = OUTPUT FROM CPU
0 = PORT 1 SIDE CONTROLS DATA DIRECTION 1 = PORT 2 SIDE CONTROLS
0 = CLEAR FIFO BUFFER
0 = PORT 1 SIDE CONTROLS CLEAR 1 = PORT 2 SIDE CONTROLS
**ONLY WHEN PORT 2 IS AN I/O PORT OTHERWISE THIS BIT RETURNS 0
BITS 0-3 ARE NOT USED AND MUST BE PROGRAMMED 0
DATA DIRECTION BIT 1 = INPUT TO CPU 0 = OUTPUT FROM CPU
0 = PORT 1 SIDE CONTROLS DATA DIRECTION* 1 = PORT 2 SIDE CONTROLS DATA DIRECTION
0 = CLEAR FIFO BUFFER
0 = PORT 1 SIDE CONTROLS CLEAR* 1 = PORT 2 SIDE CONTROLS CLEAR
*READ-ONLY BITS
Control Register 0
Address: 0000
(Read/Write)
Control Register 2*
Address: 1001
(Read/Write)
1 = RESET
1 = RT. JUST. ADDRESS (RJA)
(B1) (B0)*
0 0 = Z-BUS CPU
0 1 = NON-Z-BUS CPU
1 0 = 3-WIRE HS I/O
1 1 = INTERLOCKED HS
PROGRAMS PORT 2 MODE
1 = VECTOR INCLUDES STATUS (VIS)
1 = NO VECTOR ON INTERRUPT (NV)
1 = DISABLE LOWER DAISY CHAIN (DLC)
1 = INTERRUPTS ENABLED (MIE)
READ ONLY FROM PORT 2 SIDE
1 = PORT 2 SIDE ENABLED
1 = PORT 2 SIDE HANDSHAKE ENABLED
BITS 2-7 NOT USED (MUST BE PROGRAMMED 0)
*THIS REGISTER READS ALL 0'S FROM PORT 2 SIDE
Control Register 1
Address: 0001
(Read/Write)
1 = REQUEST/WAIT ENABLED
0 = WAIT
1 = REQUEST
1 = START REQUEST ON BYTE COUNT
1 = STOP REQUEST ON PATTERN MATCH
1 = MESSAGE MAILBOX REGISTER UNDER SERVICE*
1 = MESSAGE MAILBOX REGISTER FULL*
1 = FREEZE BYTE COUNT REGISTER
NOT USED (MUST BE PROGRAMMED 0)

REGISTER ARCHITECTURE (continued)

Bit 5 (Data Direction, Port 2/Port 1), for CPU-to-CPU operation, if set, specifies that Port 2 CPU controls the direction of data flow through FIO RAM. If bit 5 is reset, Port 1 CPU controls the Data Direction signal. If Port 2 is programmed as an I/O port, this bit programs pin 34 as either an input or an output (0 = output, 1 = input). This bit can be programmed only by the Port 1 CPU. This is read-only bit for the Port 2 CPU.

Bit 4 (Data Direction), if set, specifies that data moves from the FIO into the CPU (1 = input to CPU ; 0 = output from CPU). If bit 4 is reset, data moves from the controlling CPU into the FIO. When Port 2 is an I/O port, a zero (0) on the Data Direction pin (pin 34) means that Port 2 is an output handshake port. Conversely when the Data Direction pin is a one (1), Port 2 is an input handshake port.

Bits 3 and 1 are Port 2 outputs (pins 30 and 32, respectively) when Port 2 is programmed as an I/O port. These bits always read 0 for the Port 2 CPU.

Bit 2 is not used and must be programmed to 0. This bit always reads 0.

Bit 0 reads pin 33, a Port 2 input, when Port 2 is programmed as an I/O port. When Port 2 is not programmed as an I/O port, this bit reads 0. This bit always reads 0 for the Port 2 CPU.

Data Buffer Register. The Data Buffer register (figure 8) latches the byte written to or read from FIO RAM. When the Pattern Match register contains a compare byte, it is compared with the bytes passing through the Data Buffer register.

Byte Count Registers. There are two Byte Count registers (figure 8). One contains the ongoing count of bytes present in FIO RAM ; the other holds a byte value that is compared with this ongoing count.

Byte Count Register. The Byte Count register contains an ongoing count of the number of bytes in FIO RAM. This value is the number of bytes written into RAM minus the number of bytes read from RAM. The largest value contained in this register is 80_H (128 decimal). This value can be frozen by setting bit D_6 in Control Register 1. This halts the count so an accurate read can be accomplished. This is a read-only register.

Byte Count Compare Register. The Byte Count Compare register is loaded with a byte value that is

Figure 8 : Data Buffer and Byte Count Registers.

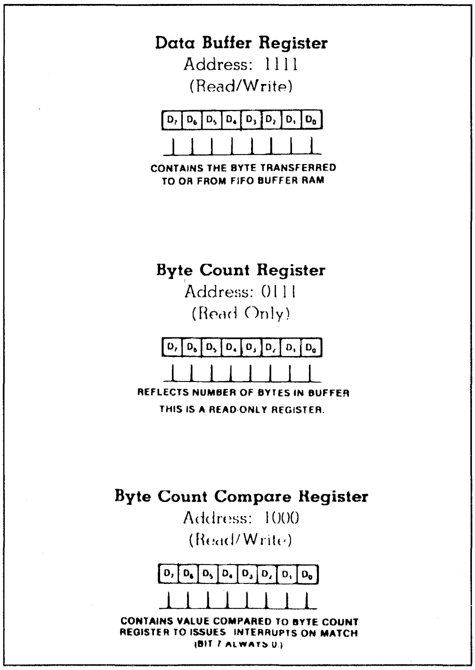
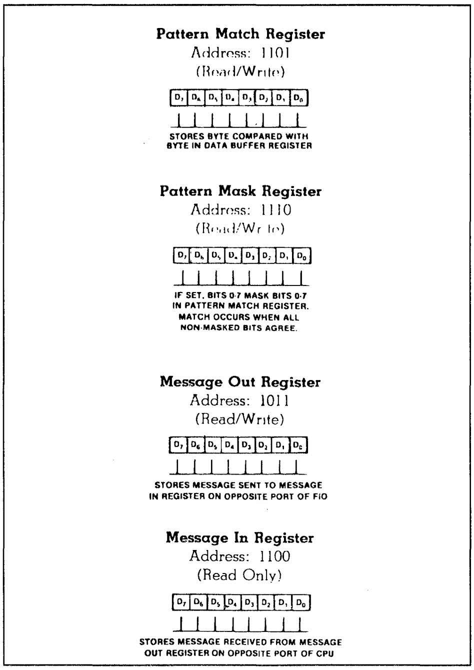

Data Buffer Register
Address: 1111
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
CONTAINS THE BYTE TRANSFERRED TO OR FROM FIFO BUFFER RAM
Byte Count Register
Address: 0111
(Read Only)
D7 D6 D5 D4 D3 D2 D1 D0
REFLECTS NUMBER OF BYTES IN BUFFER THIS IS A READ-ONLY REGISTER.
Byte Count Compare Register
Address: 1000
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
CONTAINS VALUE COMPARED TO BYTE COUNT REGISTER TO ISSUES INTERRUPTS ON MATCH (BIT 7 ALWAYS 0)

Figure 9 : Pattern Match and Message Registers.


Pattern Match Register
Address: 1101
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
STORES BYTE COMPARED WITH BYTE IN DATA BUFFER REGISTER
Pattern Mask Register
Address: 1110
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
IF SET, BITS 0-7 MASK BITS 0-7 IN PATTERN MATCH REGISTER. MATCH OCCURS WHEN ALL NON-MASKED BITS AGREE.
Message Out Register
Address: 1011
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
STORES MESSAGE SENT TO MESSAGE IN REGISTER ON OPPOSITE PORT OF FIO
Message In Register
Address: 1100
(Read Only)
D7 D6 D5 D4 D3 D2 D1 D0
STORES MESSAGE RECEIVED FROM MESSAGE OUT REGISTER ON OPPOSITE PORT OF CPU

REGISTER ARCHITECTURE (continued)

continuously compared with the ongoing byte count in the Byte Count register. The largest programmable value is 7F$_H$ (127 decimal). Bit 7 always reads 0. If the Byte Count Compare Interrupt bit is set (see Interrupt Status Register 2), an interrupt occurs upon a true match.

Pattern Match Registers. The Pattern Match register contains a byte for comparison with the byte in the Data Buffer. The Pattern Mask register contains a byte pattern used to force a true match.

The Pattern Match and Data Buffer registers are in an unknown state on power-up or after reset and may therefore match. This match may set the Pattern Match IP and the Pattern Match flag.

Pattern Match Register. The Pattern Match register (figure 9) contains a byte for comparison with the byte in the Data Buffer register. As each byte in a data transaction passes through the Data Buffer register, it is compared with the value programmed in the Pattern Mathc register. Upon a true match, an interrupt can be generated if enabled in Interrupt Status Register 1. One or more bits in the Pattern Match register can be masked by the value in the Pattern Mask register.

Pattern Mask Register. The Pattern Mask register (figure 9) contains a bit pattern used to force a true match between the pattern in the Pattern Match register and the pattern in the Data Buffer register. If a Pattern Mask bit is set, the corresponding bit in the Pattern Match register always matches the corresponding bit in the Data Buffer register. A match occurs when all nonmasked bits agree. All 1s in this register forces a pattern match. This register is cleared to 0 by reset.

Message Registers. The Message registers (figure 9) transfer messages between CPUs.

Message Out Register. The CPU sends a message by writing the byte into its Message Out register which also puts it in the other CPU's Message In register. When this is done, the receiving CPU receives an interrupt request (if the IE bit is enabled in Interrupt Status Register 0). The Message Out register of one CPU is the Message In register for the other CPU.

Message In Register. The CPU receiving a message byte also receives an interrupt request (if the IE bit is enabled in Interrupt Status Register 0), which signifies that a message has appeared in its Message In register. This is a read-only register.

Interrupt Registers. There are four Interrupt Status registers, shown in figure 10, which control and monitor the FIO internal interrupt functions. Prioritized interrupts can occur if they are enabled. The CPU can read the appropriate status register to see if a specific interrupt condition is enabled ; other bits in the Interrupt Status registers show if an interrupt is pending or if it under service. Each interrupt condition is described by three Interrupt Status bits : IE (Interrupt Enabled), IP (Interrupt Pending) and IUS (Interrupt Under Service).

Interrupt Status Register 0. In Interrupt Status Register 0, only bits 7, 6, and 5 are used. Bits 4-0 must be programmed to 0 ; these bits always read 0. All bits are cleared to 0 by reset.

Bit 7 is the Message Interrupt Under Service (IUS) bit and, when set, indicates that a Message interrupt is under service by the receiving CPU.

Bit 6 is the Message Interrupt Enable (IE) bit and, when set, indicates that a message interrupt will be issued when a message is received in the Message In register.

Bit 5 is the Message Interrupt Pending (IP) bit and, when set, indicates that a message interrupt to the receiving CPU is pending. This bit is reset when the read of the Message In register is completed.

Interrupt Status Register 1. Interrupt Status Register 1 controls and monitors Data Direction Change interrupts and Pattern Match interrupts. It also shows if a true pattern match has taken places. All bits except D_1 and D_0 are cleared by reset. Bits D_1 and D_0 may be a 1 or 0 depending upon whether a match condition exists or not.

Bits 7, 6 and 5 show the status of Data Direction Change interrupts and are cleared by reset. When bit 7 is set, a Data Direction Change interrupt is under service. When bit 6 is set, Data Direction Change interrupts are enabled. When bit 5 is set, a Data Direction Change interrupt is pending. Bits 7 and 5 can also be set or reset by program command. Bit 6 is programmed set or reset.

Bit 4 is not used and must be programmed to 0. This bit always reads 0.

Bits 3, 2, and 1 show the status of Pattern Match interrupts. When bit 3 is set, a Pattern Match interrupt is under service. When bit 2 is set, Pattern Match interrupts are enabled. When bit 1 is set, a Pattern Match interrupt is pending. Bits 3 and 1 can also be

REGISTER ARCHITECTURE (continued)

set or reset by program command. Bit 2 is programmed set or reset. Because of a possible match condition on power-up or reset, the match IP bit should be cleared before enabling the Pattern Match IE bit.

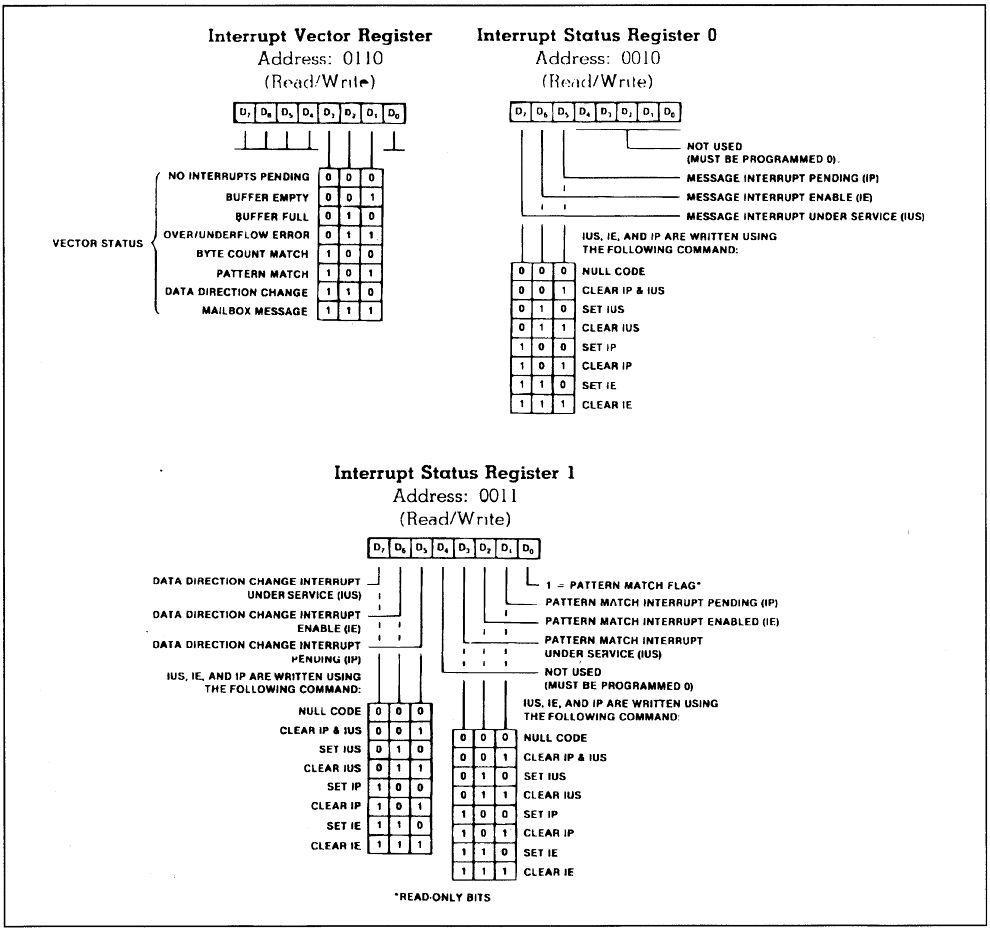
Bit 0 (Pattern Match Flag) is 1 whenever a true pattern match exists between the Data Buffer and the Pattern Match register. The Data Buffer register and the Pattern Match register initialize in an undefined state during reset or power-up. Consequently, the Pattern Match Flag bit can be set, indicating a false pattern match. If Pattern Match interrupts are not going to be used, this bit can be put in a known state, 1, by writing FF$_H$ to the Pattern Mask register.

Interrupt Status Register 2. Interrupt Status Register 2 controls and monitors Byte Count Compare and Overflow and Underflow Error interrupts. This register is cleared to 0 by reset.

Bits 7, 6, and 5 show the status of Byte Count Compare interrupts. When bit 7 is set, a Byte Count Compare interrupt is under service. When bit 6 is set, Byte Count Compare Interrupts are enabled. When bit 5 is set, a Byte Count Compare Interrupt is pending. Bits 7 and 5 can also be set or reset by program command. Bit 6 is programmed set or reset.

Bit 4 is set whenever an attempt is made to write into a full FIO (Overflow Error). This bit is cleared when

Figure 10 : Interrupt Registers 0-3 Interrupt Vector Register.


Interrupt Vector Register
Address: 0110
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
VECTOR STATUS
NO INTERRUPTS PENDING 0 0 0
BUFFER EMPTY 0 0 1
BUFFER FULL 0 1 0
OVER/UNDERFLOW ERROR 0 1 1
BYTE COUNT MATCH 1 0 0
PATTERN MATCH 1 0 1
DATA DIRECTION CHANGE 1 1 0
MAILBOX MESSAGE 1 1 1
Interrupt Status Register 0
Address: 0010
(Read/Write)
NOT USED (MUST BE PROGRAMMED 0)
MESSAGE INTERRUPT PENDING (IP)
MESSAGE INTERRUPT ENABLE (IE)
MESSAGE INTERRUPT UNDER SERVICE (IUS)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING COMMAND:
0 0 0 NULL CODE
0 0 1 CLEAR IP & IUS
0 1 0 SET IUS
0 1 1 CLEAR IUS
1 0 0 SET IP
1 0 1 CLEAR IP
1 1 0 SET IE
1 1 1 CLEAR IE
Interrupt Status Register 1
Address: 0011
(Read/Write)
DATA DIRECTION CHANGE INTERRUPT UNDER SERVICE (IUS)
DATA DIRECTION CHANGE INTERRUPT ENABLE (IE)
DATA DIRECTION CHANGE INTERRUPT PENDING (IP)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING COMMAND:
NULL CODE 0 0 0
CLEAR IP & IUS 0 0 1
SET IUS 0 1 0
CLEAR IUS 0 1 1
SET IP 1 0 0
CLEAR IP 1 0 1
SET IE 1 1 0
CLEAR IE 1 1 1
1 = PATTERN MATCH FLAG*
PATTERN MATCH INTERRUPT PENDING (IP)
PATTERN MATCH INTERRUPT ENABLED (IE)
PATTERN MATCH INTERRUPT UNDER SERVICE (IUS)
NOT USED (MUST BE PROGRAMMED 0)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING COMMAND:
0 0 0 NULL CODE
0 0 1 CLEAR IP & IUS
0 1 0 SET IUS
0 1 1 CLEAR IUS
1 0 0 SET IP
1 0 1 CLEAR IP
1 1 0 SET IE
1 1 1 CLEAR IE
*READ-ONLY BITS

REGISTER ARCHITECTURE (continued)

the Error IP bit is cleared. This bit is not set when the Wait function is programmed.

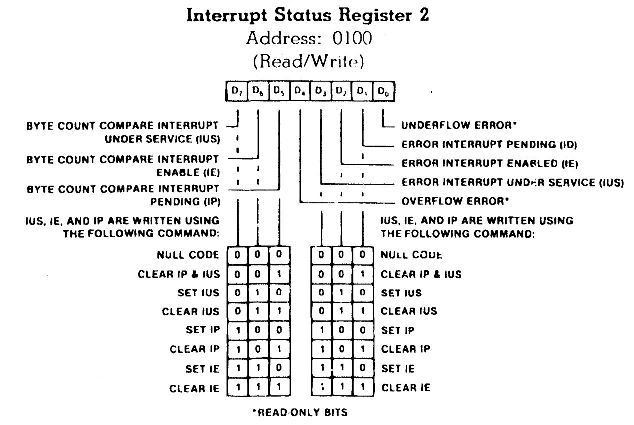
Bits 3, 2, and 1 show the status of Error interrupts. These bits control overflow and underflow errors ; the CPU must read bits 0 and 4 to determine whether overflow or underflow conditions apply. When bit 3 is set, an Error interrupt is under service. When bit 2 is set, Error interrupts are enabled. When bit 1 is set, an Error interrupt is pending. Bits 3 and 1 can also be set or reset by program command. Bit 2 is programmed set or reset.

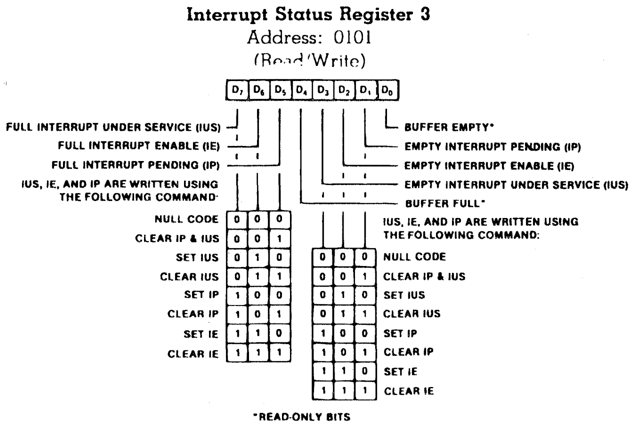
Bit 0 is set whenever an attempt is made to read from an empty FIO (underflow error). This bit is cleared when the Error IP is cleared. This bit is not set when the Wait function is programmed.

Interrupt Status Register 3. Interrupt Status Register 3 controls and monitors Buffer Full and Buffer Empty interrupts. All bits except D0 are cleared by reset.

Bits 7, 6, and 5 show the status of Buffer Full interrupts. When bit 7 is set, a Buffer Full interrupt is under service. When bit 6 is set, Buffer Full interrupts are enabled. When bit 5 is set, a Buffer Full interrupt

Figure 10 : Interrupt Registers 0-3 and Interrupt Vector Register (continued).


Interrupt Status Register 2
Address: 0100
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
BYTE COUNT COMPARE INTERRUPT UNDER SERVICE (IUS)
BYTE COUNT COMPARE INTERRUPT ENABLE (IE)
BYTE COUNT COMPARE INTERRUPT PENDING (IP)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING COMMAND:
NULL CODE 0 0 0
CLEAR IP & IUS 0 0 1
SET IUS 0 1 0
CLEAR IUS 0 1 1
SET IP 1 0 0
CLEAR IP 1 0 1
SET IE 1 1 0
CLEAR IE 1 1 1
UNDERFLOW ERROR*
ERROR INTERRUPT PENDING (IP)
ERROR INTERRUPT ENABLED (IE)
ERROR INTERRUPT UNDER SERVICE (IUS)
OVERFLOW ERROR*
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING COMMAND:
0 0 0 NULL CODE
0 0 1 CLEAR IP & IUS
0 1 0 SET IUS
0 1 1 CLEAR IUS
1 0 0 SET IP
1 0 1 CLEAR IP
1 1 0 SET IE
1 1 1 CLEAR IE
*READ-ONLY BITS



REGISTER ARCHITECTURE (continued)

is pending. Bits 7 and 5 can also be set or reset by program command. Bit 6 is programmed set or reset. In CPU-to-I/O operation for handshake, the full IP bit is set when the buffer is full and the Full pin (pin 37) is High. For the 3-wire handshake the Buffer Full IP is set when the FIO buffer is full.

Bit 4 is a 1 when the FIO buffer is full in CPU-to-CPU operation. In CPU-to-I/O operation for 2-wire handshake, bit 4 is 1 when the buffer is full and the Full pin (pin 37) is High. For the 3-wire handshake this bit is a 1 when the FIO buffer is full.

Bits 3, 2, and 1 show the status of Buffer Empty interrupts. When bit 3 is set, a Buffer Empty interrupts is under service. When bit 2 is set, Buffer Empty interrupts are enabled. When bit 1 is set, a Buffer Empty interrupt is pending. In CPU-to-I/O operation, the Empty IP bit is set when the buffer is empty and the Empty pin (pin 37) is High.

Bit 0 is 1 when the FIO buffer is empty in CPU-to-CPU operation. In CPU-to-I/O operation, bit 0 is 1 when the buffer is empty and when the Empty pin (pin 36) is High.

Interrupt Vector Register. The Interrupt Vector register (figure 10) holds a byte used as the address of an interrupt service routine (or of a table indexing into such groups of routines). This register can be used in two ways. The Interrupt Vector register can be programmed with a byte address that is gated onto the address bus lines during the Interrupt Acknowledge cycle. This provides a direct vector to a generalized interrupt service routine. If bit 4 of Control Register 0 (Vector Includes Status) is set, however, the Interrupt Vector register includes a 3-bit status code reflecting the reason for the interrupt request as the contents of bits 3, 2, and 1. When MIE is 1, other than during an Interrupt Acknowledge cycle, the Interrupt Vector register always reflects the FIO status in these bits, regardless of whether or not the Vector Includes Status bit is set. When MIE is 0, the base vector can be read back. This provides a means to read what was written to this register (also see table 5). The bit codes are :

Code Bits 3	2	1	Encoded Data
1	1	1	Received Message in Message In Register
1	1	0	Change in Data Transaction Direction
1	0	1	Valid Pattern Match
1	0	0	Valid Byte Count Compare
0	1	1	Overflow or Underflow Error
0	1	0	Buffer Full
0	0	1	Buffer Empty
0	0	0	No Interrupts Pending

Table 5 : Data Read from Interrupt Vector Register.

	MIE = 1 VIS = 1	**MIE = 1 VIS = 0**	**MIE = 0 VIS = 1 or 0**
Interrupt Acknowledge Read (INTACK = 0)	Status With Vector	No Status With Vector (base vector)	Interrupts Are Disabled (high impedance)
Non Interrupt Acknowledge Read (INTACK = 1)	Status With Vector	Status With Vector	No Status With Vector (base vector)

FUNCTIONAL DESCRIPTION

The FIO provides an interface between two CPUs or between a CPU and an I/O device. It interfaces to both Z-BUS and non-Z-BUS systems and emulates 2-wire and 3-wire I/O protocols. Figure 11 shows possible FIO operations.

These interfacing tasks require different sets of control signals. After removing the FIO from its reset state and programming the various registers, the FIO assumes one of the 12 operating modes shown in Table 1. In each mode, Port 2 presents one of five possible groups of signals on pins A-J. These signals correspond to the Z-BUS high-byte configuration, Z-BUS low-byte configuration, non-Z-BUS (general microprocessor) configuration, 2-wire handshake configuration, and 3-wire handshake configuration. Port 1 presents one of three groups of signals on pins A-J, corresponding to Z-BUS high-byte, Z-BUS low-byte, or non-Z-BUS configurations.

This chapter describes the signals used in FIO data transactions ; discusses reset operation and uses for the 12 operating modes ; and explains how control signals form interfaces to other devices, how DMA transactions occur, and how the FIO handles internal interrupts and participates in the system-level, daisy-chain interrupt protocols.

INTERFACING SIGNAL

The FIO manages data transfers with control signals, some issued from the bus master CPU, satellite CPU, DMA device, or I/O device, some created from external logic, and some issued from within the FIO itself. There are five basic signal configurations that combine to form 12 operating modes, as shown in Table 1. The control signals used in each configuration are shown mapped to their respective pins in figure 2. The signals fall into various groups, as described in the following sections. Reset is not actually a control signal, but since the condition is caused by control signal states (or programming) it is treated as such here.

Z-BUS High-Byte and Low-Byte. The Z-BUS high-byte and low-byte microprocessor configurations are normally used together to form a 16-bit multiplexed address/data bus interface that is compatible with the Z-BUS. The Z-BUS low-byte microprocessor configuration can be used by itself as an 8-bit interface to a multiplexed bus. The Z-BUS high-byte microprocessor configuration can also be used this way, but it lacks interrupt interfacing signals.

Signals. In the Z-BUS cofigurations, the FIO uses the following set of signals :

Request/Wait, DMA Strobe, Data Strobe, Read/Write, Chip Select, Address Strobe, and the interrupt arbitration signals Interrupt Acknowledge, Interrupt Request, Interrupt Enable In, and Interrupt Enable Out.

$\overline{\text{REQ}}/\overline{\text{WAIT}}$ *(Request/Wait, output, 3-state, active Low).* $\overline{\text{REQ}}$ is sent to a DMA device to begin a DMA transfer, and $\overline{\text{WAIT}}$ (an open drain output) tells the CPU that the FIO buffer is full (if data is output from the CPU to the FIO) or that the FIO buffer is empty (if data is input to the CPU from the FIO).

$\overline{\text{DMASTB}}$ *(DMA Strobe, input, active Low).* This signal is received by the FIO from a DMA controller. When Low, it acts like a Read/Write signal, strobing data to or from the Data Buffer register directly during DMA transactions.

$\overline{\text{DS}}$ *(Data Strobe, input, active Low).* When Low, $\overline{\text{DS}}$ specifies that the multiplexed address/data bus is carrying data rather than address information.

R/$\overline{\text{W}}$ *(Read/Write, input).* Read and Write signals specify read and write operations. When R/$\overline{\text{W}}$ is Low, writes are enabled. When R/W is High, reads are enabled.

$\overline{\text{CS}}$ *(Chip Select, input, active Low).* $\overline{\text{CS}}$ is usually a signal derived from the controlling CPU's signals by external logic. When Low, $\overline{\text{CS}}$ selects the FIO and enables it for operation. This signal is latched on the rising edge of AS.

$\overline{\text{AS}}$ *(Address Strobe, input, active Low).* When low, $\overline{\text{AS}}$ specifies that the multiplexed address/data bus is carrying address information rather than data.

Z-BUS Low-Byte Configuration Only. In the Z-BUS low-byte configuration, the FIO uses four interrupt arbitration signals to participate in the daisy-chain prioritized interrupt arbitration scheme.

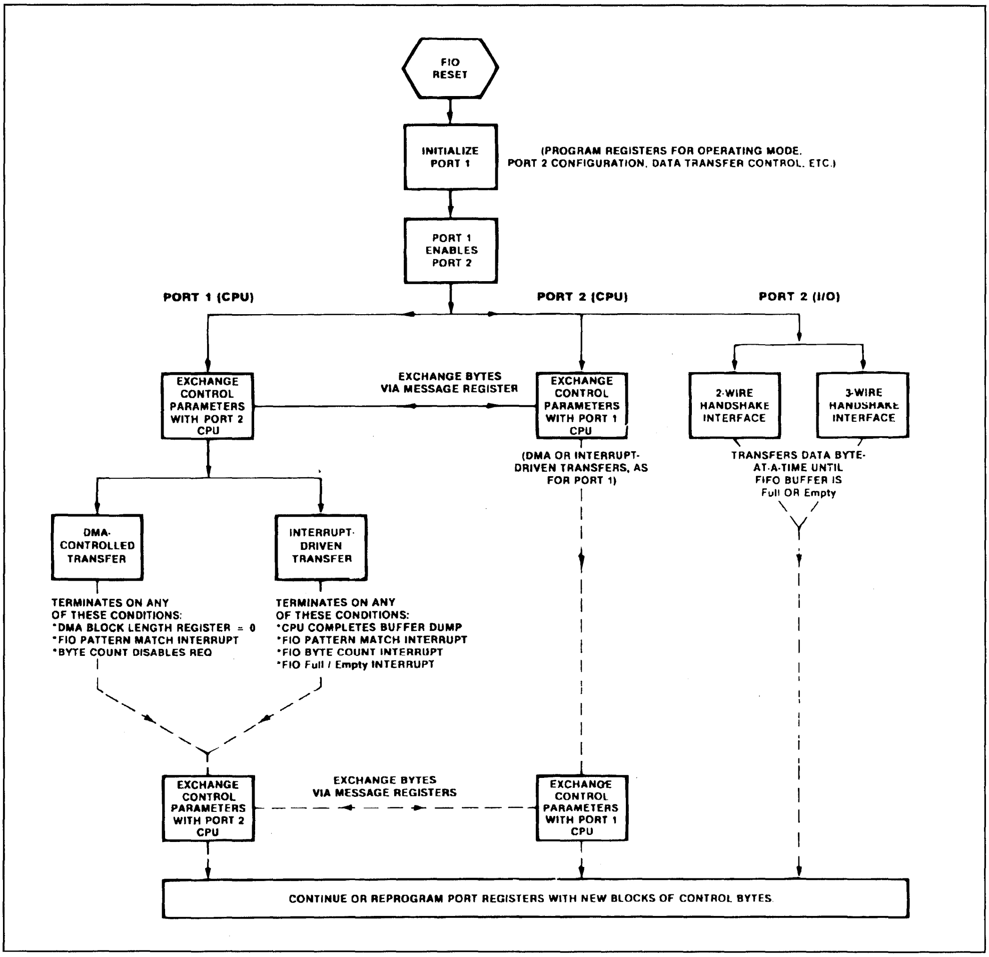
$\overline{\text{INTACK}}$ *(Interrupt Acknowledge, input, active Low),* IEI *(Interrupt Enable In, input, active High),* IEO *(Interrupt Enable Out, output, active High),* $\overline{\text{INT}}$ *(Interrupt Request, output, open drain, active Low).* $\overline{\text{INTACK}}$, IEI, IEO, and $\overline{\text{INT}}$ control FIO interrupt operation. $\overline{\text{INT}}$ requests an interrupt from the CPU when

INTERFACING SIGNALS (continued)

active (Low) ; INTACK is the interrupt acknowledgement from the CPU. IEI is the input for the daisy-chain, prioritized interrupt enable signal. IEO is the corresponding output for the same signal.

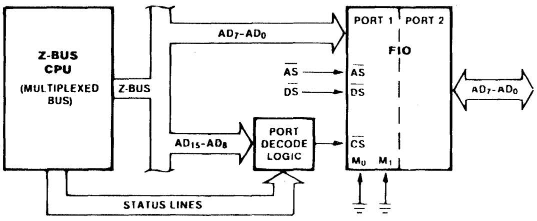
Figure 12 shows a typical use of the Z-BUS low-byte configuration. The M_0 and M_1 pins are shown tied to ground ; this specifies Z-BUS low-byte configuration.

Figure 11 : FIO Operational Summary.


FIO RESET
INITIALIZE PORT 1
(PROGRAM REGISTERS FOR OPERATING MODE. PORT 2 CONFIGURATION, DATA TRANSFER CONTROL, ETC.)
PORT 1 ENABLES PORT 2
PORT 1 (CPU)
PORT 2 (CPU)
PORT 2 (I/O)
EXCHANGE CONTROL PARAMETERS WITH PORT 2 CPU
EXCHANGE BYTES VIA MESSAGE REGISTER
EXCHANGE CONTROL PARAMETERS WITH PORT 1 CPU
2-WIRE HANDSHAKE INTERFACE
3-WIRE HANDSHAKE INTERFACE
(DMA OR INTERRUPT-DRIVEN TRANSFERS, AS FOR PORT 1)
TRANSFERS DATA BYTE-AT-A-TIME UNTIL FIFO BUFFER IS Full OR Empty
DMA-CONTROLLED TRANSFER
INTERRUPT-DRIVEN TRANSFER
TERMINATES ON ANY OF THESE CONDITIONS:
•DMA BLOCK LENGTH REGISTER = 0
•FIO PATTERN MATCH INTERRUPT
•BYTE COUNT DISABLES REQ
TERMINATES ON ANY OF THESE CONDITIONS:
•CPU COMPLETES BUFFER DUMP
•FIO PATTERN MATCH INTERRUPT
•FIO BYTE COUNT INTERRUPT
•FIO Full / Empty INTERRUPT
EXCHANGE CONTROL PARAMETERS WITH PORT 2 CPU
EXCHANGE BYTES VIA MESSAGE REGISTERS
EXCHANGE CONTROL PARAMETERS WITH PORT 1 CPU
CONTINUE OR REPROGRAM PORT REGISTERS WITH NEW BLOCKS OF CONTROL BYTES.

INTERFACING SIGNALS (continued)

Figure 12 : Z-BUS Low-Byte Configuration.



Z-BUS Low Byte Interface Operation. Address/Data lines AD_0-AD_{15} leave the CPU. During the time $\overline{AS}$ is active, signifying that the address/data bus currently carries and address, lines AD_0 – AD_7 carry the 4-bit address of the internal register addressed within the FIO.

Once the Z-BUS low-byte configuration has been specified as shown in Table 1, the $\overline{CS}$ signal is Low, and the address information has been delivered (as shown by the removal of an active $\overline{AS}$ Signal), the $\overline{DS}$ signal goes Low and data appears on AD_0 – AD_7. When bit 0 is set in Control register 0, the FIO reset, all control registers cleared, and all outputs (except IEO on the port 1 side) floating at high impedance, the FIO is ready to program.

Register Addressing. In the Z-BUS low-byte configuration, the FIO allows two methods for register addressing under control of the Right Justify Address bit (bit D_1) of Control Register 0. When bit D_1 is reset (the default condition), address bus lines AD_1 – AD_4 are used for register addressing, and the other Address/Data lines are ignored. This convention assures compatibility with the Z8000 byte I/O addressing protocols. When D_1 is set, address bus lines AD_0-AD_3 are used for register addresses and the other Address/Data lines are ignored. Lines AD_8 – AD_{15} carry the 8-bit port address decoded by the port decode logic along with status to provide an active $\overline{CS}$ signal ; $\overline{AS}$ latches it into the FIO. Table 6 describes Z-BUS low-byte register addressing.

Table 6 : Register Addressing.

Z-BUS Low RJA = 0 / RJA = 1	AD_7-AD_5 / AD_7-AD_4	AD_4 / AD_3	AD_3 / AD_2	AD_2 / AD_1	AD_1 / AD_0	AD_0
Description						
Control Register 0	x	0	0	0	0	x
Control Register 1	x	0	0	0	1	x
Interrupt Status Register 0	x	0	0	1	0	x
Interrupt Status Register 1	x	0	0	1	1	x
Interrupt Status Register 2	x	0	1	0	0	x
Interrupt Status Register 3	x	0	1	0	1	x
Interrupt Vector Register	x	0	1	1	0	x
Byte Count Register	x	0	1	1	1	x
Byte Count Comparison Register	x	1	0	0	0	x
Control Register 2*	x	1	0	0	1	x
Control Register 3	x	1	0	1	0	x
Message Out Register	x	1	0	1	1	x
Message in Register	x	1	1	0	0	x
Pattern Match Register	x	1	1	0	1	x
Pattern Mask Register	x	1	1	1	0	x
Data Buffer Register	x	1	1	1	1	x

x = Don't care
* Register is only on Port 1 side

INTERFACING SIGNALS (continued)

Reset Operation. In the Z-BUS configurations, the FIO is hardware reset when both AS and DS are forced Low (normally an illegal condition) or it is software reset by writing a 1 to the reset bit in Control Register 0. When Port 1 is reset, Port 2 is also reset. If the Port 1 CPU reads the Port 1 registers during reset, they return all 0s except Control Register 0, which reads 01_H. All of Port 2's signal lines are floating in this state, and all inputs are ignored. If Port 2 is reset by itself, Port 1 is not reset. When the Port 2 CPU reads the Port 2 registers during reset, they return all 0s except Control Register 0, which reads 01_H.

Before data transactions can take place, the FIO must be taken out of the reset state by writing 00_H to Control Register 0. No other bits in this register can be programmed while clearing the reset bit. The Port 1 CPU then enables Port 2 by setting bit D_0 of the Port 1 Control Register 2. The Port 2 CPU can determine when it is enabled by reading its Control Register 0, which is read as a "floating" data bus if not enabled or as 01_H if enabled.

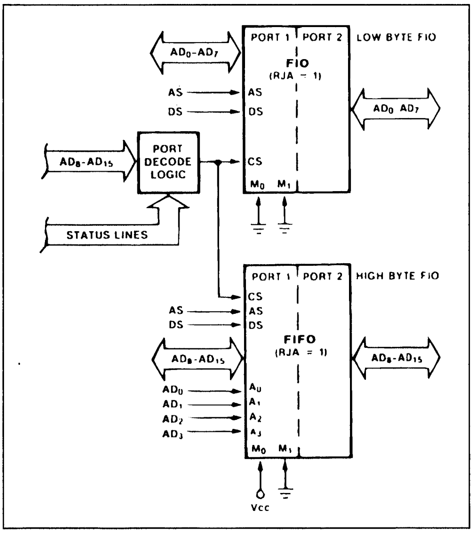
Interrupts. The Z-BUS low-byte configuration supports the prioritized daisy-chain interrupt protocols. $\overline{INTACK}$, IEI, IEO, and $\overline{INT}$ control FIO interrupt operation. $\overline{INT}$ sends an interrupt signal to the CPU ; $\overline{INTACK}$ is the interrupt acknowledgement from the CPU. IEI is the input for the daisy-chain prioritized interrupt signal. IEO is the corresponding output for the same signal. When using two FIOs as a 16-bit Z-BUS interface, the low-byte FIO must handle external interrupt transactions, since the corresponding pins on the high-byte FIO are used for internal register addressing. This is not a handicap, since all $\overline{INT}$ conditions are the same for both FIOs, except for the pattern match condition.

Figure 13 shows a typical use of the Z-BUS low-byte and high-byte configurations with two FIOs, forming a 16-bit multiplexed bus interface. The single FIO in figure 12 runs parallel with another FIO in the Z-BUS high-byte configuration. The Z-BUS high-byte configuration is specified as shown in Table 1.

Z-BUS High Byte Interface Operation. As in the Z-BUS low-byte configuration, Address/Data lines $AD_0 - AD_{15}$ leave the Port 1 CPU. During an active AS signal, lines $AD_0 - AD_7$ carry the address of the internal register addressed within the FIO.

While $\overline{AS}$ is active, lines $AD_8 - AD_{15}$ carry the port address to the port decode logic, which issues an active $\overline{CS}$ signal. This signal is also supplied to the high-byte FIO. The RJA bit is not used on the high-byte FIO.

Figure 13 : Z-BUS High-Byte and Low-Byte Configuration.



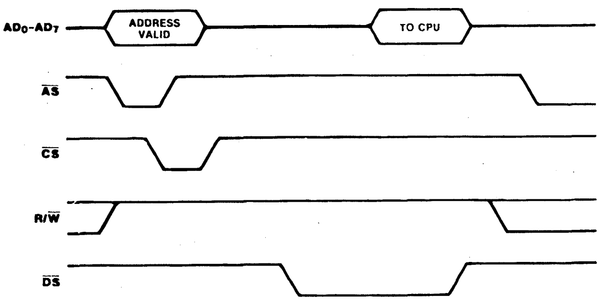
Once the low-byte and high-byte configurations have been specified as shown in Table 1, the $\overline{CS}$ signal is Low, and the address information has been delivered (as shown by the removal of an active AS signal), the $\overline{DS}$ signal goes Low, indicating that true data is on the Address/Data lines.

Register Addressing. Both FIOs require the same internal register address, but lines $AD_0 - AD_7$ go to the low-byte FIO only. For this reason, the signals on the four Address/Data pins used to specify FIO internal register addresses must be supplied to pins $A_0 - A_3$ of the high-byte FIO, as shown in figure 14. Since pins $AD_0 - AD_3$ are used for addressing, the low-byte FIO must handle external interrupt interfacing. Table 7 describes Z-BUS high byte register addressing.

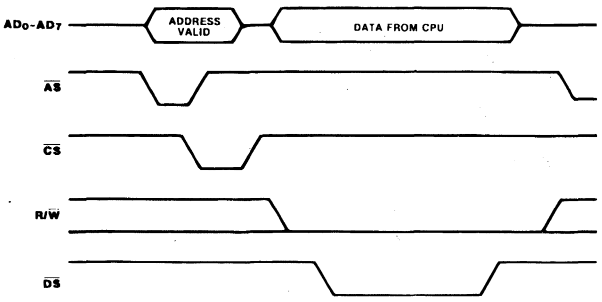
Reset Operation. In the Z-BUS configurations, the FIO is hardware reset when both $\overline{AS}$ and $\overline{DS}$ are forced Low (normally an illegal condition) or it is software reset by writing a 1 to the reset bit in Control Register 0. When Port 1 is reset, Port 2 is also reset. If the Port 1 CPU reads the Port 1 registers during reset, they return all 0s except Control Register 0, which reads 01_H. All of Port 2's signal lines are floating in this state, and all inputs are ignored. If Port 2 is reset by itself, Port 1 is not reset. When the Port 2 CPU reads the Port 2 registers during reset, they return all 0s except Control Register 0, which reads 01_H.

INTERFACING SIGNALS (continued)

Figure 14: Z-BUS Configuration Timing.



Z-BUS Read Cycle Timing



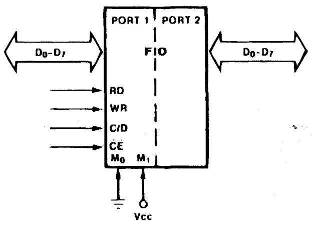
Z-BUS Write Cycle Timing

Before data transactions can take place, the FIO must be taken out of the reset state by writing 00_H to Control Register 0. No other bits in this register can be programmed while clearing the reset bit. The Port 1 CPU then enables Port 2 by setting bit D_0 of the Port 1 Control Register 2. The Port 2 CPU can determine when it is enabled by reading its Control Register 0, which is read as a "floating" data bus if not enabled or as 01_H if enabled.

Z-BUS High - and Low-Byte Programming. The FIOs, like other Z-BUS-compatible peripheral devices, accept byte-serial programming. The CPU must replicate the programming byte sent to $AD_0 - AD_7$ on the $AD_8 - AD_{15}$ address lines, insuring that the same programming byte goes to the high-byte FIO as to the low-byte FIO. The first programming byte is typically sent to Control Register 0. The FIO is then ready to program. Once programming is complete, 16-bit data transfers can take place, with the low-byte FIO transferring the byte on lines $AD_0 - AD_7$, and the high-byte FIO transferring the byte on lines $AD_8 - AD_{15}$.

Non-Z-BUS (nonmultiplexed) **Microprocessor.** The non-Z-BUS (nonmultiplexed) microprocessor configuration interfaces the FIO to a variety of microprocessors that use separate data and address uses, such as the Z80. Figure 15 shows a typical non-Z-BUS configuration. Figure 16 shows non-Z-BUS configuration timing information.

Figure 15 : Non-Z-BUS (nonmultiplexed) Micrpprocessor Configuration.



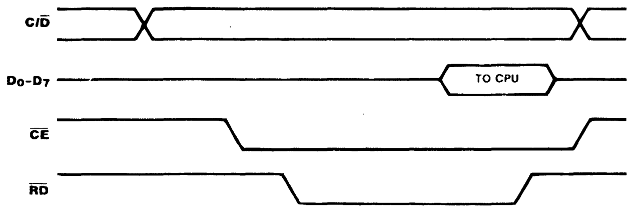
INTERFACING SIGNALS (continued)

Table 7 : Register Addressing.

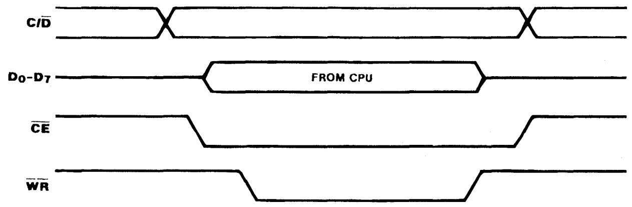
Z-BUS High Byte	A_3	A_2	A_1	A_0
Description				
Control Register 0	0	0	0	0
Control Register 1	0	0	0	1
Interrupt Status Register 0	0	0	1	0
Interrupt Status Register 1	0	0	1	1
Interrupt Status Register 2	0	1	0	0
Interrupt Status Register 3	0	1	0	1
Interrupt Vector Register	0	1	1	0
Byte Count Register	0	1	1	1
Byte Count Comparison Register	1	0	0	0
Control Register 2*	1	0	0	1
Control Register 3	1	0	1	0
Message Out Register	1	0	1	1
Message in Register	1	1	0	0
Pattern Match Register	1	1	0	1
Pattern Mask Register	1	1	1	0
Data Buffer Register	1	1	1	1

* Register is only on Port 1 side

Figure 16 : Non-Z-BUS (nonmultiplexed) Microprocessor Configuration Timing.



Non-Z-BUS Read Cycle Timing.



Non-Z-BUS Write Cycle Timing

INTERFACING SIGNALS (continued)

Signals. In the non-Z-BUS (nonmultiplexed) microprocessor configurations, the following signals control data transactions : Request/Wait, DMA Acknowledge, Read, Write, Chip Enable, Control/Data, Interrupt Acknowledge, Interrupt Enable Input, Interrupt Enable Output, and Interrupt Request.

$\overline{\text{REQ}}/\overline{\text{WAIT}}$ *(Request/Wait, output, 3-state, active Low).* The request signal is sent to a DMA device to begin a DMA transfer, and the Wait (open drain output) signal tells the CPU that the FIO buffer is full (if data is output from the CPU to the FIO) or that the FIO buffer is empty (if data is input to the CPU from the FIO). Bit D_1 of Control Register 1 selects the Request or Wait signals.

$\overline{\text{DACK}}$ *(DMA Acknowledge, input, active Low).* This signal is received by the FIO from a DMA controller. When Low, it forces the next read or write to the FIO to go through the Data Buffer register, regardless of the C/$\overline{\text{D}}$ conditions.

$\overline{\text{RD}}$ *(Read, input, active Low).* The Read signal tells the FIO that the CPU wants to read data from it.

$\overline{\text{WR}}$ *(Write, input, active Low).* The Write signal tells the FIO that the CPU wants to write data to it.

$\overline{\text{CE}}$ *(Chip Enable, input, active Low).* The Chip Enable signal selects the FIO when the CPU needs to carry out a data transaction involving it. The $\overline{\text{CE}}$ signal is usually generated from CPU address and status signals by external logic.

C/$\overline{\text{D}}$ *(Control/Data, input).* The C/D signal is used to distinguish between control and data bytes.

$\overline{\text{INTACK}}$ *(Interrupt Acknowledge, input, active Low),* IEI *(Interrupt Enable In, input, active High),* IEO *(Interrupt Enable Out, output, active High),* $\overline{\text{INT}}$ *(Interrupt Request, output open drain, active Low).* $\overline{\text{INTACK}}$, IEI, IEO, and $\overline{\text{INT}}$ control FIO interrupt operation. $\overline{\text{INT}}$ requests an interrupt from the CPU when Low, $\overline{\text{INTACK}}$ is the interrupt acknowledgement from the CPU. IEI is the input for the daisy-chain, prioritized interrupt enable signal. IEO is the corresponding output for the same signal.

Register Addressing. In the nonmultiplexed configurations, the FIO connects to the system data bus via $D_0 - D_7$, but not to the address bus. The CPU must provide address information on the data bus, and the FIO must distinguish between address and data bytes. The Control/Data (C/$\overline{\text{D}}$) pin does this with the help of the following programming conventions.

Figure 17 shows the register access state diagram. Before programming either port of the FIO, it should be reset, clearing all control registers. Getting out of the reset state is described in the next section.

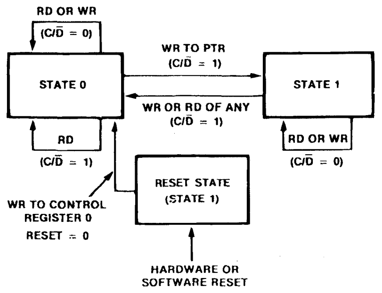
To move data bytes to or from the FIFO buffer, simply lower C/$\overline{\text{D}}$ to 0 (C/$\overline{\text{D}}$ is typically tied to an address pin) and read or write directly into the Data Buffer register. To read or write to all registers (including the Data Buffer register) is a two step operation. First, write the address (C/$\overline{\text{D}}$ = 1) of the register to be accessed into the Pointer Register (state 0 to state 1). The Pointer Register (4 bits) holds the register address. Second, read or write (C/$\overline{\text{D}}$ = 1) to the register addressed previously (state 1 to state 0).

The Pointer Register is not modified or cleared when going from state 1 to state 0. The Pointer Register is only changed by a write (C/$\overline{\text{D}}$ = 1) while in state 0. Consequently, status monitoring can be done while in state 0 by doing a read with C/$\overline{\text{D}}$ = 1.

For example, if the last register address was Control Register 2, then by doing a read with C/$\overline{\text{D}}$ = 1 (stay in state 0) the contents of Control Register 2 will again be read out. This may be useful for doing polling operations.

In the non-Z-BUS configuration, the RJA bit (D_1 of Control Register 0) is set. This specifies that bits 0-3 carry the address of the FIO internal registers and supports such CPUs as the Z80. Table 8 describes the Non-Z-BUS register addressing.

Figure 17 : Register Access State Diagram.



INTERFACING SIGNALS (continued)

Table 8 : Register Addressing.

Non-Z-BUS	D_7 - D_4	D_3	D_2	D_1	D_0
Description					
Control Register 0	x	0	0	0	0
Control Register 1	x	0	0	0	1
Interrupt Status Register 0	x	0	0	1	0
Interrupt Status Register 1	x	0	0	1	1
Interrupt Status Register 2	x	0	1	0	0
Interrupt Status Register 3	x	0	1	0	1
Interrupt Vector Register	x	0	1	1	0
Byte Count Register	x	0	1	1	1
Byte Count Comparison Register	x	1	0	0	0
Control Register 2*	x	1	0	0	1
Control Register 3	x	1	0	1	0
Message Out Register	x	1	0	1	1
Message in Register	x	1	1	0	0
Pattern Match Register	x	1	1	0	1
Pattern Mask Register	x	1	1	1	0
Data Buffer Register	x	1	1	1	1

x = Don't care
* Register is only on Port 1 side

Reset Operation. In non-Z-BUS configuration, the FIO is hardware reset by forcing both $\overline{RD}$ and $\overline{WR}$ Low (normally an illegal condition) or by writing a 1 to the reset bit in Control Register 0. After reset is asserted, the only register that can be read from or written to is Control Register 0 (Control Register 0 will read a 01_H). If C/$\overline{D}$ is 1, the next byte writes into Control Register 0. When in the reset state, a write should not be done when C/$\overline{D}$ is 0. The reset state is exited by writing 00_H when C/$\overline{D}$ is 1. When Port 1 is reset, Port 2 is also reset. All of Port 2's signal lines are floating in this state and all inputs are ignored. If the Port 1 or Port 2 CPU (if enabled) reads any of its registers during reset, they return to 01_H (Control Register 0).

Software Reset. To software reset the FIO, it must be in state 1 and the Pointer register must point to Control Register 0. Table 9 outlines a method for resetting the FIO. The left column explains the software procedures used if the port was already reset. The right column shows the procedures used if the port was not reset. This procedure resets the FIO and then removes the reset making the FIO ready for programming. The table assumes that C/$\overline{D}$ is 1.

In Reset. If the port is already reset, reading it will have no effect and will return 01_H, since the reset bit is set. The Write 00_H command writes 00 into Control Register 0, since writes go to that register only during reset. This write takes the port out of the reset state and into state 0. The Write 01_H loads the Pointer Register with Control Register 1's address. This command places the port in state 1. The next write, Write 00_H, puts 00_H into Control Register 1. This has no effect, since the register was cleared during reset.

Not Reset. If the port is not reset, and hence in an unknown state (state 1 or state 0), a read of any kind (when C/$\overline{D}$ is 1) puts the port into state 0. The Write 00_H loads the Pointer register to Control Register 0's address and puts the port into state 1. The Write 01_H command sets the reset bit in Control Register 0 and puts the port into the reset state. The next write (Write 00_H) goes into Control Register 0, which removes the reset condition. At this point, the FIO registers can be programmed.

2-Wire Handshake I/O. The 2-wire handshake I/O configuration (Port 2 side only) is used to interface a CPU (multiplexed or not) to a single I/O device. The action of the FIO must be acknowledged by the other half of the handshake before the next transaction can take place. Figure 18 shows a typical 2-wire handshake configuration.

INTERFACING SIGNALS (continued)

Table 9 : Non-Z-BUS Software Reset Routines.

In Reset	Software	Not Reset
Reset State (state 1)	Read (any register)	State 0 or State 1
Reset State (state 1) (read = 01)	Write 00	State 0 (read = x)
Removes Reset (state 0)	Write 01	Point to Control Register 0 (state 1)
Point to Control Register 1 (state 1)	Write 00	Resets Port Reset State (state 1)
Write 00 to Control Register 1 (state 0)		Removes Reset (state 0)

Note : All Reads and Writes with C/$\overline{D}$ = 1.

While Reset is not a signal issued or received by the FIO, in CPU-to-I/O transactions the I/O port is reset by the CPU Port 1. When Port 1 is reset, all of Port 2's signal lines are floating in this state, and all inputs are ignored. Before data transactions can take place, the Port 1 CPU must enable Port 2 by setting bit D_0 of the Port 1 Control register 2.

The only register that is accessible on the Port 2 side, in this mode, is the Data Buffer register. All the other control registers are inaccessible.

Signals. In the 2-wire handshake I/O configuration, the FIO port uses Ready For Data/Data Available, Acknowledge Input, Clear, Empty, Full, Data Direction, Output Enable, two output lines, and an input line.

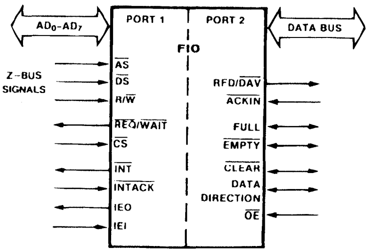
RFD/$\overline{\text{DAV}}$ *(Ready For Data/Data Available, output).* The Ready For Data signal (active High) is issued from the port and alerts external devices that it is ready to receive data. Data Available (active Low) indicates that the port has data to transmit.

$\overline{\text{ACKIN}}$ *(Acknowledge Input, input, active Low).* The Acknowledge Input signal, received by the FIO, notifies the CPU that transmitted data has been accepted by the external device or that data is valid on the bus.

$\overline{\text{CLEAR}}$ *(input or output, active Low),* EMPTY *(output/input, open drain, active High),* FULL *(output/input, open drain, active High).* These signals pertain to the condition of the FIO RAM buffer when the FIO performs I/O data transactions. Clear flushes all data from RAM ; when active (High), Empty and Full indicate that the FIFO buffer is empty or full during CPU-to-I/O transactions. The Buffer Full and Buffer Empty bits (D_0 and D_4 of Interrupt Status Register 3, respectively) are set when these pins are High.

DATA DIR *(Data Direction, input or output, active High).* The Data Direction signal allows the CPU or the I/O device to change the direction of data transactions. The control of data direction is programmed by the Port 1 CPU, via the Control register. When the Data Direction pin (pin 34) is Low, Port 2 is in output handshake configuration. Similarly, when the Data Direction pin is High, Port 2 is in input handshake configuration.

Figure 18 : Wire Handshake I/O Configuration.



$\overline{\text{OE}}$ *(Output Enable, input, active Low).* When Output Enable is Low, data is propagated into Port 2 data lines. When OE is High, the data lines are at high impedance.

IN_0, IN_2, and OUT_3 are three signal lines that can be used as simple I/O bits.

Once the configuration is specified, two bits in the Port 1 Control Register 2 control the operation of Port 2. If D_0 is set, Port 2 is enabled. If D_1 is set, the Port 2 handshake signals are enabled.

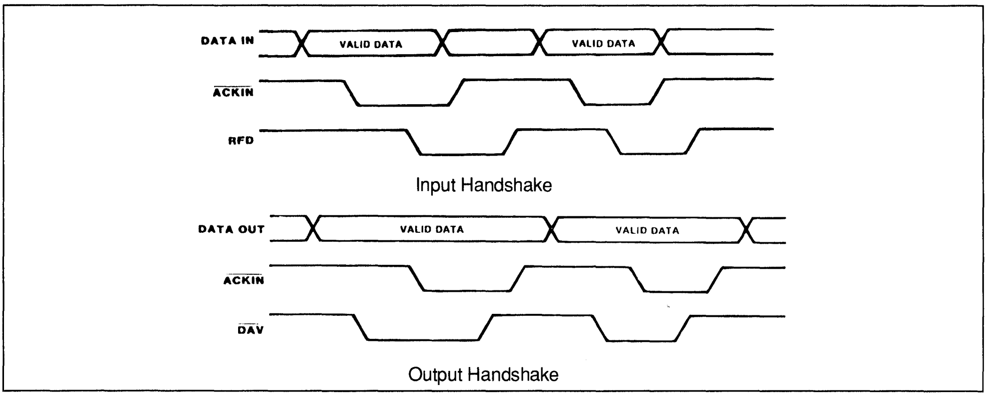
2-Wire Handshake Operation. The FIO's 2-wire handshake is an interlocked handshake in which each action is acknowledged by the other device. This handshake protocol is compatible with the Z8, CIO, FIFO, UPC, and other commercially available parts. Figure 19 gives 2-wire handshake timing information. For both input and output handshake $\overline{\text{ACKIN}}$ must be High when the handshake is enabled (D_1 of Control Register 2 is set).

Output Handshake. For output handshake (Data Direction = 0) the $\overline{\text{DAV}}$ signal will go Low, indicating that data is available. This can occur only when $\overline{\text{ACKIN}}$ is High and there is data in the Data Buffer register. The $\overline{\text{DAV}}$ signal will stay Low until the $\overline{\text{ACKIN}}$ signal goes Low, indicating that the data has been accepted. The FIO's response to $\overline{\text{ACKIN}}$ Low is to bring the $\overline{\text{DAV}}$ signal High. At some later time the receiving device will bring $\overline{\text{ACKIN}}$ back High, in-

SGS-THOMSON MICROELECTRONICS

INTERFACING SIGNALS (continued)

Figure 19 : 2-Wire Handshake I/O Configuration Timing.



dicating that it is ready for the next data byte. This process continues until the FIFO buffer is empty, the DAV will go High and remain so.

The Enable Handshake bit (D_1 of Control Register 2), when Low, forces the DAV signal High and internally forces ACKIN High. Care should be taken when disabling the handshake operation if the ACKIN signal is Low. Putting this bit Low at the wrong time may violate the handshake interlock and cause improper operation. A known time when this bit can go Low when the FIFO buffer is empty.

Input Handshake. For input handshake (Data Direction = 1), the RFD signal starts out High, indicating that the FIFO buffer is ready for incoming data transfers. Next ACKIN will go Low indicating that data is available on the data bus. The FIO responds by bringing RFD Low, signaling the acceptance of the data. Some time after RFD is Low, $\overline{\text{ACKIN}}$ will again go High. When the FIFO buffer is again ready for a new byte of data, it puts RFD High. This process continues until the FIFO buffer is full, at which time the RFD will go Low and remain so.

The Enable Handshake bit, when Low, forces RFD High and internally forces $\overline{\text{ACKIN}}$ High. Care should be taken when disabling the handshake operation if the $\overline{\text{ACKIN}}$ signal is Low. Putting this bit Low at the wrong time may violate the handshake interlock and cause improper operation. A known time when this bit can go Low when the FIFO buffer is empty.

In CPU-to-I/O transactions, exercise caution when changing data direction and clearing the FIFO buffer. Since DATA DIRECTION and CLEAR are programmed in the same register, they both can be changed during a single write to Control Register 3. Putting the Clear bit Low and changing the Data Direction bit is okay. Putting the Clear bit High and changing the Data Direction bit may cause improper operation. The Data Direction bit should only be changed when the Clear bit is Low or going Low.

The input and output lines, IN_0, OUT_1, and OUT_3 are, respectively, an input to bit D_0 of Port 2 Control Register 3 and outputs from bits D_1 and D_3. D_0 is a read-only bit. These three signal lines are used as simple I/O bits.

Full and Empty Operation. Both the Full and Empty pins are used as bidirectional signals. (See figures 20 and 21). As open drain output lines they can be wire-ANDed with other FIFOs or FIOs to give system status. As inputs, Full and Empty are used to set their respective IPs and also to show the status of the pins.

Figure 20 : Full Pin.

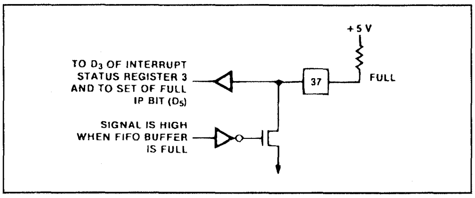
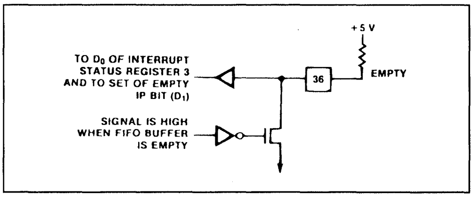

+5 V
TO D_3 OF INTERRUPT STATUS REGISTER 3 AND TO SET OF FULL IP BIT (D_5)
37
FULL
SIGNAL IS HIGH WHEN FIFO BUFFER IS FULL

Figure 21 : Empty Pin.


+5 V
TO D_0 OF INTERRUPT STATUS REGISTER 3 AND TO SET OF EMPTY IP BIT (D_1)
36
EMPTY
SIGNAL IS HIGH WHEN FIFO BUFFER IS EMPTY

INTERFACING SIGNALS (continued)

Figure 22 : 3-Wire Handshake I/O Configuration.

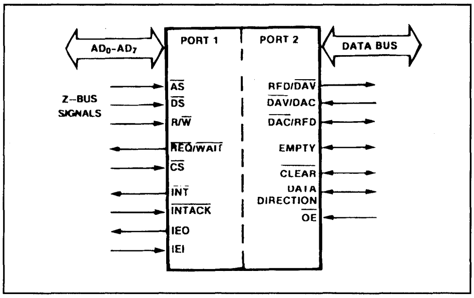


Table 10 : Full and Empty Status.

Number of Bytes in FIFO	Empty	Full
0	High	Low
1-127	Low	Low
128	Low	High

Table 10 shows the Full and Empty signals with only an external pull-up on each pin.

3-Wire Handshake I/O. The 3-wire hand-shake I/O configuration is designed to interface a CPU to many I/O ports simultaneously. Figure 22 shows a typical 3-wire handshake I/O configuration.

While Reset is not a signal issued or received by the FIO, in CPU-to-I/O transactions the I/O port is reset by the action of the CPU port. When Port 1 is reset, Port 2 is also reset. All of Port 2's signal lines are floating in this state, and all inputs are ignored. Before data transactions can take place, the Port 1 CPU must enable Port 2 by setting bit D_0 of the Port 1 Control register 2.

Signals. In the 3-wire handshake I/O configuration, the FIO port uses the following signals : Ready For Data/Data Available, Data Available/Data Accepted, Data Accepted/Ready For Data, Clear, Empty, Data Direction, Output Enable, two output lines, and an input line.

RFD/$\overline{\text{DAV}}$. *(Ready For Data/Data Available, output).* When Port 2 is configured as an input handshake, the Ready For Data signal is issued from the port and, on its rising edge, alerts external devices that it is ready to receive data. When configured as a data output handshake, the falling edge of the Data Available signal indicates tha the port has data to transmit.

DAV/DAC *(Data Available/Data Accepted, input).* In the input handshake configuration, Data Available indicates that data is ready for transmission to the FIO ; in the output handshake configuration, the rising edge of Data Accepted indicates that peripheral devices have received the data.

DAC/RFD *(Data Accepted/Ready For Data, output/input).* The DAC signal is an output when Port 2 is configured as an input handshake and the RFD signal is an input when Port 2 is configured as an output handshake.

$\overline{\text{CLEAR}}$ *(input or output, active Low)*, EMPTY *(output/input open drain, active High).* These two signals pertain to the condition of the FIO RAM buffer when the FIO performs I/O data transactions. $\overline{\text{CLEAR}}$ flushes all data from RAM ; EMPTY indicates, when active (High), that the FIFO buffer is empty during CPU-to-I/O transactions. The buffer empty bit (D_0 of Interrupt Status Register 3) is set when pin 36 is High.

DATA DIR *(Data Direction, input or output, active High).* The Data Direction signal allows the CPU or the I/O device to change the direction of data transactions. The direction is controlled by the Port 1 CPU, via Control Register 3. When the Data direction pin (pin 34) is Low, Port 2 is defined as having an output (source) handshake. When the Data Direction pin is High, Port 2 is defined as having an input (acceptor) handshake.

$\overline{\text{OE}}$ *(Output Enable, input, active Low).* When Output Enable is active (Low), output signals are propagated into Port 2 data lines. When OE is inactive (High), the data lines are at high impedance.

IN_0, IN_2, and OUT_3 are three signal lines available in the I/O modes.

3-Wire Handshake Operation. This configuration is like the handshake protocol used in the IEEE-488 standard. Using this handshake, the lines of many I/O ports can be bused together with external open-drain output drivers to signal when all the ports have accepted data and all are ready for data. Since the data direction of Port 2 can be changed under software control, bidirectional transfers can be performed. When interfacing to a bus such as the IEEE-488, external drivers must be used to meet the IEEE-488 bus specifications. Figure 23 shows the 3-wire handshake I/O timing information.

Output Handshake. For output handshake (Data Direction = 0), the FIO uses one output $\overline{\text{DAV}}$ *(Data Valid)* and two inputs RFD *(Ready For Data)* and DAC *(Data Accepted)* Before any bytes are written into the FIO buffer, the handshake must be enabled *(set D_1 of Control Register 1).* The handshake begins when the RFD input goes High (with DAC Low) indicating that the external device (s) are ready for data. When there is data in the Data Buffer register,

INTERFACING SIGNALS (continued)

the FIO puts the $\overline{DAV}$ signal Low indicating that the data is valid on the bus. The external device (s) now indicate that they are not ready for data by putting RFD Low. When the device (s) have accepted the data, DAC goes High indicating that the data on the bus is no longer needed. The FIO responds by bringing $\overline{DAV}$ High and indicating that the data on the bus may not be valid.

This handshake operation continues until the external device (s) cannot accept any more data ; at this point RFD stays Low, or the FIFO buffer is empty, in which case $\overline{DAV}$ stays High.

When the Enable Handshake bit is Low the $\overline{DAV}$ output is forced High.

Input Handshake. For input handshake (Data Direction = 1) the FIO uses two outputs, RFD (Ready For Data) and DAC (Data Accepted), and one input, $\overline{DAV}$ (Data Valid). The handshake starts by the FIO having RFD High, indicating that it is ready to accept data (DAC is Low). The external device will indicate that data is valid on the bus by forcing $\overline{DAV}$ Low. The FIO responds by bringing RFD Low and then putting DAC High, indicating that it is not ready for data and the data has been accepted. Some time later $\overline{DAV}$ will go High, indicating that the data on the bus is no longer valid. The FIO responds to this condition by bringing DAC back Low. The handshake operation continues until the external device has no more data to send, in which case it leaves $\overline{DAV}$ High, or the FIFO buffer is full, in which case RFD stays Low.

When the Enable Handshake bit is Low, RFD is forced High and DAC is forced Low.

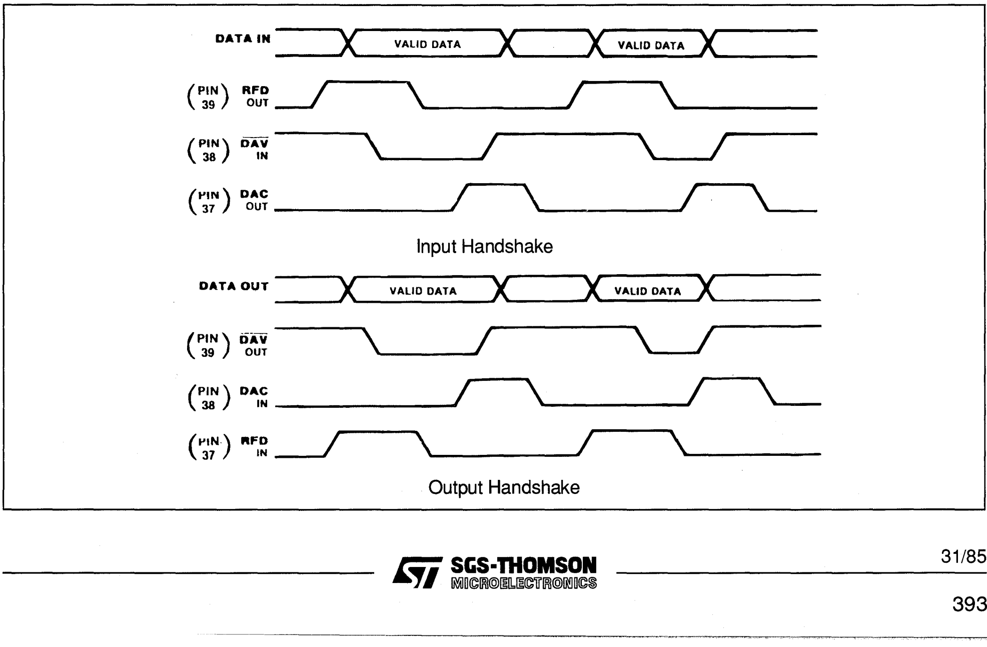
This configuration can be used only with Port 2 and must interface to the I/O device. Therefore, a CPU cannot directly program Port 2 via the Port 2 pins ; Port 2 operation is specified by the Port 1 internal registers. Bits D_2 and D_3 of Port 1 Control Register 0 define the Port 2 configuration. Resetting D_2 and setting D_3 enables Port 2 in the 3-wire handshake configuration.

Once the configuration is specified, two bits in the Port 1 Control Register 2 control the operation of Port 2. If D_0 is set, Port 2 is enabled. If D_1 is set, the Port 2 handshake signals are enabled.

In CPU-to-I/O transactions, care should be taken when changing data direction and clearing the FIFO buffer. Since DATA DIRECTION and CLEAR are programmed in the same register, they both can be changed during a single write to Control Register 3. Putting the Clear bit Low and changing the data direction bit is okay. Putting the Clear bit High and changing the data direction bit may cause improper operation. The data direction bit should only be changed when the Clear bit is Low or going Low.

The input and output lines (IN_0, OUT_1, and OUT_3) are, respectively, an input to bit D_0 of Port 2 Control register 3 and outputs from bits D_1 and D_3. D_0 is a read-only bit. These three signal lines are used for communication between FIOs and FIFOs in a networking environment.

Figure 23 : 3-Wire Handshake I/O Configuration Timing.



INTERFACING SIGNALS (continued)

$\overline{\text{WAIT}}$ OPERATION

Output $\overline{\text{WAIT}}$. When data is output by the CPU (Data Direction = 0), the REQ/$\overline{\text{WAIT}}$ pin is active (Low) only when the FIFO buffer is full, the chip is selected, and the FIFO buffer is addressed. $\overline{\text{WAIT}}$ goes inactive when the FIFO buffer is not full.

Input WAIT. When data is input by the CPU (Data Direction = 1), the REQ/$\overline{\text{WAIT}}$ pin becomes active (Low) only when the FIFO buffer is empty, the chip is selected, and the FIFO buffer is addressed. $\overline{\text{WAIT}}$ goes inactive when the FIFO buffer is not empty.

The release of the FIO $\overline{\text{WAIT}}$ signal is asynchronous with respect to the CPU $\overline{\text{WAIT}}$ input. The circuit shown in figure 24 synchronizes the FIO $\overline{\text{WAIT}}$ signal with the expected Z8000 CPU WAIT input.

REQUEST (DMA) OPERATION

The FIO works particularly well with DMA devices in both Z-BUS and non-Z-BUS (nonmultiplexed) microprocessor modes. DMA operation can take place on both sides of the FIO simultaneously ; however, DMA operation cannot take place on the Port 2 side when the Port 2 side is in an I/O configuration.

The FIO supports two types of DMA operation, "flyby" and "sequential", both compatible with devices such as the Z80 DMA controller. In the flyby operation, data transfers occur every machine cycle on either on both sides of the FIO. The DMA controller issues a memory address with appropriate control signals and $\overline{\text{DACK}}$ or $\overline{\text{DMASTB}}$; the FIO receives or sends the byte strobed into the bus. The byte never passes through the DMA, it merely "flies by" between the memory and the FIO. In sequential operation, the DMA controller reads the byte from memory and then writes it to the FIO buffer or vice versa, addressing the Data Buffer register as a peripheral device. In this operation $\overline{\text{DACK}}$ or $\overline{\text{DMASTB}}$ should be tied to V_{CC}. "Flyby" is not a mode or a bit that is set in the FIO. To use flyby the Request operation must be enabled and the $\overline{\text{DMASTB/DACK}}$ pin is used.

The Request signal is inactive until the Clear bit is inactive (High). For example, when configuring the FIO for DMA operation, bits D_0 and D_1 in the controlling CPU's Control Register 1 can be set while the Clear bit is active without triggering a valid Request signal and initiating the transfer. Once the Clear bit goes inactive (Low), the DMA transfer into the FIO can begin, since the FIFO buffer is empty.

Z-BUS Flyby Operation. In figures 25 and 26, which show Z-BUS FIO-to-memory and memory-to-FIO transfers, the $\overline{\text{DMASTB}}$ signal strobes data to and from the FIO buffer. $\overline{\text{AS}}$ pulses Low to gate the memory address from the DMA into the Address/Data bus. Once $\overline{\text{AS}}$ goes High and $\overline{\text{DMASTB}}$ goes Low, data is read from or written into the FIO buffer directly without having to write the Data Buffer address during each read. $\overline{\text{DMASTB}}$ acts as a read or write signal ; the data direction bit specifies the direction of the transfer (the R/$\overline{\text{W}}$ pin is ignored). There is no effect if the DMA tries to read data when the FIO expects input, or vice versa.

The $\overline{\text{CS}}$ signal is not ignored by the FIO and therefore must be kept invalid (High) during the DMA transfer. This is normally acccomplished by default since the DMA device is addressing memory.

Non-Z-BUS Flyby Operation. In figure 27 and 28, showing non-Z-BUS FIO-to-memory and memory-to-FIO transfers, the $\overline{\text{DACK}}$ signal generated by the DMA device acknowledges the FIO's DMA request. After $\overline{\text{DACK}}$ goes true, the DMA device places the memory address on the memory address bus, and data is gated from memory into the data bus. $\overline{\text{DACK}}$ forces the next read or write to the FIO into the FIO buffer directly without having to write the Data Buffer address during each read (the C/$\overline{\text{D}}$ pin is ignored). The data direction bit specifies the direction of the transfer. There is no effect if the DMA tries to read data when the FIO expects input, or vice versa. The $\overline{\text{CE}}$ signal is not ignored by the FIO and therefore must be kept invalid (High) during the DMA transfer. This is normally accomplished by default since the DMA device is addressing memory.

Request Hysteresis Operation. The FIO has a feature that enhances DMA transactions in both flyby and sequential operation. Figures 29 and 30 are request hysteresis diagrams. These diagrams describe the use of the Byte Count Compare register during DMA transactions and demonstrate how DMA-to-FIO transfers can proceed. To engage this feature, set bit D_2 of Control Register 1.

REQUEST (DMA) OPERATION (continued)

When data is written into the FIO by the DMA controller, the FIO initiates the cycle by issuing an active Request signal. (Refer to figure 29). $\overline{REQ}$ stays active (Low). The FIO is ready to receive bytes from the bus. $\overline{REQ}$ remains active until the buffer is full. It then goes High and remains inactive while the CPU or I/O device reads the bytes from the FIFO buffer until the number of bytes in the buffer is equal to the value programmed into the Byte Count Compare register. $\overline{REQ}$ goes Low once more, and the sequence begins again.

Figure 24 : FIO-to-CPU WAIT Synchronization Circuit.

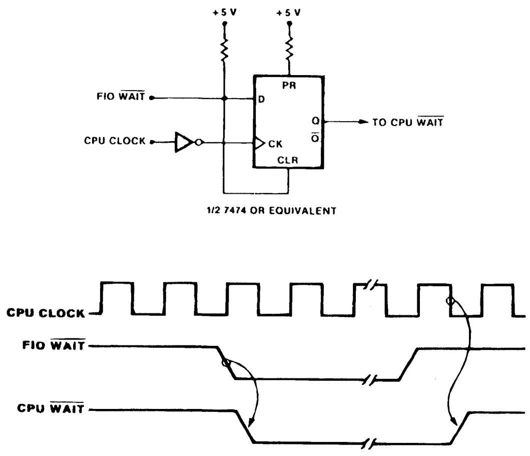
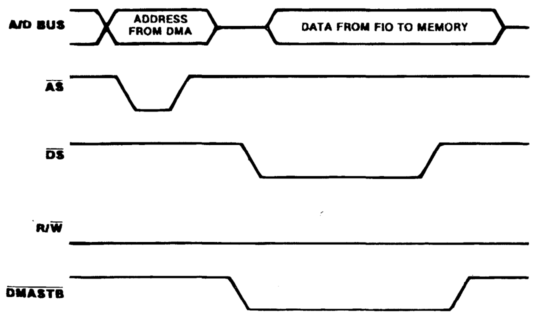


Figure 25 : Z-BUS FIO-to-Memory Data Transactions.



REQUEST (DMA) OPERATION (continued)

Figure 26 : Z-BUS Memory-to-FIO Data Transactions.

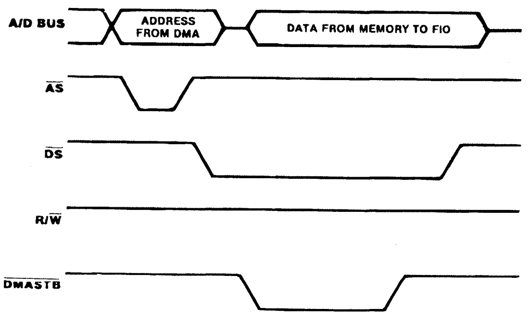
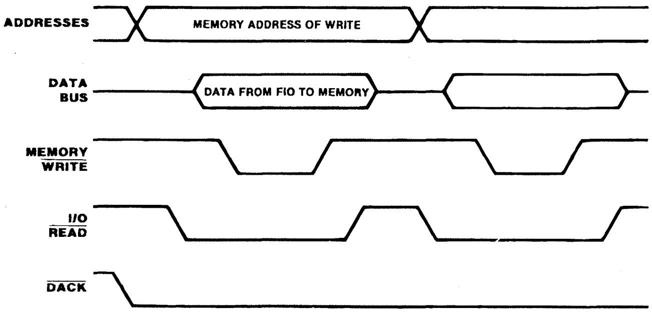


Figure 27 : Non-Z-BUS FIO-to-Memory Data Transactions.



REQUEST (DMA) OPERATION (continued)

Figure 28 : Non-Z-BUS Memory-to-FIO Data Transactions.

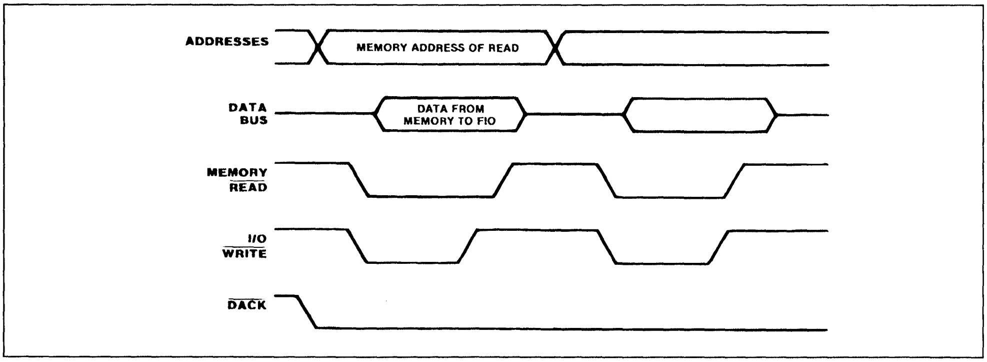
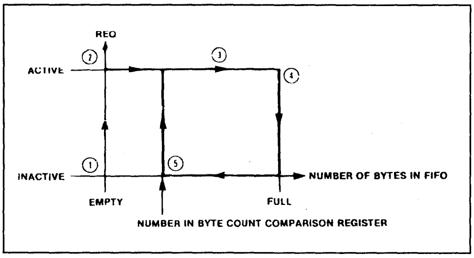
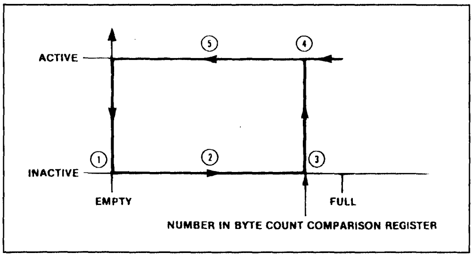


Figure 29 : Request Hysteresis Diagram : Write to FIO.



Notes :
1. FIO empty.
2. REQUEST enabled, FIO requests DMA transfer.
3. DMA transfers data into the FIO.
4. FIFO full, REQUEST inactive.
5. The FIFO empties from the opposite port until the number of bytes in the FIFO buffer is the same as the number programmed in the Byte Count Comparison register.

Figure 30 : Request Hysteresis Diagram : Read from FIO.



Notes :
1. FIFO empty.
2. CPU/DMA fills FIFO buffer from the opposite port.
3. Number of bytes in FIFO buffer is the same as the number of bytes programmed in the Byte Count Comparison register.
4. REQUEST goes active.
5. DMA transfers data out of FIO until it is empty.

When data is read from the FIO by the DMA controller, the FIO receives bytes from the CPU. (Refer to figure 30). The Request line is inactive until the number of bytes in the buffer is equal to the value programmed in the Byte Count Compare register. $\overline{\text{REQ}}$ then goes Low and remains Low until the DMA controller has emptied the buffer. Then $\overline{\text{REQ}}$ goes High and the sequence starts over again. If both ports are operating in CPU modes using DMA, this feature allows DMA operations to continue independently, triggering the Request signals from the two different Byte Count Compare register values.

If both ports are reading and writing at approximately the same rate, the buffer becomes a scratchpad memory for continuous DMA transfers. In this way, the length of the DMA block transfers can be much longer than the FIFO's buffer.

INTERRUPT OPERATION

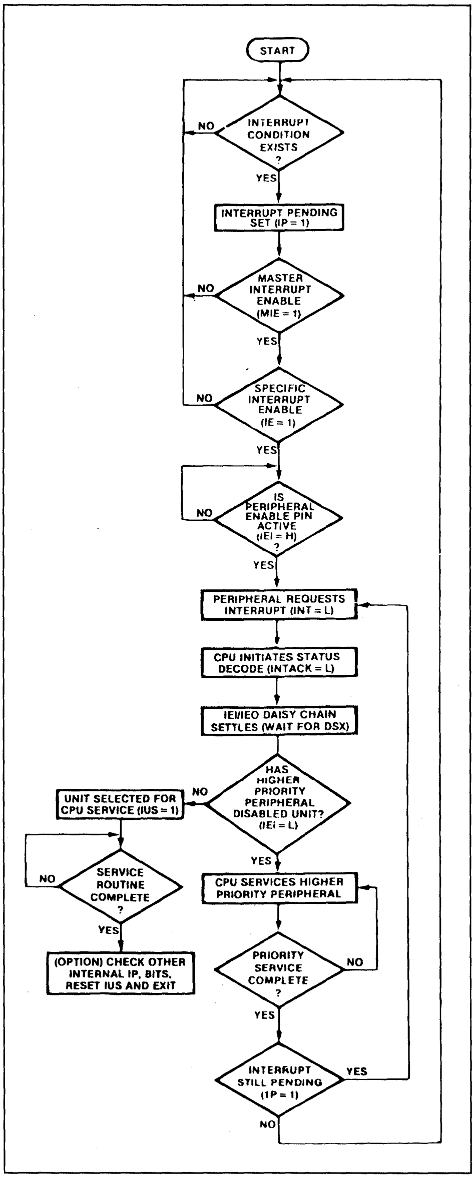
The FIO generates interrupt requests when programmed to do so and when one of seven specified conditions occurs. A complete interrupt cycle consists of an interrupt request followed by an interrupt acknowledge transaction. The CPU acknowledges interrupt request, according to the Z-BUS daisy-chain priority interrupt protocols. The FIO then places an address vector on the bus, providing the starting location of either an interrupt service routine or an index table into a group of such routines. The reason for the interrupt is encoded into the address vector, which becomes a pointer to specialized interrupt service routines.

Figure 31 is a flowchart of a peripheral interrupt sequence and explains the action within a single interrupting device. As the flowchart indicates, the peripheral inherently checks the status of all IP bits only if IEI is active. The IEI line is inactive only if the IEO of a higher-priority device is Low.

The interrupt Status registers specify whether or not interrupts will be generated during true interrupt conditions. If such interrupt conditions occur and the appropriate interrupt is enabled, the FIO communicates the interrupt request to the CPU.

Interrupt Priority. The seven prioritized conditions that can trigger the generation of an interrupt request follow in order of priority : Message Pending, Change in Data Direction, Pattern Match, Byte Count Compare, Overflow/Underflow Error Buffer Full, and Buffer Empty conditions. Each interrupt condition has three corresponding control bits in the Interrupt Status registers : Interrupt Enable bit (IE) enables interrupt generation, Interrupt Pending bit (IP) alerts the CPU that an interrupt is waiting for service, and Interrupt Under Service bit (IUS) shows if the interrupt is being serviced.

Figure 31 : Interrupt Flowchard.


START
INTERRUPT CONDITION EXISTS ?
NO
YES
INTERRUPT PENDING SET (IP = 1)
MASTER INTERRUPT ENABLE (MIE = 1)
NO
YES
SPECIFIC INTERRUPT ENABLE (IE = 1)
NO
YES
IS PERIPHERAL ENABLE PIN ACTIVE (IEI = H) ?
NO
YES
PERIPHERAL REQUESTS INTERRUPT (INT = L)
CPU INITIATES STATUS DECODE (INTACK = L)
IEI/IEO DAISY CHAIN SETTLES (WAIT FOR DSX)
HAS HIGHER PRIORITY PERIPHERAL DISABLED UNIT? (IEI = L)
NO
YES
UNIT SELECTED FOR CPU SERVICE (IUS = 1)
SERVICE ROUTINE COMPLETE ?
NO
YES
(OPTION) CHECK OTHER INTERNAL IP. BITS. RESET IUS AND EXIT
CPU SERVICES HIGHER PRIORITY PERIPHERAL
PRIORITY SERVICE COMPLETE ?
NO
YES
INTERRUPT STILL PENDING (IP = 1)
YES
NO

INTERRUPT OPERATION (continued)

Figure 32 : Z-BUS Interrupt Acknowledge Timing.

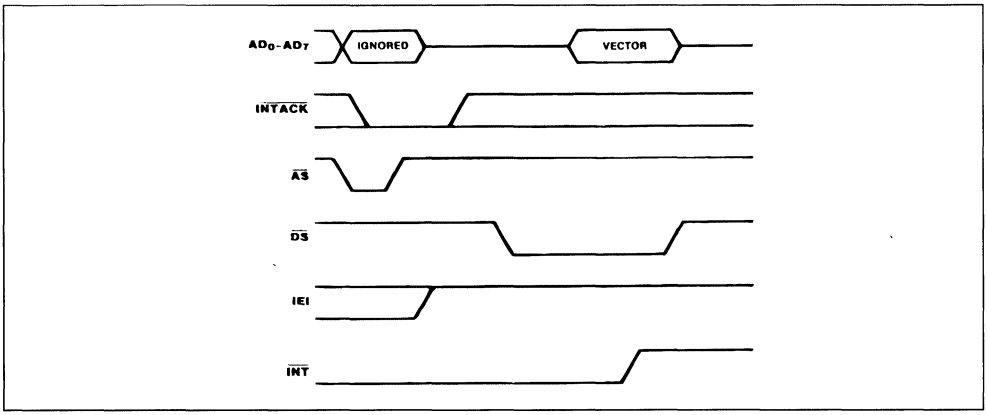
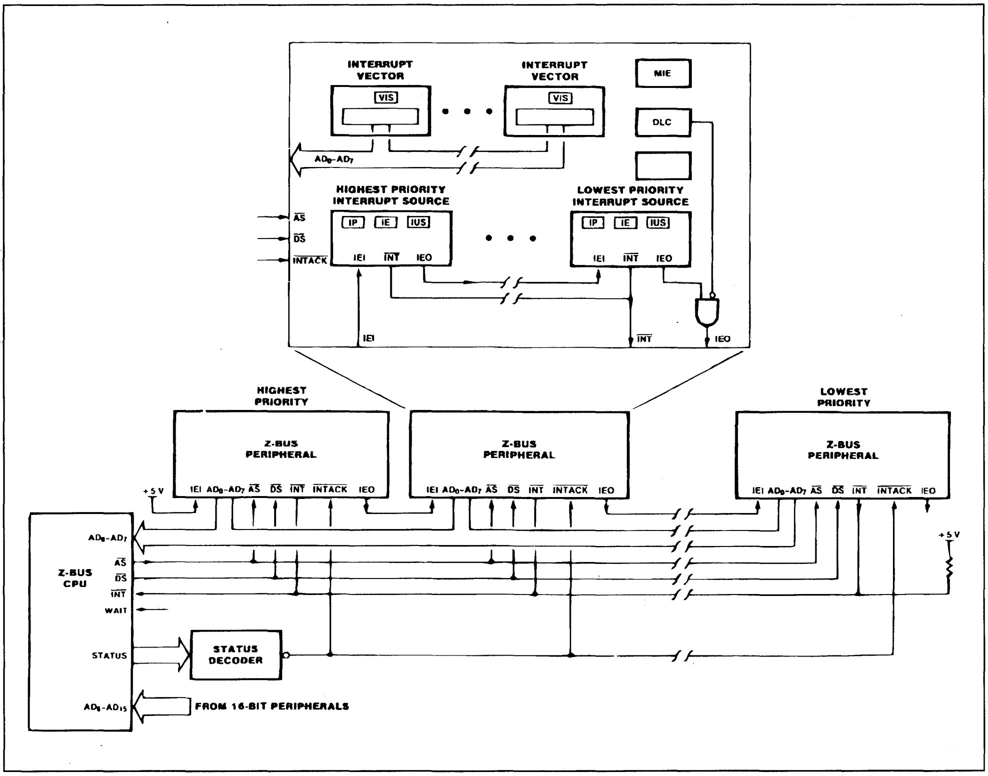


Figure 33 : Z-BUS Interrupt Arbitration.



SGS-THOMSON MICROELECTRONICS

INTERRUPT OPERATION (continued)

Setting and Clearing of Interrupt Control Bits. The setting and clearing of the IP, IUS, and IE bits is done under the command structure shown in figure 34. In this method the individual bits in a register are not set or cleared directly. This guards against the accidental clearing of an IP bit during a write and the possible loss of an interrupt request. Note the 3-bit code to set the message Interrupt Under Service bit shown beneath the Interrupt Status Register 0 drawing in figure 10. To set this bit, write 01000000 (40_H) to this register. This write does not directly set bit D_6, however. It sets bit D_7, shown in figure 10 as the message IUS bit. When Interrupt Status Register 0 is read following the write, it reads as 10000000 (80_H). These 3-bit groups, then, are described as codes that set or clear the Interrupt Status bits, rather than setting the bit in the corresponding register position. The same format is used with all four Interrupt Status registers.

Refer to the Interrupt Status registers shown in figure 10. The IUS and IP bits can be set two ways-when the condition itself occurs or by program control. These bits are cleared by program control. The IE bit can be set and cleared only by writing to them.

Z-BUS Interrupt Operation. Refer to figures 32 and 33. The FIO generates an interrupt request by lowering the $\overline{INT}$ line, only if such interrupt requests are enabled (IE is 1, MIE is 1), it has an Interrupt Pending (IP = 1), it does not have an Interrupt Under Service (IUS is 0), no higher-priority interrupt is being serviced (IEI is 1), and no interrupt acknowledge transaction is taking place (as indicated by $\overline{INTACK}$ High at the last rising edge of $\overline{AS}$). IEO is not pulled down by the FIO at this time ; IEO continues to follow IEI until an interrupt acknowledge transaction occurs.

Some time after $\overline{INT}$ has been pulled Low, the CPU initiates an interrupt acknowledge transaction. Between the rising edge of $\overline{AS}$ and the falling edge of $\overline{DS}$, the IEI/IEO daisy chain settles. Any Z-BUS peripheral with one of its Interrupts Pending (IP is 1) or one of its Interrupts Under Service (IUS is 1) holds its IEO line Low ; all others make IEO follow IEI.

When $\overline{DS}$ falls, only the highest priority interrupt source with a pending interrupt (IP is 1) has its IEI input High, its IE bit set to 1, and its IUS bit set to 0. This is the interrupt source being acknowledged, and at this point it sets its IUS bit to 1. If its NV bit is 0, the FIO identifies itself by placing its interrupt vector from the Interrupt Vector register on Address/Data lines $AD_0 - AD_7$. If NV is 1, the FIO's $AD_0 - AD_7$ lines remain floating, allowing external logic to supply a vector. (All Z-BUS interrupts require a vector to identify the requesting device).

If the FIO's VIS is 1, the vector also contains status information, coded into bits D_1 – D_3, which further describe the source of the interrupt within the FIO's logic. If VIS is 0, the vector held in the FIO is output without status included (base vector). The bit codes are given in Table 11. IPs are set by an AS following the event.

Figure 34 : IP. IUS and IE Command Code.

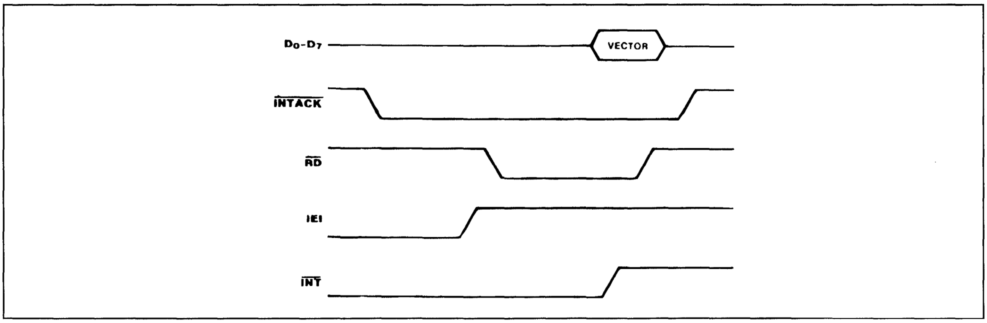
0	0	0	NULL CODE
0	0	1	CLEAR IP & IUS
0	1	0	SET IUS
0	1	1	CLEAR IUS
1	0	0	SET IP
1	0	0	CLEAR IP
1	1	0	SET IE
1	1	1	CLEAR IE

Table 11 : Interrupt Status Bit Codes.

Bits 3	2	1	Codes
1	1	1	Received Message in Message In Register
1	1	0	Change in Data Transaction Direction
1	0	1	Valid Pattern Match
1	0	0	Valid Byte Count Compare
0	1	1	Overflow or Underflow Error
0	1	0	Buffer Full
0	0	1	Buffer Empty
0	0	0	No Interrupts Pending

INTERRUPT OPERATION (continued)

Figure 35 : Non-Z-BUS Interrupt Acknowledge Cycle.



Non-Z-BUS Interrupt Operation. Figure 35 shows the complete non-Z-BUS interrupt acknowledge cycle timing.

When in non-Z-BUS configurations, the IP bit is not set while the port is in state 1. Any interrupt condition that occurs (such as a message received) does not set the corresponding IP bit. Thus, to minimize interrupt latency, the FIO should stay in state 0. The IP bit is set when the port returns to State 0.

The FIO generates an interrupt request (IP is 1 : Interrupt Pending) by lowering the INT line, only if such interrupt requests are enabled (IE is 1, MIE is 1), it has an interrupt pending (IP = 1), it does not have an Interrupt Under Service (IUS is 0), no higher-priority interrupt is being serviced (IEI is 1) and no interrupt acknowledge transaction is taking place. IEO is not pulled down by the FIO at this time ; IEO continues to follow IEI until an interrupt acknowledge transaction occurs.

Some time after $\overline{INT}$ has been pulled Low, the CPU initiates an interrupt acknowledge transaction. Between the falling edge of $\overline{INTACK}$ and the falling edge of $\overline{RD}$, the IEI/IEO daisy chain settles. Any peripheral with one of its Interrupts Pending (IP is 1) or one of its Interrupts Under Service (IUS is 1) holds its IEO line Low ; all others make IEO follow IEI.

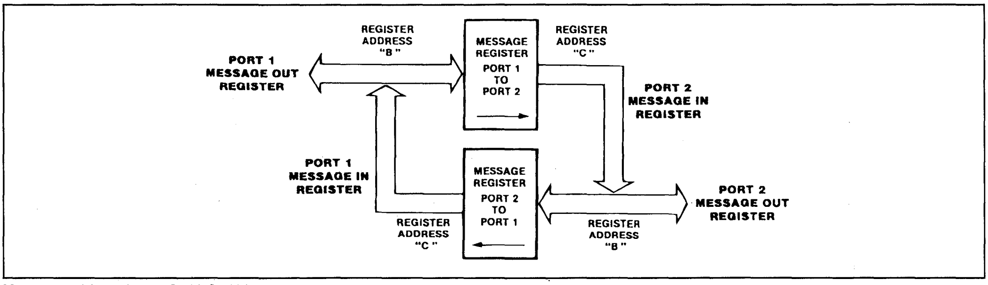
When RD falls, only the highest priority interrupt source with a pending interrupt (IP is 1) has its IEI input High, its IE bit set to 1, and its IUS bit set to 0. This is the interrupt source being acknowledged, and at this point it sets its IUS bit to 1. If its NV bit is 0, the FIO identifies itself by placing its interrupt vector from the Interrupt Vector register on data lines $D_0 - D_7$. If NV is 1, the FIO's $D_0 - D_7$ lines remain floating, allowing external logic to supply a vector.

If the FIO's VIS is 1, the vector also contains status information, coded into bits D_1-D_3, which further describe the source of the interrupt within the FIO's logic. If VIS is 0, the vector held in the FIO is output without status included. The bit codes are in Table 11.

MESSAGE REGISTER OPERATION

The two message registers provide a "mailbox" that is a means for two CPUs use to communicate with each other around the FIO's buffer. Each port's CPU is alerted when the opposite port's CPU sends a message.

Figure 36 : Message Register Operation.



REGISTER ADDRESS "B"
MESSAGE REGISTER PORT 1 TO PORT 2
REGISTER ADDRESS "C"
PORT 1 MESSAGE OUT REGISTER
PORT 2 MESSAGE IN REGISTER
PORT 1 MESSAGE IN REGISTER
MESSAGE REGISTER PORT 2 TO PORT 1
PORT 2 MESSAGE OUT REGISTER
REGISTER ADDRESS "C"
REGISTER ADDRESS "B"

Note : usable only for CPU/CPU interface.

MESSAGE REGISTER OPERATION

Figure 36 describes the message flow between two message registers. When Port 1's CPU writes to the Message Out register which is also Port 2's Message In register, Port 2's Message Interrupt Pending bit is set. If the Message Interrupt bit is enabled, Port 2's CPU is interrupted. Port 2's Message IP bit can be read from the Port 1 side (D_5 of Control Register 1) ; when the Port 2 CPU reads the data from its Message In register, the Port 2 IP is cleared. The Port 1 CPU can therefore tell when the message has been read and can now send another message or follow whatever protocol that it set up between the two CPUs. The same transfer can be made from Port 2's CPU to the Port 1 CPU.

ERROR OPERATION

In CPU configurations, the FIO detects overflow or underflow errors. Overflow is defined as performing a write to the Data Buffer register when the FIFO buffer is full, with the data direction bit at 0. This write is ignored, and the Byte Count register does not change value.

Underflow is defined as a read of the Data Buffer register when the FIFO buffer is empty, with the data direction bit set to 1. The contents of such a read are undefined and the Byte Count register does not change value.

Since the error conditions are mutually exclusive, one error Interrupt Pending bit (D2 of Interrupt Status Register 2) serves for both error conditions, with bit D_0 as 1 indicating an underflow error and bit D_4 as 1 indicating an overflow error. When either condition occurs, the error IP bit is set, regardless of the state of the error Interrupt Enable bit, and the appropriate Over/Underflow bit is set as well. The error IP bit is reset under program control, and the Over/Underflow bits are reset simultaneously.

If the Wait function is enabled, no error conditions should occur.

The error logic detects errors only for its own CPU. For example, if Port 1 is receiving data (Data Direction bit is 0), only overflow errors will be detected. If Port 2 has an underflow error, it will not set Port 1's error IP bit.

PATTERN MATCH OPERATION

The Pattern Match register contains a byte to compare with the byte in the data buffer. The Pattern Mask register contains a bit pattern used to mask bits used in the comparison.

Pattern Match Register. The Pattern Match register (figure 9) contains a byte for comparison with the byte in the Data Buffer register. As each byte in a data transaction passes through the Data Buffer register, it is compared with the value programmed in the Pattern Match register. Upon a true match, an interrupt can be generated if enabled in Interrupts Status register 1. One or more bits in the Pattern Match register can be masked by the value in the Pattern Mask register.

Pattern Mask Register. The Pattern Mask register (figure 9) contains a bit pattern used to force a true match between the pattern in the Pattern Match register and the pattern in the Data Buffer register. If a pattern mask bit is set, the corresponding bit in the Pattern Match register always matches with the corresponding bit in the Data Buffer register. All 1s in this register forces a pattern match.

BYTE COUNT COMPARE OPERATION

The Byte Count Compare register holds a byte that is continuously compared with the count in the Byte Count register. If the Byte Count Compare Interrupt Enable bit (D_6) is set in Interrupt Status register 2, an interrupt occurs upon a true match. The comparison operation is not affected by the Freeze Byte Count bit.

If the byte being programmed in the Byte Count Compare register equals the number of bytes that are currently in the FIFO, then the byte count IP bit is set. The range of valid values for the comparison is from 0 to 127 ($7F_H$).

BYTE COUNT REGISTER FREEZE OPERATION

The Byte Count register tracks the number of bytes read from and written to the FIO's FIFO buffer. Since the count changes with each read or write cycle, a provision is made for "freezing" the count so the CPU can read the register's value. When set, bit 6 in Control Register 1 freezes the current value in Byte Count register.

If there are reads or writes after the freeze bit is set, a read of the Byte Count register shows the frozen value and does not reflect the reads or writes that occur between the setting of the freeze bit and the read of the Byte Count register. The read clears the freeze bit. Until another read or write takes place, the frozen value in the Byte Count register is not updated with the current value.

For example, if the Byte Count register shows 10 transactions when the freeze bit is set, and there are five more transactions between the time the freeze bit is set and the time the Byte Count register is read, the read will return the frozen value, 10. If no further transactions (after the five) take place, another read

BYTE COUNT REGISTER FREEZE OPERATION (continued)

of the Byte Count register will still return 10 ; if a read or write takes place, the current value (10 + 5 ± any other transactions) is returned.

$\overline{\text{CLEAR}}$ AND DATA DIRECTION OPERATION

Care should be taken when changing the data direction and clearing the FIFO buffer. Changing the Data Direction pin/bit when the $\overline{\text{CLEAR}}$ pin/bit is High and the FIFO buffer is not empty will cause improper operation. When changing the Data Direction pin/bit the FIFO buffer must be empty. Putting the $\overline{\text{CLEAR}}$ pin/bit Low assures an empty FIFO buffer.

CPU-TO-CPU Operation. Clear Operation. Figure 37, showing $\overline{\text{CLEAR}}$ operation in CPU-to-CPU operation. When bit D_7 of Port 1 is 0, then Port 1 has control over the Clear Bit. When bit D_6 is 1, the FIFO buffer is not clear and is therefore ready to read or write data. If D_7 is then set to 1, and control of the clear function transfers to Port 2. It should be noted that if the Port 2 side is reset when it has control of the $\overline{\text{CLEAR}}$ bit, the $\overline{\text{CLEAR}}$ bit is also reset (0).

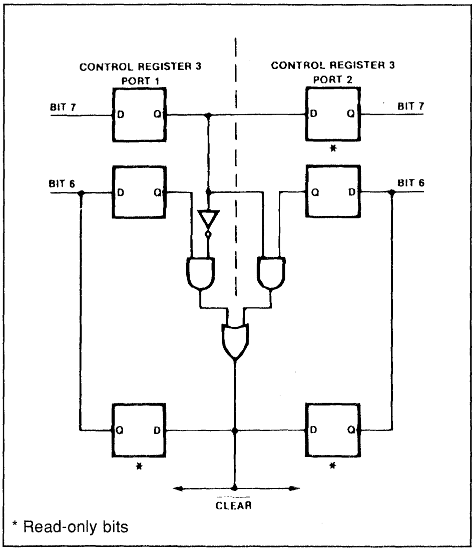
Data Direction Operation. Figure 38 shows the data direction operation for CPU-to-CPU operation. When bit D_5 is 0, Port 1 controls data direction. When D_5 is 1, Port 2 controls data direction. Bit D_4 of the appropriate CPU controls the actual transaction direction, relative to the controlling CPU. When D_4 is 0, data is output from the CPU, and when D_4 is 1, data is input to the CPU. An inverter added to the logic ensures an accurate operation for each side. It should be noted that if the Port 2 side is reset when it has control of the Data Direction bit, the Data Direction bit is also reset. Thus, Port 2's Data Direction bit is 0 and Port 1's Data Direction bit is 1.

CPU-TO-I/O Operation. Clear Operation. Figure 39 shows CLEAR operation in CPU-to-I/O operation. In these transactions, bit D_7 controls whether the FIFO buffer is cleared by resetting D_6 in Control Register 3 or by a system CLEAR signal input on pin 35. If D_7 is 0, Port 1's bit D_6 of Control Register 3 clears the FIO when it is 0. If D_7 is 1, the FIO clears when a system CLEAR signal (active Low) is applied to pin 35.

Data Direction Operation. Figure 40 shows the data direction logic. When bit D_5 is 0, Port 1's bit D_6 in Control Register 3 controls data direction. When D_5 is 1, data direction control is via a system signal received at pin 34. The data direction bit and pin 34 are always the same (no inversion) logic level.

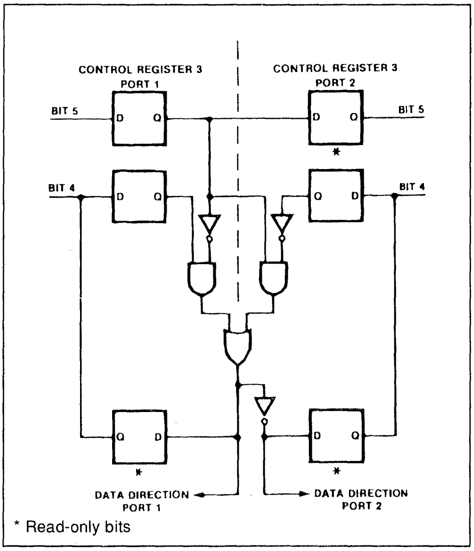
Regardless of whether Port 1 or Port 2 has control of the data direction, Pin 34 determines Port 2 handshake. Pin 34 Low (0) defines the Output Handshake operation ; Pin 34 High (1) defines the Input Handshake operation.

Figure 37 : $\overline{\text{CLEAR}}$: CPU-to-CPU Operation.



* Read-only bits

Figure 38 : Data Direction : CPU-to-CPU Operation.



* Read-only bits

Figure 39 : $\overline{\text{CLEAR}}$: CPU-to-I/O Operation.

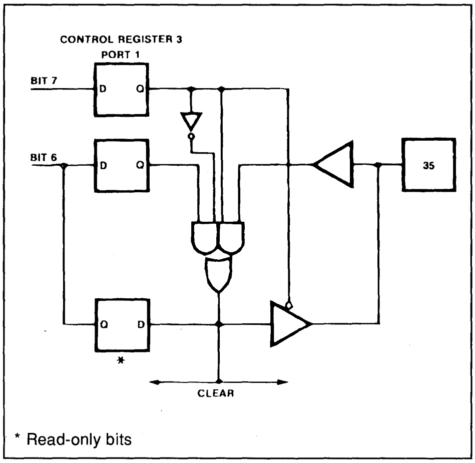
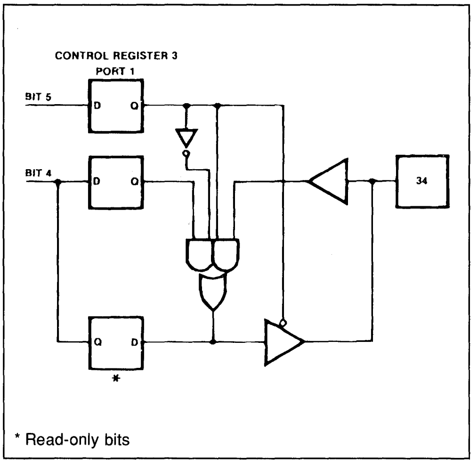


Figure 40 : Data Direction : CPU-to-I/O Operation.



INTERFACING THE FIO

The FIO interfaces to many kinds of devices. The Z-BUS high-byte and low-byte configurations accomodate such devices as the Z8000, Z8, and 8085, which require a multiplexed address/data bus. Devices such as the Z80, 8080, and 6800 use separate address and data buses which are supported by the FIO's non-Z-BUS (nonmultiplexed) microprocessor configuration. The 2-wire handshake I/O configuration is the most commonly used parallel I/O protocol, and the 3-wire handshake I/O configuration supports a multi-acceptor protocol.

This chapter introduces interfacing the FIO to other devices, shows how the FIO interfaces to various bus structures, and shows how buffer interfaces larger than 128 bytes can be constructed with FIOs and FIFOs.

INTERFACING CONSIDERATIONS

Z-BUS Interfacing. All Z-BUS memory or I/O transactions use the $\overline{\text{AS}}$ (Address Strobe) and $\overline{\text{DS}}$ (Data Strobe) signals. These signals define the information currently on the bus as address information or data. An active $\overline{\text{AS}}$ signal latches the $\overline{\text{CS}}$ (Chip Select) signal, which is decoded from the correct I/O address and enables the FIO. During an active $\overline{\text{DS}}$ signal, data is available on the bus. If R/$\overline{\text{W}}$ is High during $\overline{\text{DS}}$, data is read from I/O devices of from memory. If R/$\overline{\text{W}}$ is Low, data is written to I/O devices or to memory.

Z-BUS I/O transactions are asynchronous with the CPU clock. I/O transactions are similar to memory transactions, except that I/O transactions are longer. Memory transactions use three clock cycles for completion, but a single wait state is added automatically to I/O transaction timing, and additional wait-states can be inserted by forcing the $\overline{\text{WAIT}}$ line Low for as long as required.

Z8000 Interfacing. When interfacing to the Z8001 or Z8002 (see figure 4.1) in the Z-BUS low-byte configuration, the RJA bit (D_1 of Control Register 0) can be either 1 or 0. When RJA is 0 the FIO takes the address information on AD_1-AD_4. This configuration is compatible with using Z8000 byte I/O (INB, OUTB) instructions. In byte I/O, the AD_0 bit specifies which byte on the data bus is read or written to ($AD_0 = 0$ uses $AD_8 - AD_{15}$; $AD_0 = 1$ uses $AD_0 - D_7$). Typically, all byte I/O devices have odd addresses.

When RJA is 1 the FIO takes the address information on AD_0-AD_3. This configuration is compatible with using Z8000 word I/O instructions (IN, OUT). The word I/O instructions allow AD_0 to be used as an address bit to the FIO. This is useful when using a High and a Low byte FIO to do word transfers. This configuration is also useful for the user who wants consecutive I/O addresses and is not concerned with Z8000 register space.

The Z-BUS high-byte and low-byte configurations, as well as the non-Z-BUS microprocessor configurations, support the daisy-chain priority interrupt

INTERFACING CONSIDERATIONS (continued)

Figure 41 : Z8000 I/O Timing.

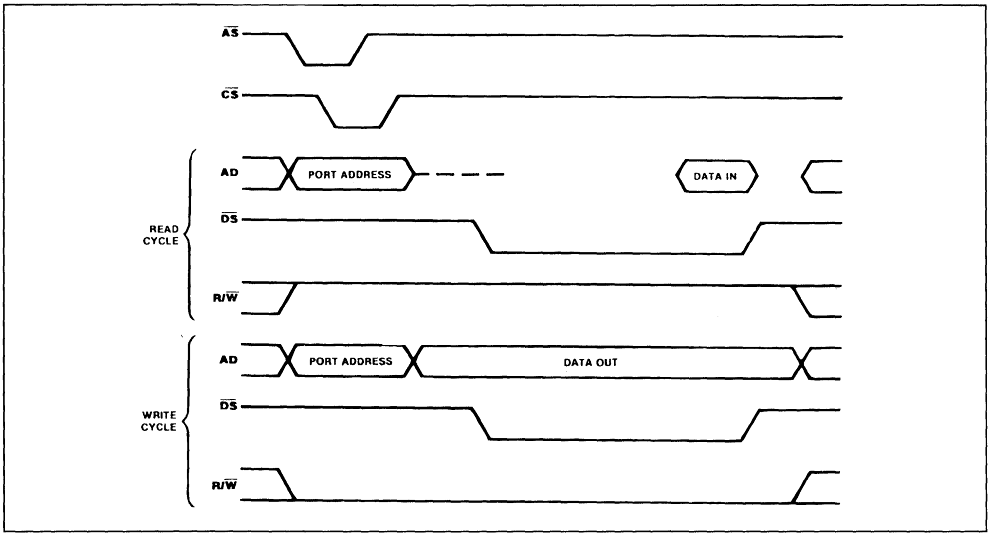
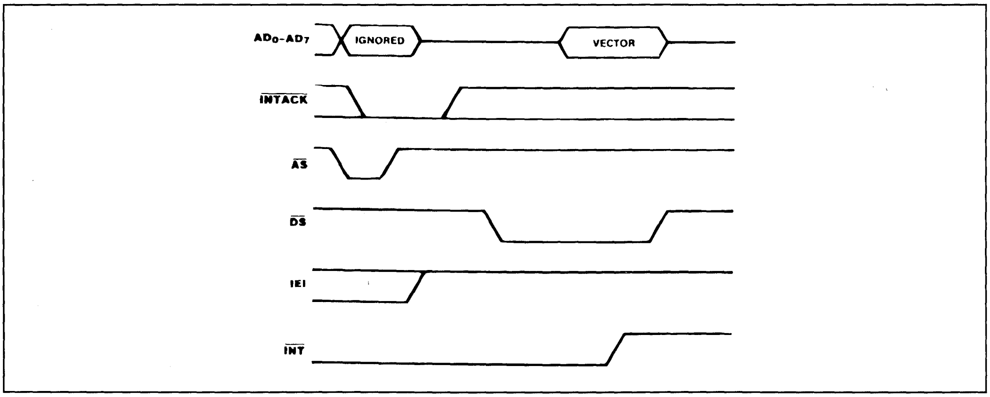


Figure 42 : Z8000 Interrupt Acknowledge Timing.



questing interrupt service to lock other, lower-priority devices out of the interrupt control bus, thus preventing interrupt generation by these devices. Higher-priority devices can still issue interrupt requests that override the original requestor's interrupt service routine.

Additional discrete logic is required to decode the CPU status lines to provide an interrupt acknowledge signal for the FIO. The Z8000 CPU adds five additional wait states (between $\overline{AS}$ rising and $\overline{DS}$ falling) to the interrupt transaction timing (figure 42) as daisy-chain settling time and to allow time for the FIO to place its interrupt vector into the bus.

Non-Z-BUS Interfacing. This interfacing configuration is used with CPUs having separate address and data lines. The FIO is connected only to the data bus of such CPUs, so a signal, C/$\overline{D}$, must distinguish between control bytes and data bytes. In the non-Z-BUS configuration, the addresses of the FIO internal registers must appear on data lines D_0 – D_3. Ac-

INTERFACING CONSIDERATIONS (continued)

cordingly, the RJA bit (D_1 of Control Register 0) is always set by the FIO.

In this configuration, the $\overline{CE}$ (Chip Enable) signal must be decoded from the I/O address of the FIO by external logic. Since $\overline{CE}$ is not latched, it must remain true during the entire I/O transaction.

Z80 Interface. I/O Request ($\overline{IORQ}$) and Memory Request ($\overline{MREQ}$) specify I/O or memory transactions ; $\overline{WR}$ and $\overline{RD}$ specify write or read operations. During I/O operation, the Z80 inserts a wait state into the timing to give the FIO time to decode its address and to activate its own $\overline{WAIT}$ line if additional time is needed. Figure 43 shows this timing.

The Z80 samples the $\overline{INT}$ line at the rising edge of the last clock cycle at the conclusion of every instruction. If an interrupt request is detected, the CPU issues an M1 signal without a corresponding $\overline{MREQ}$ signal. The Z80 adds two Wait states to allow the IEI/IEO daisy chain interrupt lines to settle. External logic must generate an $\overline{INTACK}$ signal from these signal conditions. Figure 44 shows various signals from the Z80 and the signals generated from these and sent to the FIO. Figure 45 shows the external logic used to interface the FIO and other 8500 series peripherals to the Z80 bus. When the generated $\overline{INTACK}$ and $\overline{RD}$ are both Low (active), the FIO places its interrupt vector on the bus, if enabled to do so.

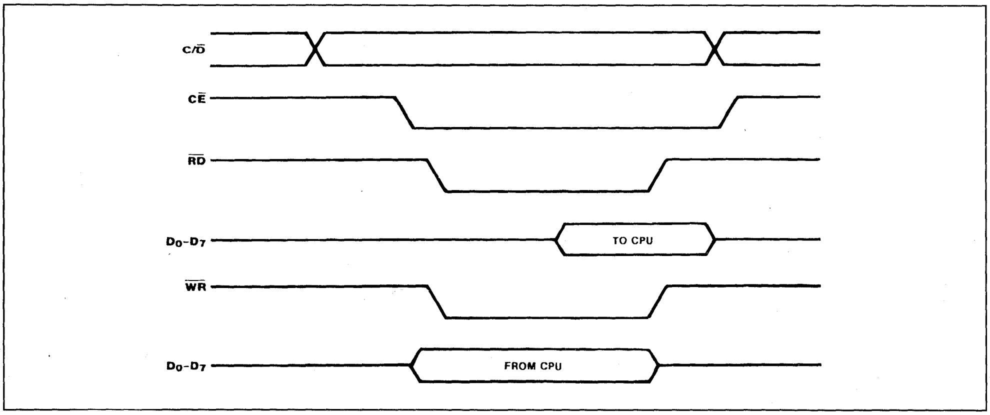
2-Wire (Interlocked) Handshake I/O Interfacing. The 2-wire handshake configuration is usually used when a single I/O device communicates with a CPU ; the 3-wire-handshake configuration (similar to the IEEE-488 standard interface) is used when a CPU is interfacing to a group of devices.

2-Wire Handshake Configuration. During a data transfer into the FIO, RFD/DAV is forced High, signaling to the peripheral device that the FIO is ready for data. The peripheral places the data on the bus and acknowledges the transaction with $\overline{ACKIN}$ Low, latching the data into the FIO Data Buffer register on the falling edge. RFD then goes Low until the Data Buffer register is emptied into the buffer memory, at which time RFD goes High (active) if $\overline{ACKIN}$ has gone High (inactive). The cycle repeats until the peripheral is through writing or until the buffer memory is full. When the buffer memory is full, RFD goes Low and remains so.

During a data transfer from the FIO, $\overline{DAV}$ goes active (Low) to signify that data is available from the FIO's Data Buffer register. If $\overline{ACKIN}$ is High, signifying that the previous read from the FIO is completed, the device latches the data from the FIO and forces $\overline{ACKIN}$ Low. The FIO forces $\overline{DAV}$ High when the Data Buffer register is read. $\overline{ACKIN}$ strobes the next byte from the buffer to the Data Buffer register, $\overline{DAV}$ goes Low, and the cycle repeats until the FIO buffer is empty when $\overline{DAV}$ goes High and remains so.

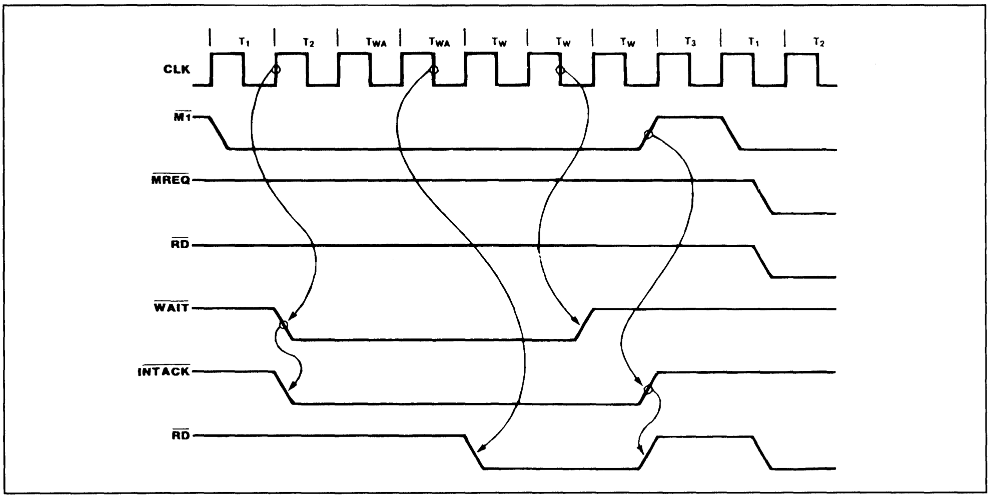
3-Wire Handshake Configuration. During a data transfer into the FIO, the FIO forces RFD High (active) to signal that it is ready for data. The $\overline{DAV}$ input on pin 38 goes Low (active) to strobe the incoming data from the bus lines into the Data Buffer register. The DAC input goes High on pin 37 to show the input device that the data has been accepted.

Figure 43 : Non-Z-BUS I/O Timing.



INTERFACING CONSIDERATIONS (continued)

Figure 44 : Z80 Interrupt Acknowledge Timing.

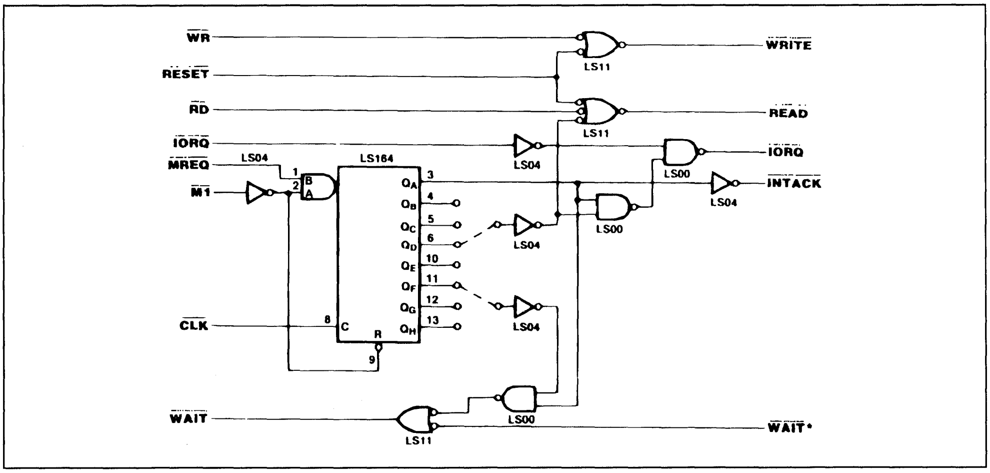


RFD on pin 39 remains Low until the data has been moved from the Data Buffer register to the buffer memory. When RFD goes High again, the cycle repeats until the FIO buffer is full. During a data transfer from the FIO, the RFD signal from the peripheral devices goes High to show they are ready for data. $\overline{DAV}$ on pin 39 goes Low (active) to show that there is valid data on the bus. The peripheral devices return DAC High (active) on pin 37, signifying that the data has been accepted. When the peripherals are ready for more data, RFD reappears High on pin 37 and the cycle begins again. Output continues until the FIO buffer is empty, which forces $\overline{DAV}$ High (inactive). $\overline{DAV}$ stays High until the buffer is reloaded from the CPU port. The peripherals can stop the transaction by holding RFD Low on pin 37.

FIO EXPANSION INTERFACING

FIOs can be combined to create 16-bit or wider interfaces and to create buffers larger than 128 bytes. The following text and illustrations show how FIOs

Figure 45 : External Logic Interfacing the Z80 to 8500 Series Peripherals.



SGS-THOMSON MICROELECTRONICS

INTERFACING CONSIDERATIONS (continued)

can be combined or used with the Z8060 FIFO buffer to create a buffer size of 512 bytes.

Buffer size can be a critical factor in an I/O-bound system. A larger buffer allows longer intervals between situations that require CPU intervention and can cut I/O overhead significantly. Using a device as powerful as the FIO, which conducts pattern matches and generates interrupts upon a variety of conditions, the CPU spends less time polling I/O devices. The FIO also conducts duplex DMA operation at system speeds, transferring a byte each cycle, since the buffer can be read from and written to simultaneously.

CPU-TO-CPU 256-Byte FIO Buffer Expansion. Figure 46 shows a configuration of FIOs that creates a 256-byte buffer. Port 2 of one FIO is interfaced to Port 2 of the other FIO, and both are programmmed in the 2-wire handshake configuration.

The Empty and Full pins are wire-ORed to communicate these conditions only when both FIOs reflect the same condition.

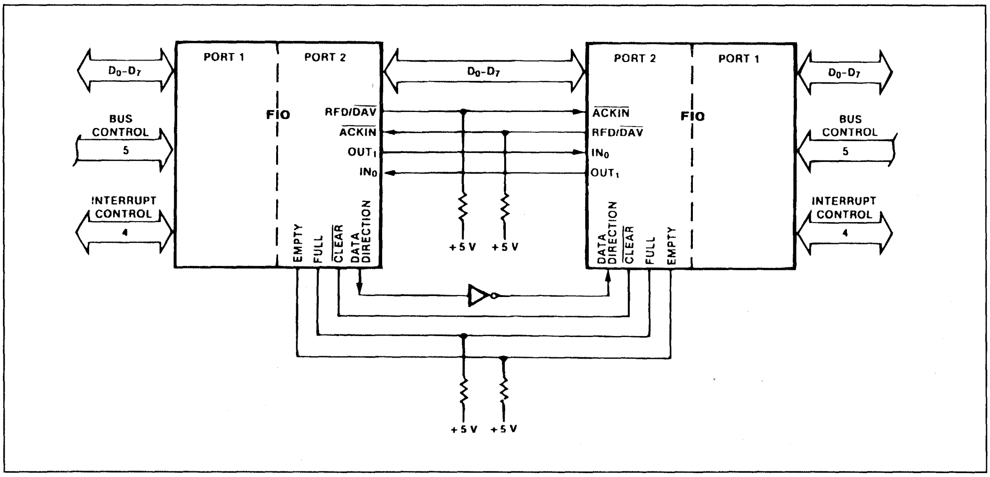
The message register feature does not operate in any of the expanded configurations, so some other provision must be made for communication between two CPUs controlling a multiple-FIO configuration. This can be done by using the IN_0 and OUT_1 signals (mapped to pins 33 and 32, respectively, in the 2-wire handshake configuration). In the 2-wire handshake configuration, the Port 1 CPU can set or reset bit D_1 of Port 1 Control Register 3. The state of this bit is reflected in bit D_0 and can be read by the OUT_1 line from the opposite FIO. Conversely, the opposite FIO can set its own bit D_1, simultaneously setting bit D_0, which can then be read by the first FIO.

The expansion process modifies the operation of the Byte Count register. When data is written to a port, it moves (bubbles) to the first bit position of the buffer opposite the one that is writing. For example, if the left CPU is writing 256 bytes to the FIOs shown in the figure 46, the right FIO buffer will be loaded with 128 bytes before any bytes are stored into the left FIO buffer.

If 100 bytes are then written into this configuration from the left CPU, the right CPU can read its Byte Count register and obtain an accurate read of 100 bytes. If the left FIO reads its Byte Count register, however, it will be empty. If another 100 bytes are then transferred from left to right, the right CPU will read the buffer of its FIO as full (128 bytes loaded) and the left FIO will show 200 minus 128 (or 72) bytes in its buffer. When 256 bytes are written to this configuration, the wire-ORed full line will reflect a buffer-full state by going High.

During a read from these filled buffers, each time a byte is read from the right FIO by its CPU, a byte from the left FIO trickles across to the right FIO. Therefore, the left FIO buffer is emptied before the right FIO begins its countdown to empty. When both buffers are empty, the wire-ORed empty line reflects a buffer-empty state by going High. The empty and full interrupts are modified as well. Empty interrupts

Figure 46 : CPU-to-CPU 256-Byte FIO Buffer Expansion.



FIO EXPANSION INTERFACING (continued)

signify that both buffers are empty and full interrupts signify that both buffers are full.

In this example, the left FIO has control of Data Direction. Its Data Direction pin is an output and the right FIO's Data Direction pin is an input. To ensure proper operation an inverter is placed between the two FIO's Data Direction pins. This provides, for example, that when the left FIO is in Output Handshake (Data DIrection = 0) mode the right FIO will be in Input Handshake mode (Data Direction = 1).

Figure 47 : CPU-to-CPU 512-Byte FIO Buffer Expansion.

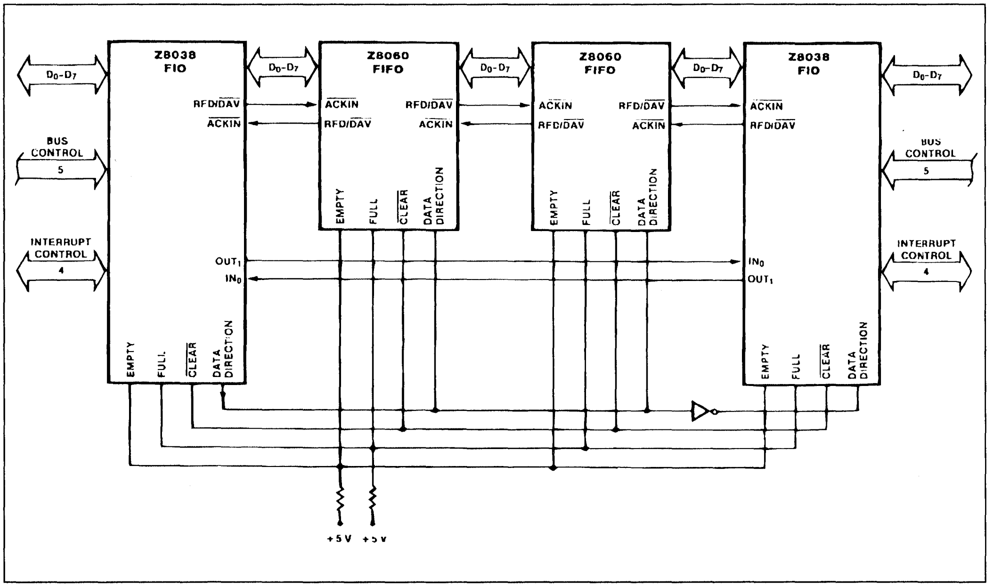
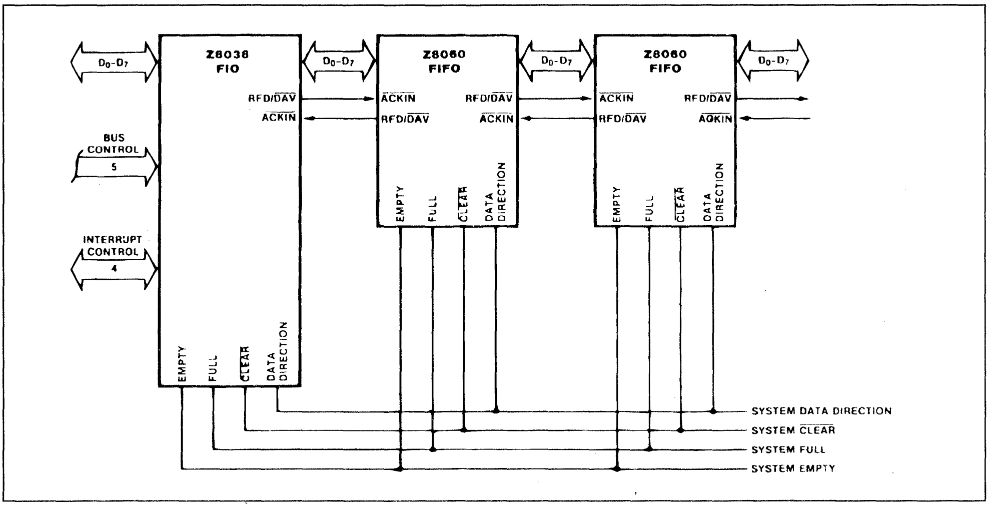


Figure 48 : CPU-to-I/O 384-Byte FIO Buffer Expansion.



FIO EXPANSION INTERFACING (continued)

CPU-TO-CPU 512-Byte FIO-TO-FIFO Buffer Expansion. Figure 47 illustrates a 512-byte buffer interface constructed from two Z8038 FIOs and two Z8060 FIFOs. The Z8060 only operates in the 2-wire hand-shake configuration. The number of FIFOs used in such a configuration is limited only by economics and space.

The FIOs and FIFOs share the same signal configurations used in the previous example, i.e., wire-ORed Full and Empty signals. In this configuration, however, all buffers must be filled (512 bytes) before the Full signal is active, and all buffers must be empty before the Empty signal is active.

CPU-TO-Peripheral 384-Byte FIO Buffer Expansion. Figure 48 illustrates a 384-byte buffer interface constructed from one FIO and two FIFOs. Additional FIFOs can be added as needed, each increasing the buffer capacity. As in the other configurations, the 2-wire handshake configuration must be used.

INTERFACING THE FIO

The FIO manages data transactions between CPUs and between CPUs and peripherals. In some applications, external logic is required to interface the FIO properly, and the software must support the target configuration. This chapter gives two examples of the FIO interfacing to devices : one example shows how the FIO interfaces between the Z8002 CPU and a controller, and the other example shows a Z8000-to-FIO-to-Z80 interface.

FIO INTERFACE BETWEEN Z8002 AND A DISPLAY CONTROLLER

Figure 49 shows a typcial application for the FIO : interfacing a display controller to the Z-BUS. The Z-BUS master, a Z8002 CPU, controls the transactions through Port 1 of the FIO. Port 1 of the FIO is configured in the Z-BUS low byte mode by tying pins 19 (M_1) and 21 (M_0) to ground. In this configuration, the Port 1 Data lines D_0 - D_7 connect direclty to Z-BUS Address/Data lines AD_0 - D_7, and the FIO control registers are directly addressable via these signal lines. The FIO also accepts the Z-BUS interrupt signals directly. The FIO is mapped into I/O locations FFEO - FFFF. Figure 51 shows the software module to interface to the Z8002.

The Z8002 CPU loads the FIO buffer with data for the display controller. When the FIO buffer is full, the WAIT line prevents buffer overflow ; when the buffer is empty, the FIO generates an interrupt request. Clearing the FIO buffer also resets the display controller.

The signal connections between the FIO'd Port 2 and the display controller are straight-forward. The DATA AVAILABLE signal selects the controller when active ; the controller's READY line notifies the FIO's Acknowledge In line when it will accept more data. System RESET and the FIO CLEAR signals are ANDed to create the controller RESET signal.

In this example, the Z8002 CPU adresses the FIO using byte I/O instructions. Therefore, the FIO's RJA bit (D1 of Control Register 0) is 0 and the I/O addresses are all odd.

Figure 49 : FIO to Display Controller.

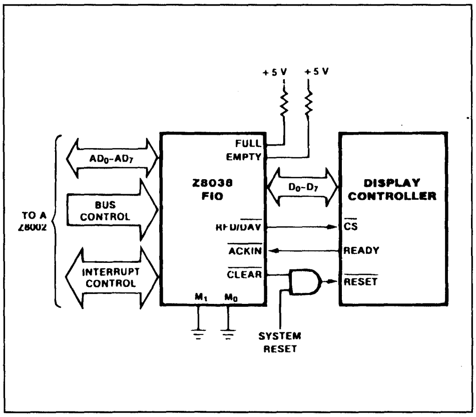
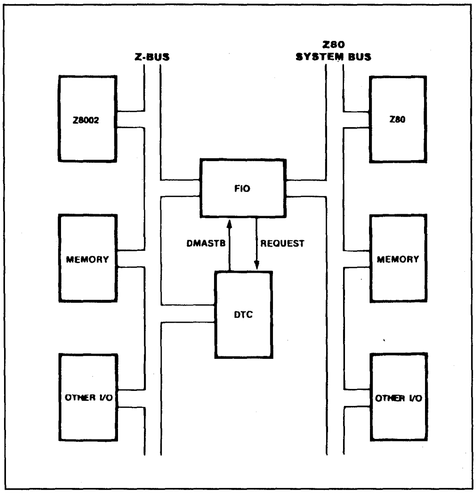


Figure 50 : FIO Interface between the Z8002 and a Display Controller.



SGS-THOMSON MICROELECTRONICS

FIO INTERFACE BETWEEN Z8002 AND A DISPLAY CONTROLLER (continued)

Figure 51 : Z8002 Initialization Module for the FIO.

```
FIO MODULE                    ! THIS MODULE CONTAINS A PROCEDURE TO INITIALIZE
                        AN FIO FOR USE WITH A DISPLAY CONTROLLER. THE PORT 1 SIDE OF THE
                        FIO IS CONNECTED TO THE LOW-ORDER HALF OF THE Z-BUS, AND THE
                        PORT 2 SIDE IS AN INTERLOCKED HANDSHAKE TO OUTPUT DATA TO THE
                        DISPLAY CONTROLLER. THE FIO IS CHIP SELECTED BY I/O ADDRESSED
                        FFEO  FFFF. !

CONSTANT                          ! FIO REGISTER ADDRESSES !
    CNTL_0 : = % FFE1             CNTL_1 : = % FFE3
    CNTL_2 : = % FFF3             CNTL_3 : = % FFF5
    INT_STATO : = % FFE5          INT_STAT1 : = % FFE7
    INT = STAT2 : = % FFE9        INT_STAT3 : = % FFEB
    INT_VECTOR : = % FFED         FIO = DATA : = % FFFF

    VECTOR : = % F0               ! BASE VECTOR RETURNED DURING INTERRUPT ACK.
                                    SEQUENCE !

GLOBAL
    FIO_INIT PROCEDURE            ! FIO INITIALIZATION PROCEDURE !
      ENTRY
         PUSH @ R15, R0           ! SAVE R0 !

         LDB RL0, # % 01
         OUTB # % FFEO RL0        ! RESET FIO !
         CLRB RL0
         OUTB CNTL_0, RL0         ! TURN OFF RESET !

         LDB RL0, # % (2) 10011100
         OUTB CNTL_0, RL0         ! MIE ON, VECTOR INCLUDES STATUS, INTERLOCKED
                                    H.S. ON PORT 2 SIDE, RJA = 0 (AD1  AD4) !

         LDB RL0, # % (2) 00000001
         OUTB CNTL_1, RL0         ! WAIT ENABLED !

         LDB RL0, # % (2) 01000000
         OUTB CNTL_3, RL0         ! PORT 1 SIDE CONTROLS CLEAR AND DATA DIRECTION,
                                    PORT 1 ACCEPTS CPU OUTPUT, BUFFER CAN HOLD
                                    DATA (CLEAR REMOVED) !

         LDB RL0, # VECTOR
         OUTB INT_VECTOR, RL0     ! LOAD BASE INTERRUPT VECTOR, VECTOR RETURNED
                                    DURING INTERRUPT ACK. WILL CONTAIN STATUS IN
                                    BITS 1, 2, AND 3 !

         LDB RL0, # % (2) 00000011
         OUTB CNTL_2, RL0         ! ENABLE PORT 2 AND PORT 2 HANDSHAKE !

         POP R0, @ R15            ! RESTORE R0 !
         RET
    END FIO_INIT
END FIO
```

FIO INTERFACE BETWEEN Z-BUS AND Z80

Figure 50 shows the FIO interfacing the Z-BUS to the Z80 bus. In this example, the Z-BUS connects to Port 1 of the FIO via Address/Data lines $AD_0 - AD_7$, as in the previous example. The Z8002 CPU controls the $\overline{\text{CLEAR}}$ and DATA DIRECTION functions. The DTC device fills and empties the buffer by DMA transactions to and from Z-BUS memory. Port 1 of the FIO is mapped into I/O locations FFEO – FFFF. Figure 52 shows the Z80 interrupt acknowledge timing.

The logic necessary to generate this timing is shown in figure 53.

In the software given in figure 54, a Z8000 routine initializes Port 1 to receive data from the Z8002 system. The FIO requests a DMA transaction when the buffer contains 10 or less bytes. The following Z80 program initializes Port 2. The FIO's data port is at address FE, and the control port is at address FF.

Figure 52 : Z80 Interrupt Acknowledge Timing.

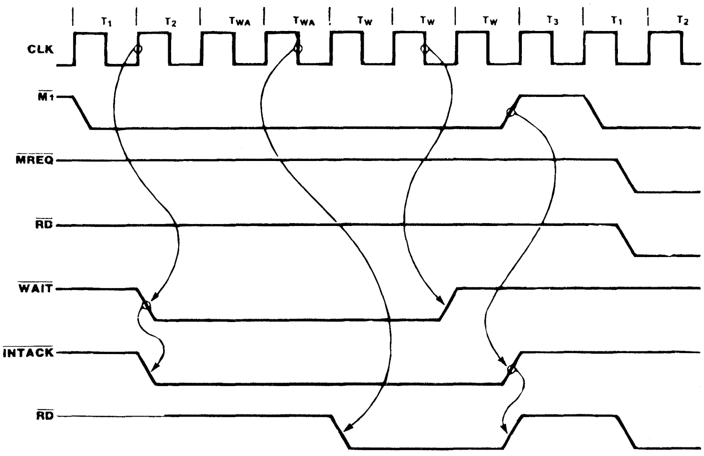
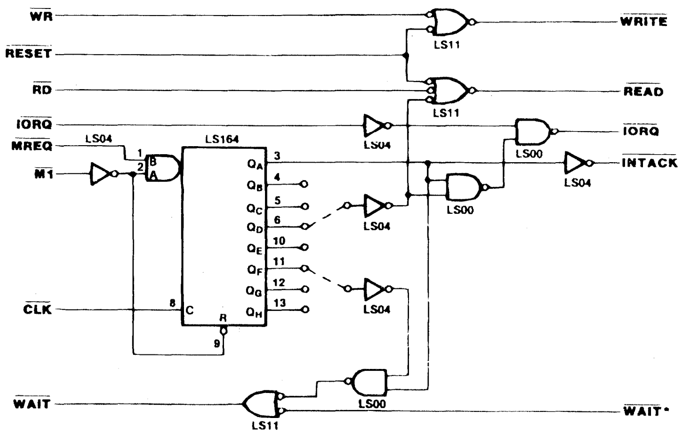


Figure 53 : Z80 $\overline{\text{INTACK}}$ Generation for the FIO.



SGS-THOMSON MICROELECTRONICS

FIO INTERFACE BETWEEN Z-BUS AND Z80 (continued)

Figure 54 : FIO Port 1 Initialization..

```
FIO_Z8000_TO_Z80 MODULE             ! Z8000 PROGRAM TO INITIALIZE FIO WHICH INTER-
                          FACES Z8000 SYSTEM TO Z80 SYSTEM. THE SYSTEM IS INITIALIZED SO THAT
                          THE Z8000 IS SENDING DATA TO THE Z80 VIA THE FIFO BUFFER. THE Z8000
                          IS IN CONTROL OF DATA DIRECTION AND BUFFER CLEARING. TRANSFERS
                          BETWEEN MEMORY AND THE FIO ON THE Z8000 SIDE ARE HANDLED BY A
                          DMA CONTROLLER. THE Z8000 IS CONNECTED TO THE PORT 1 SIDE OF THE
                          FIO ; THE FIO IS CHIP SELECTED BY I/O PORT ADDRESSES FFEO THROUGH
                          FFFF, AND IS CONNECTED TO THE LOWER HALF OF THE Z-BUS !

CONSTANT                            ! FIO REGISTER ADDRESSES !
    CNTL_0 : = % FFE1               CNTL_1 : = % FFE3
    CNTL_2 : = % FFF3               CNTL_3 : = % FFF5
    INT_STAT0 : = % FFE5            INT_STAT1 : = % FFE7
    INT_STAT2 : = % FFE9            INT_STAT3 : = % FFEB
    BYTE_CNT : = % FFEF             INT_VECTOR : = % FFED
    FIO_DATA : = % FFFF             DATA_CNT_CMP : = % FFF1
    MSG_IN : = % FFF9               MSG_OUT : = % FFF7
    VECTOR : = % 80                 ! BASE VECTOR !
    STARTING_CNT : = 10             ! INITIAL BYTE COMPARISON REGISTER COUNT !

GLOBAL
    FIO_INIT PROCEDURE
       ENTRY
           PUSH @ R15, R0           ! SAVE R0 !

           LDB RL0, # % 01
           OUTB # % FFE0, RL0       ! RESET FIO !
           CLRB RL0
           OUTB CNTL_0, RL0         ! TURN OFF RESET !
           LDB RL0, # % (2) 10010100
           OUTB CNTL_0, RL0         ! MIE ON, VECTOR INCLUDES STATUS, PORT 2 IS NON-
                                      Z-BUS CPU, REG. ADDRESSES SHIFTED LEFT 1 BIT !
           LDB RL0, # % (2) 01000000
           OUTB CNTL_3, RL0         ! PORT 1 SIDE CONTROLS CLEAR AND DATA DIRECTION,
                                      PORT 1 GETS CPU OUTPUT !
           LDB RL0, # VECTOR
           OUTB INT_VECTOR, RL0     ! LOAD BASE VECTOR, VECTOR RETURNED DURING INT.
                                      ACK. INCLUDES STATUS !
           LDB RL0, # % (2) 11000000
           OUTB INT_STAT0, RL0      ! ENABLE MAILBOX REGISTER INTERRUPT !
           LDB RL0, # % (2) 00000001
           OUTB CNTL_2, RL0         ! ENABLE PORT 2 SIDE !
           LDB RL0, # STARTING_CNT
           OUTB DATA_CNT_CMP, RL0   ! LOAD COUNT OF 10 INTO BYTE COUNT COMPARE RE-
                                      GISTER, REQUEST WILL BE MADE TO DMA WHEN BUFF-
                                      ER HAS 10 OR LESS BYTES IN IT !
           LDB RL0, # % (2) 0000011
           OUTB CNTL_1, RL0         ! REQUEST TO DMA ENABLED !
           POP R0, @ R15            ! RESTORE R0 !
           RET
    END FIO_INIT
END FIO_Z8000_TO_Z80
```

FIO INTERFACE BETWEEN Z-BUS AND Z80 (continued)

Figure 55 : FIO Port 2 Initialization.

```
;          THIS Z80 PROGRAM INITIALIZES PORT 2 OF AN FIO
;          USED TO CONNECT A Z8000 AND Z80 SYSTEM TOGETHER
;          WITH THE Z80 RECEIVING DATA FROM THE FIO, VIA
;          AN INTERRUPT WHEN THE FIFO BUFFER IS FULL
;          THE FIO'S DATA PORT IS AT PORT FE HEX, AND THE
;          CONTROL PORT IS AT PORT FF HEX.

FIODAT     EQU    0FEH               ; FIO DATA PORT ADDRESS
FIOCTL     EQU    FIODAT + 1         ; FIO CONTROL PORT ADDR
FIOVEC     EQU    0010H              ; FIO INT. VECTOR ADDR
START :
           IN     A, (FIOCTL)        ; INSURE STATE 0
           XOR    A                  ; CLEAR REG A
           OUT    (FIOCTL), A        ; REMOVE RESET
           LD     A, 1               ; POINT TO REG 1
           OUT    (FIOCTL), A
           XOR    A                  ; CLEAR REG A
           OUT    (FIOCTL), A        ; CLEAR RESET
           OUT    (FIOCTL), A        ; POINT TO REG 0
           LD     A, 1               ; SET RESET BIT
           OUT    (FIOCTL), A        ; FIO IS RESET NOW
WAITLP :
           IN     A, (FIOCTL)        ; READ REG 0
           CP     000000001          ; CHECK PORT 2 MODE
           JR     Z, WAITLP          ; LOOP UNTIL SET
           XOR    A                  ; CLEAR RESET
           OUT    (FIOCTL), A
           LD     HL, FIOLST         ; LOAD INIT LIST PTR
           LD     C, FIOCTL          ; LOAD FIO PORT ADDR
           LD     B, FIOEND-FIOST    ; LOAD LIST LENGTH
           OTIR
           RET
FIOLST :
           DEFB   0                  ; CTRL REGISTER
           DEFB   10010010B          ; MIE, VIS, RJA
           DEFB   1                  ; CTRL REG 1
           DEFB   00000001B          ; NO DMA, WAIT ENABLED
           DEFB   6                  ; INT VECTOR REG
           DEFB   FIOVEC. AND. 255   ; LOWER INTERRUPT VECTOR
           DEFB   2                  ; INT STATUS REG 0
           DEFB   11000000B          ; SET MESSAGE IE
           DEFB   3                  ; INT STATUS REG 1
           DEFB   00001010B          ; CLEAR PATTERN MATCH IP
           DEFB   3
           DEFB   11001100B          ; SET PM IE, SET DDC IE
           DEFB   4                  ; INT STATUS REG 2
           DEFB   00001100B          ; SET ERROR IE
           DEFB   5                  ; INT STATUS REG 3
           DEFB   11001100B          ; SET FULL IE, EMPTY IE
FIOEND :   EQU    $

           END
```

PIN ASSIGNMENTS

Z-BUS Low Byte Mode : Port 1 Side.

Pin Signals	Pin Numbers	Signal Description
AD0-AD7 (address/data)	11-18	Multiplexed Bidirectional Address/data Lines, Z-BUS Compatible
$\overline{\text{REQ}}/\overline{\text{WAIT}}$ (request/wait)	1	Output, Active Low. $\overline{\text{REQUEST}}$ (ready) line for DMA Transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data Transfers.
$\overline{\text{DMASTB}}$ (direct Memory Access Strobe)	2	Input, Active Low. Strobes DMA Data to and from the FIFO Buffer.
$\overline{\text{DS}}$ (data strobe)	3	Input, Active Low. Provides Timing for Data Transfer to or from FIO.
R/$\overline{\text{W}}$ (read/write)	4	Input ; Active high signals CPU read from FIO ; Active low signals CPU write to FIO.
$\overline{\text{CS}}$ (chip select)	5	Input, Active Low. Enables FIO. Latched on the Rising Edge of AS.
$\overline{\text{AS}}$ (address strobe)	6	Input, Active Low. Addresses, CS and $\overline{\text{INTACK}}$ sampled while AS Low.
$\overline{\text{INTACK}}$ (interrupt acknowledge)	7	Input, Active Low. Acknowledges an Interrupt. Latched on the Rising Edge of AS.
IEO (interrupt enable out)	8	Output, Active High. Sends interrupt enable to lower priority device IEI pin.
IEI (interrupt enable in)	9	Input, Active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{\text{INT}}$ (interrupt)	10	Output, Open Drain, Active Low. Signals FIO interrupt request to CPU.

Z-BUS Low Byte Mode : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
AD0-AD7 (address/data)	29-22	Multiplexed Bidirectional Address/data Lines, Z-BUS Compatible
$\overline{\text{REQ}}/\overline{\text{WAIT}}$ (request/wait)	39	Output, Active Low. $\overline{\text{REQUEST}}$ (ready) line for DMA Transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data Transfers.
$\overline{\text{DMASTB}}$ (direct Memory Access Strobe)	38	Input, Active Low. Strobes DMA Data to and from the FIFO Buffer.
$\overline{\text{DS}}$ (data strobe)	37	Input, Active Low. Provides Timing for Data Transfer to or from FIO.
R/$\overline{\text{W}}$ (read/write)	36	Input ; Active high signals CPU read from FIO ; Active low signals CPU write to FIO.
CS (chip select)	35	Input, Active Low. Enables FIO. Latched on the Rising Edge of AS.
AS (address strobe)	34	Input, Active Low. Addresses, $\overline{\text{CS}}$ and $\overline{\text{INTACK}}$ sampled while AS Low.
$\overline{\text{INTACK}}$ (interrupt acknowledge)	33	Input, Active Low. Acknowledges an Interrupt. Latched on the Rising Edge of AS.
IEO (interrupt enable out)	32	Output, Active High. Sends interrupt enable to lower priority device IEI pin.
IEI (interrupt enable in)	31	Input, Active High. Receives interrupt enable from higher priority device IEO signal.
INT (interrupt)	30	Output, Open Drain, Active Low. Signals FIO interrupt request to CPU.

PIN ASSIGNMENTS (continued)

Non-Z-BUS Mode : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D_0-D_7 (data)	29-22	Bidirectional Data Bus
$\overline{\text{REQ}}$/WAIT (request/wait)	39	Output, Active Low. $\overline{\text{REQUEST}}$ (ready) Line for DMA Transfer ; WAIT Line (open-drain) output for Synchronized CPU and FIO Data Transfers.
$\overline{\text{DACK}}$ (DMA acknowledge)	38	Input, Active Low. DMA Acknowledge
$\overline{\text{RD}}$ (read)	37	Input Active low. Signals CPU read from FIO.
$\overline{\text{WR}}$ (write)	36	Input Active Low. Signals CPU write to FIO.
$\overline{\text{CE}}$ (chip select)	35	Input, Active Low. Used to Select FIO.
C/$\overline{\text{D}}$ (control/data)	34	Input, Active High. Indentifies Data Byte on D0-D7.
$\overline{\text{INTACK}}$	33	Input, Active Low. Acknowledges an interrupt
IEO (interrupt enable out)	32	Output, Active high. Sends interrupt enable to lower priority device IEI pin.
IEI (interrupt enable in)	31	Input, Active high. Recives interrupt enable from higher priority device IEO signal.
$\overline{\text{INT}}$ (interrupt)	30	Output, Open Drain, Active Low. Signals FIO interrupt to CPU.

Figure 56 : Z-BUS-Low-Byte to Z-BUS-Low-Byte.

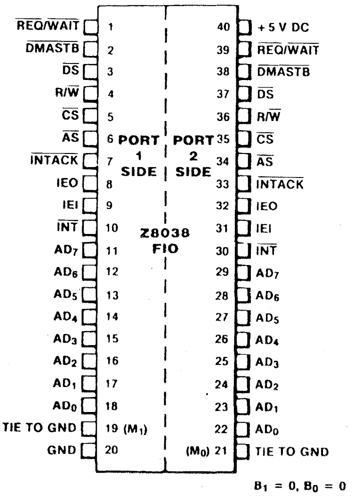
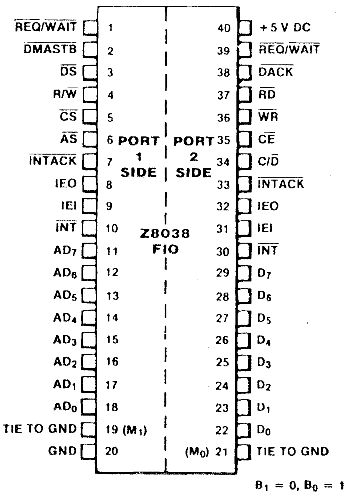


Figure 57 : Z-BUS-Low-Byte to Non-Z-BUS.



PIN ASSIGNMENTS (continued)

Z-BUS Low Byte Mode : Port 1 Side.

Pin Signals	Pin Numbers	Signal Description
AD0-AD7 (address/data)	11-18	Multiplexed Bidirectional Address/data Lines, Z-BUS Compatible
$\overline{\text{REQ}}/\overline{\text{WAIT}}$ (request/wait)	1	Output, Active Low. $\overline{\text{REQUEST}}$ (ready) line for DMA Transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data Transfers.
$\overline{\text{DMASTB}}$ (direct Memory Access Strobe)	2	Input, Active Low. Strobes DMA Data to and from the FIFO Buffer.
$\overline{\text{DS}}$ (data strobe)	3	Input, Active Low. Provides Timing for Data Transfer to or from FIO.
R/$\overline{\text{W}}$ (read/write)	4	Input ; Active high signals CPU read from FIO ; Active low signals CPU write to FIO.
$\overline{\text{CS}}$ (chip select)	5	Input, Active Low. Enables FIO. Latched on the Rising Edge of AS.
$\overline{\text{AS}}$ (address strobe)	6	Input, Active Low. Addresses, CS and $\overline{\text{INTACK}}$ sampled while AS Low.
$\overline{\text{INTACK}}$ (interrupt acknowledge)	7	Input, Active Low. Acknowledges an Interrupt. Latched on the Rising Edge of AS.
IEO (interrupt enable out)	8	Output, Active High. Sends interrupt enable to lower priority device IEI pin.
IEI (interrupt enable in)	9	Input, Active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{\text{INT}}$ (interrupt)	10	Output, Open Drain, Active Low. Signals FIO interrupt request to CPU.

PIN ASSIGNMENTS (continued)

3-Wire Handshake : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D_0-D_7 (data)	29-22	Bidirectional Data Bus
RFD/$\overline{\text{DAV}}$ (ready for data/data available)	39	Output, RFD Active High. Signals peripherals that FIO is ready to receive data. DAV active low signals that FIO is ready to send data to peripherals.
$\overline{\text{DAV}}$/DAC (data available/data accepted)	38	Input ; $\overline{\text{DAV}}$ (active low) signals that data is valid on bus. DAC (active high) signals that ouput data is accepted by peripherals.
DAC/RFD (data accepted/ready for data)	37	Direction Controlled by Internal Programming. Both Active high. DAC (an output) signals that FIO has received data from peripheral ; RFD (an input) signals that the listeners are ready for data.
EMPTY	36	output, Input, Open Drain, Active High. Signals that FIFO buffer is empty.
$\overline{\text{CLEAR}}$	35	Programmable Input or Output, Active Low. Clears All Data from FIFO Buffer.
DATA DIR (data direction)	34	Programmable Input or Output. Active High Signals Data Input to Port 2 ; Low Signals Data Output from port 2.
IN_0	33	Input Line to D_0 of Control Register 3
OUT_0	32	Output Line From D_1 of Control Register 3
$\overline{\text{OE}}$ (output enable)	31	Input, Active low. When low, enables bus drivers. When high, floats bus drivers at high impedance.
OUT_3	30	Output line from D_3 of Control Register 3

PIN ASSIGNMENTS (continued)

2-Wire Handshake : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D_0-D_7 (data)	29-22	Bidirectional Data Bus
RFD/$\overline{\text{DAV}}$ (ready for data/data available)	39	Output, RFD Active High. Signals peripherals that FIO is ready to receive data. DAV active low signals that FIO is ready to send data to peripherals.
$\overline{\text{ACKIN}}$ (acknowledge input)	38	Input, Active Low. Signals FIO that output data is received by peripherals or that input data is valid.
FULL	37	Output, Input, Open Drain, Active High. Signals that FIO buffer is full.
EMPTY	36	Output, Input, Open Drain, Active High. Signals that FIFO buffer is empty.
$\overline{\text{CLEAR}}$	35	Programmable Input or Output, Active Low. Clears All Data from FIFO Buffer.
DATA DIR (data direction)	34	Programmable Input or Output. Active High Signals Data Input to Port 2 ; Low Signals Data Output from port 2.
IN_0	33	Input Line to D_0 of Control Register 3
OUT_0	32	Output Line From D_1 of Control Register 3
$\overline{\text{OE}}$ (output enable)	31	Input, Active low. When low, enables bus drivers. When high, floats bus drivers at high impedance.
OUT_3	30	Output line from D_3 of Control Register 3

Figure 58 : Z-BUS-Low-Byte to 2-Wire Handshake.

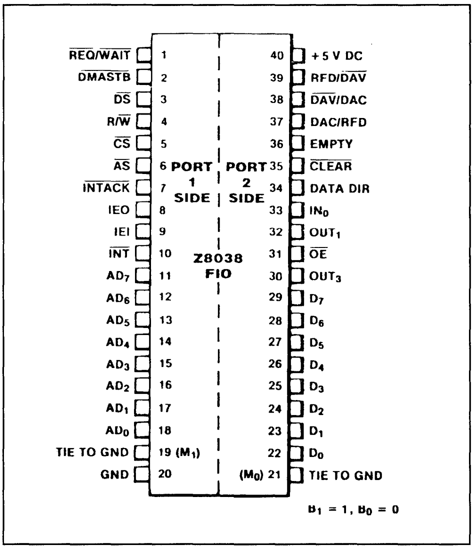
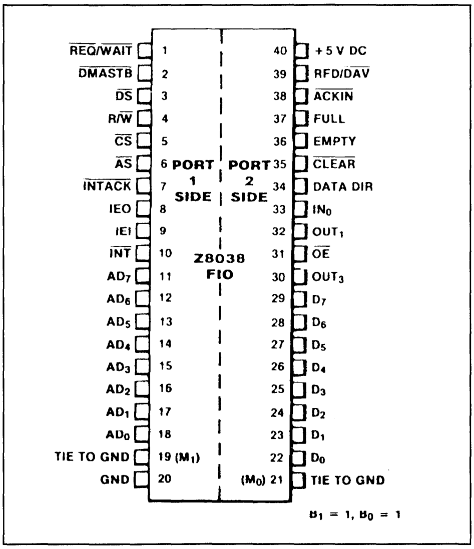


Figure 59 : Z-BUS-Low-Byte to 2-Wire Handshake.



PIN ASSIGNMENTS (continued)

Z-BUS High Byte Mode : Port 1 Side.

Pin Signals	Pin Numbers	Signal Description
AD_0-AD_7 (address/data)	11-18	Multiplexed Bidirectional Address/data Lines Z-BUS Compatible.
$\overline{REQ}/\overline{WAIT}$ (request/wait)	1	Output, Active low. REQUEST (ready) line for DMA transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data transfers.
$\overline{DMASTB}$ (direct memory access strobe)	2	Input, Active low. Strobes DMA data to and from the FIFO Buffer.
$\overline{DS}$ (data strobe)	3	Input, Active low. Provides Timing for Data Transfer to or from FIO.
$R/\overline{W}$ (read/write)	4	Input ; Active high signals CPU read from FIO ; Active low signals CPU write to FIO.
$\overline{CS}$ (chip select)	5	Input, Active Low. Enables FIO. Latched on the Rising Edge of AS.
$\overline{AS}$ (address strobe)	6	Input, Active low. Addresses, $\overline{CS}$ and $\overline{INTACK}$ are sampled while AS is low.
A_0 (address bit 0)	7	Input, Active High. With A_1, A_2 and A_3, Addresses FIO Internal Registers.
A_1 (addresses bit 1)	8	Input, Active high. With A_0, A_2 and A_3, Addresses FIO Internal Registers.
A_2 (addresses bit 2)	9	Input, Active high. With A_0, A_1 and A_3, Addresses FIO Internal Registers.
A_3 (addresses bit 3)	10	Input, Active High. With A_0, A_1, A_3, Addresses FIO Internal Reg

Z-BUS High Byte Mode : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D_0-D_7 (address/data)	29-22	Multiplexed Bidirectional Address/data Lines Z-BUS Compatible.
$\overline{REQ}/\overline{WAIT}$ (request/wait)	39	Output, Active low. REQUEST (ready) line for DMA transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data transfers.
$\overline{DMASTB}$ (direct memory access strobe)	38	Input, Active low. Strobes DMA data to and from the FIFO Buffer.
$\overline{DS}$ (data strobe)	37	Input, Active low. Provides Timing for Transfer of Data to or from FIO.
$R/\overline{W}$ (read/write)	36	Input ; Active high Signals CPU read from FIO ; Active low signals CPU write to FIO.
$\overline{CS}$ (chip select)	35	Input, Active Low. Enables FIO. Latched on the Rising Edge of AS.
$\overline{AS}$ (address strobe)	34	Input, Active low. Addresses, $\overline{CS}$ and $\overline{INTACK}$ are sampled while AS is low.
A_0 (address bit 0)	33	Input, Active High. With A_1, A_2 and A_3, Addresses FIO Internal Registers.
A_1 (addresses bit 1)	32	Input, Active high. With A_0, A_2 and A_3, Addresses FIO Internal Registers.
A_2 (addresses bit 2)	31	Input, Active high. With A_0, A_1 and A_3, Addresses FIO Internal Registers.
A_3 (addresses bit 3)	30	Input, Active High. With A_0, A_1, A_2, Addresses FIO Internal Registers.

PIN ASSIGNMENTS (continued)

Non-Z-BUS Mode : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D0-D7 (data)	29-22	Bidirectional Data Bus
$\overline{REQ}$/$\overline{WAIT}$ (request/wait)	39	Output, Active Low. $\overline{REQUEST}$ (ready) Line for DMA Transfer ; WAIT Line (open-drain) output for Synchronized CPU and FIO Data Transfers.
$\overline{DACK}$ (DMA acknowledge)	38	Input, Active Low. DMA Acknowledge
$\overline{RD}$ (read)	37	Input Active low. Signals CPU read from FIO.
$\overline{WR}$ (write)	36	Input Active Low. Signals CPU write to FIO.
$\overline{CE}$ (chip select)	35	Input, Active Low. Used to Select FIO.
C/$\overline{D}$ (control/data)	34	Input, Active High. Indentifies Control Byte on D_0-D_7 ; Active Low indentifies Data Byte on D_0-D_7.
$\overline{INTACK}$ (interrupt acknowledge)	33	Input, Active Low. Acknowledges an interrupt
IEO (interrupt enable out)	32	Output, Active high. Sends interrupt enable to lower priority device IEI pin.
IEI (interrupt enable in)	31	Input, Active high. Recives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (interrupt)	30	output, Open Drain, Active Low. Signals FIO interrupt to CPU.

Figure 60 : Z-BUS-High-Byte to Z-BUS-High-Byte.

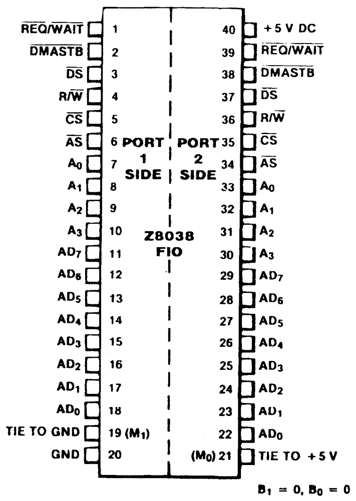
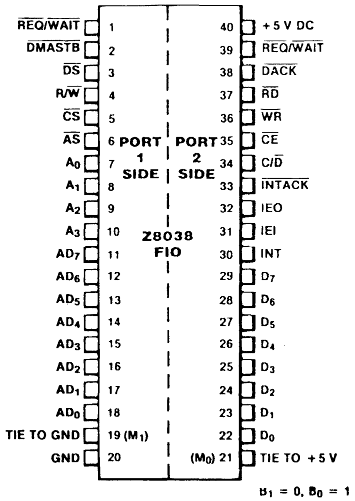


Figure 61 : Z-BUS-High-Byte to Non-Z-BUS.



PIN ASSIGNMENTS (continued)

Z-BUS High Byte Mode : Port 1 Side.

Pin Signals	Pin Numbers	Signal Description
AD0-AD7 (address/data)	11-18	Multiplexed Bidirectional Address/data Lines Z-BUS Compatible.
$\overline{\text{REQ}}/\overline{\text{WAIT}}$ (request/wait)	1	Output, Active low. $\overline{\text{REQUEST}}$ (ready) line for DMA transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data transfers.
$\overline{\text{DMASTB}}$ (direct memory access strobe)	2	Input, Active low. Strobes DMA data to and from the FIFO Buffer.
$\overline{\text{DS}}$ (data strobe)	3	Input, Active low. Provides Timing Transfer of Data to or from FIO.
R/$\overline{\text{W}}$ (read/write)	4	Input ; Active high signals CPU read from FIO ; Active low signals CPU write to FIO.
$\overline{\text{CS}}$ (chip select)	5	Input, Active Low. Enables FIO. Latched on the Rising Edge of AS.
$\overline{\text{AS}}$ (address strobe)	6	Input, Active low. Addresses, $\overline{\text{CS}}$ and $\overline{\text{INTACK}}$ are sampled while AS is low.
A_0 (address bit 0)	7	Input, Active High. With A_1, A_2 and A_3, Addresses FIO Internal Registers.
A_1 (addresses bit 1)	8	Input, Active high. With A_0, A_2 and A_3, Addresses FIO Internal Registers.
A_2 (addresses bit 2)	9	Input, Active high. With A_0, A_1 and A_3, Addresses FIO Internal Registers.
A_3 (address bit 3)	10	Input, Active High. With A_0, A_1, A_2, Addresses FIO Internal Registers.

PIN ASSIGNMENTS (continued)

3-Wire Handshake : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D_0-D_7 (data)	29-22	Bidirectional Data Bus
RFD/$\overline{\text{DAV}}$ (ready for data/data available)	39	Output, RFD Active High. Signals peripherals that FIO is ready to receive data. DAV active low signals that FIO is ready to send data to peripherals.
$\overline{\text{DAV}}$/DAC (data available/data accepted)	38	Input ; $\overline{\text{DAV}}$ (active low) signals that data is valid on bus. DAC (active high) signals that ouput data is accepted by peripherals.
DAC/RFD (data accepted/ready for data)	37	Direction Controlled by Internal Programming. Both Active high. DAC (an output) signals that FIO has received data from peripheral ; RFD (an input) signals that the listeners are ready for data.
EMPTY	36	Output, Input, Open Drain, Active High. Signals that FIFO buffer is empty.
$\overline{\text{CLEAR}}$	35	Programmable Input or Output, Active Low. Clears All Data from FIFO Buffer.
DATA DIR (data direction)	34	Programmable Input or Output. Active High Signals Data Input to Port 2 ; Low Signals Data Output from port 2.
IN_0	33	Input Line to D_0 of Control Register 3
OUT_1	32	Output Line from D_1 of Control Register 3
$\overline{\text{OE}}$ (output enable)	31	Input, Active low. When low, enables bus drivers. When high, floats bus drivers at high impedance.
OUT_3	30	Output line from D_3 of Control Register 3

PIN ASSIGNMENTS (continued)

2-Wire Handshake : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D0-D7 (data)	29-22	Bidirectional Data Bus
RFD/$\overline{\text{DAV}}$ (ready for data/data available)	39	Output, RFD Active High. Signals peripherals that FIO is ready to receive data. DAV active low signals that FIO is ready to send data to peripherals.
$\overline{\text{ACKIN}}$ (acknowledge input)	38	Input, Active Low. Signals FIO that output data is received by peripherals or that input data is valid.
FULL	37	Output, Input, Open Drain, Active High. Signals that FIO buffer is full.
EMPTY	36	output, Input, Open Drain, Active High. Signals that FIFO buffer is empty.
CLEAR	35	Programmable Input or Output, Active Low. Clears All Data from FIFO Buffer.
DATA DIR (data direction)	34	Programmable Input or Output. Active High Signals Data Input to Port 2 ; Low Signals Data Output from port 2.
IN_0	33	Input Line to D_0 of Control Register 3
OUT_0	32	Output Line From D_1 of Control Register 3
$\overline{\text{OE}}$ (output enable)	31	Input, Active low. When low, enables bus drivers. When high, floats bus drivers at high impedance.
OUT_3	30	Output line from D_3 of Control Register 3

Figure 62 : Z-BUS-High-Byte to 3-Wire Handshake.

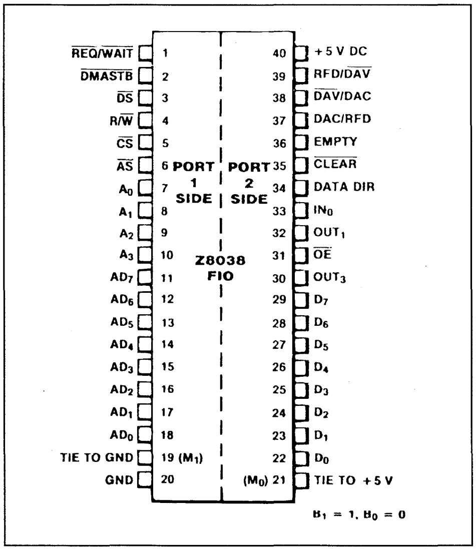
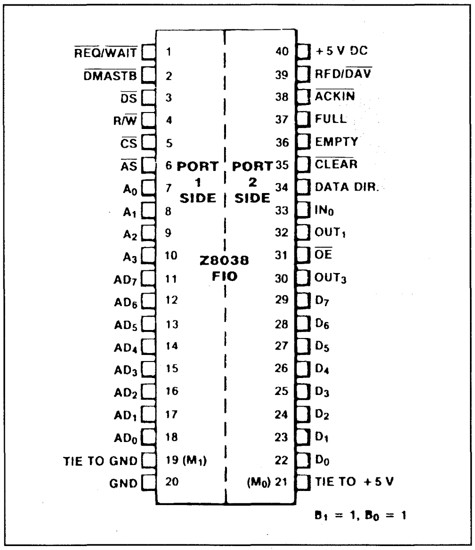


Figure 63 : Z-BUS-High-Byte to 2-Wire Handshake.



PIN ASSIGNMENTS (continued)

Z-BUS High Byte Mode : Port 1 Side.

Pin Signals	Pin Numbers	Signal Description
AD0-AD7 (address/data)	11-18	Bidirectional Data Bus
$\overline{\text{REQ}}/\overline{\text{WAIT}}$ (request/wait)	1	Output, Active Low. $\overline{\text{REQUEST}}$ (ready) line for DMA Transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data Transfers.
$\overline{\text{DACK}}$ (DMA acknowledge)	2	Input, Active Low. DMA Acknowledge
$\overline{\text{RD}}$ (read)	3	Input, Active Low. Signals CPU read from FIO.
$\overline{\text{WR}}$ (write)	4	Input, Active Low. Signals CPU write to FIO.
$\overline{\text{CE}}$ (chip select)	5	Input, Active low. Used to Select FIO.
C/$\overline{\text{D}}$ (control/data)	6	Input, Active High. Indentifies Control Byte on D_0-D_7 ; Active Low Indentifies Data Byte on D_0-D_7.
$\overline{\text{INTACK}}$ (interrupt acknowledge)	7	Input, Active Low. Acknowledges an Interrupt.
IEO (interrupt enable out)	8	Output, Active High. Sends interrupt enable to lower priority device IEI pin.
IEI (interruppt enable in)	9	Input, Active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{\text{INT}}$ (interrupt)	10	Output, Open Drain, Active Low. Signals FIO interrupt to CPU.

Z-BUS Low Byte Mode : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D_0-D_7 (address/data)	29-22	Multiplexed Bidirectional Address/data Lines, Z-BUS Compatible
$\overline{\text{REQ}}/\overline{\text{WAIT}}$ (request/wait)	39	Output, Active Low. $\overline{\text{REQUEST}}$ (ready) line for DMA Transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data Transfers.
$\overline{\text{DMASTB}}$ (direct Memory Access Strobe)	38	Input, Active Low. Strobes DMA Data to and from the FIFO Buffer.
$\overline{\text{DS}}$ (data strobe)	37	Input, Active Low. Provides Timing for Data Transfer to or from FIO.
R/$\overline{\text{W}}$ (read/write)	36	Input ; Active high signals CPU read from FIO ; Active low signals CPU write to FIO.
$\overline{\text{CS}}$ (chip select)	35	Input, Active Low. Enables FIO. Latched on the Rising Edge of AS.
$\overline{\text{AS}}$ (address strobe)	34	Input, Active Low. Addresses, $\overline{\text{CS}}$ and $\overline{\text{INTACK}}$ sampled while AS Low.
$\overline{\text{INTACK}}$ (interrupt acknowledge)	33	Input, Active Low. Acknowledges an Interrupt. Latched on the Rising Edge of AS.
IEO (interrupt enable out)	32	Output, Active High. Sends interrupt enable to lower priority device IEI pin.
IEI (interrupt enable in)	31	Input, Active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{\text{INT}}$ (interrupt)	30	Output, Open Drain, Active Low. Signals FIO interrupt request to CPU.

PIN ASSIGNMENTS (continued)

Non-Z-BUS Mode : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D_0-D_7 (data)	29-22	Bidirectional Data Bus
$\overline{REQ}/\overline{WAIT}$ (request/wait)	39	Output, Active Low. $\overline{REQUEST}$ (ready) Line for DMA Transfer ; WAIT Line (open-drain) output for Synchronized CPU and FIO Data Transfers.
$\overline{DACK}$ (DMA acknowledge)	38	Input, Active Low. DMA Acknowledge
$\overline{RD}$ (read)	37	Input Active low. Signals CPU read from FIO.
$\overline{WR}$ (write)	36	Input Active Low. Signals CPU write to FIO.
$\overline{CE}$ (chip select)	35	Input, Active Low. Used to Select FIO.
C/$\overline{D}$ (control/data)	34	Input, Active High. Identifies Control Byte on D_0-D_7 ; Active Low identifies Data Byte on D_0-D_7.
$\overline{INTACK}$	33	Input, Active Low. Acknowledges an interrupt
IEO (interrupt enable out)	32	Output, Active high. Sends interrupt enable to lower priority device IEI pin.
IEI (interrupt enable in)	31	Input, Active high. Recives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (interrupt)	30	Output, Open Drain, Active Low. Signals FIO interrupt to CPU.

Figure 64 : Non-Z-BUS to Z-BUS-Low-Byte.

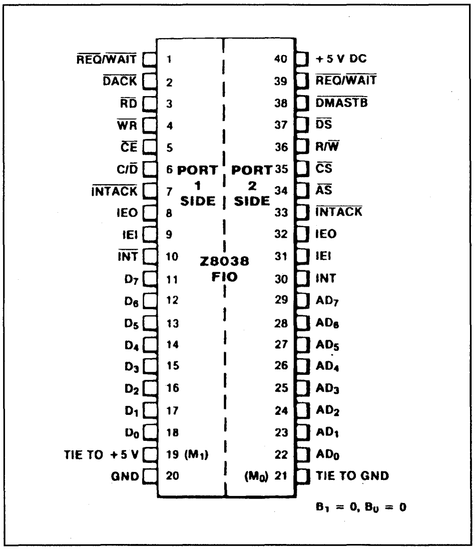
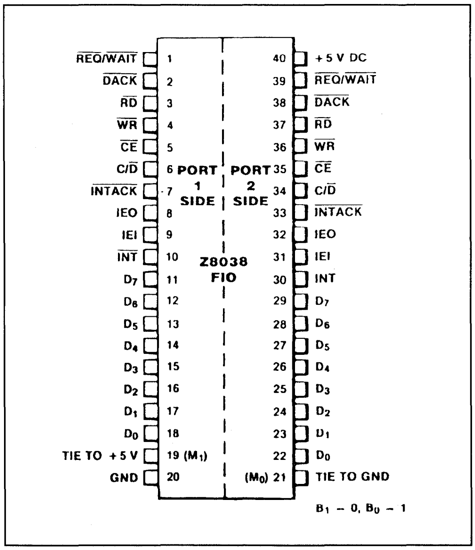


Figure 65 : Non-Z-BUS to Non-Z-BUS.



PIN ASSIGNMENTS (continued)

Non-Z-BUS Mode : Port 1 Side.

Pin Signals	Pin Numbers	Signal Description
D0-D7 (data)	11-18	Bidirectional Data Bus
$\overline{REQ}/\overline{WAIT}$ (request/wait)	1	Output, Active Low. $\overline{REQUEST}$ (ready) line for DMA Transfer ; WAIT Line (open-drain) Output for Synchronized CPU and FIO Data Transfers.
$\overline{DACK}$ (DMA acknowledge)	2	Input, Active Low. DMA Acknowledge
$\overline{RD}$ (read)	3	Input, Active Low. Signals CPU read from FIO.
$\overline{WR}$ (write)	4	Input, Active Low. Signals CPU write to FIO.
$\overline{CE}$ (chip select)	5	Input, Active low. Used to Select FIO.
$C/\overline{D}$ (control/data)	6	Input, Active High. Indentifies Control Byte on D_0-D_7 ; Active Low Indentifies Data Byte on D_0-D_7.
INTACK (interrupt acknowledge)	7	Input, Active Low. Acknowledges an Interrupt.
IEO (interrupt enable out)	8	Output, Active High. Sends interrupt enable to lower priority device IEI pin.
IEI (interruppt enable in)	9	Input, Active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (interrupt)	10	Output, Open Drain, Active Low. Signals FIO interrupt to CPU.

3-Wire Handshake : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D0-D7 (data)	29-22	Bidirectional Data Bus
RFD/$\overline{DAV}$ (ready for data/data available)	39	Output, RFD Active High. Signals peripherals that FIO is ready to receive data. DAV active low signals that FIO is ready to send data to peripherals.
$\overline{DAV}$/DAC (data available/data accepted)	38	Input ; $\overline{DAV}$ (active low) signals that data is valid on bus. DAC (active high) signals that ouput data is accepted by peripherals.
DAC/RFD (data accepted/ready for data)	37	Direction Controlled by Internal Programming. Both Active high. DAC (an output) signals that FIO has received data from peripheral ; RFD (an input) signals that the listeners are ready for data.
EMPTY	36	Output, Input, Open Drain, Active High. Signals that FIFO buffer is empty.
$\overline{CLEAR}$	35	Programmable Input or Output, Active Low. Clears All Data from FIFO Buffer.
DATA DIR (data direction)	34	Programmable Input or Output. Active High Signals Data Input to Port 2 ; Low Signals Data Output from port 2.
IN_0	33	Input Line to D_0 of Control Register 3
OUT_1	32	Output Line from D_1 of Control Register 3
$\overline{OE}$ (output enable)	31	Input, Active low. When low, enables bus drivers. When high, floats bus drivers at high impedance.
OUT_3	30	Output line from D_3 of Control Register 3

PIN ASSIGNMENTS (continued)

2-Wire Handshake : Port 2 Side.

Pin Signals	Pin Numbers	Signal Description
D_0-D_7 (data)	29-22	Bidirectional Data Bus
RFD/$\overline{\text{DAV}}$ (ready for data/data available)	39	Output, RFD Active High. Signals peripherals that FIO is ready to receive data. DAV active low signals that FIO is ready to send data to peripherals.
$\overline{\text{ACKIN}}$ (acknowledge input)	38	Input, Active Low. Signals FIO that output data is received by peripherals or that input data is valid.
FULL	37	Output, Input, Open Drain, Active High. Signals that FIO buffer is full.
EMPTY	36	Output, Input, Open Drain, Active High. Signals that FIFO buffer is empty.
$\overline{\text{CLEAR}}$	35	Programmable Input or Output, Active Low. Clears All Data from FIFO Buffer.
DATA DIR (data direction)	34	Programmable Input or Output. Active High Signals Data Input to Port 2 ; Low Signals Data Output from port 2.
IN_0	33	Input Line to D_0 of Control Register 3
OUT_0	32	Output Line From D_1 of Control Register 3
$\overline{\text{OE}}$ (output enable)	31	Input, Active low. When low, enables bus drivers. When high, floats bus drivers at high impedance.
OUT_3	30	Output line from D_3 of Control Register 3

Figure 66 : Non-Z-BUS to 3-Wire Handshake.

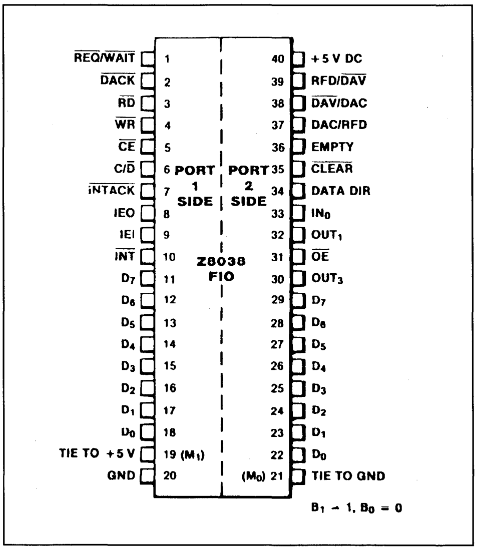
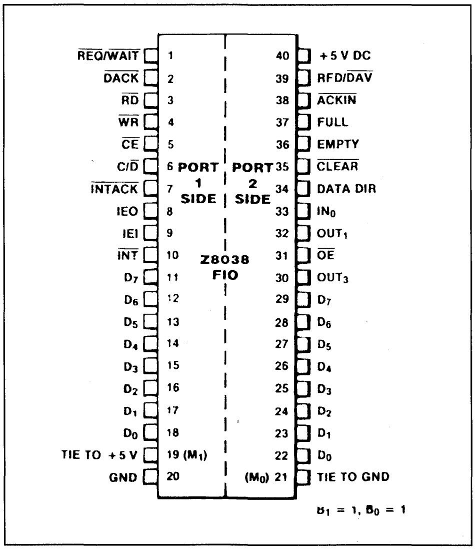
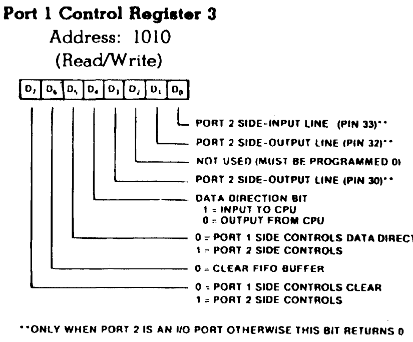


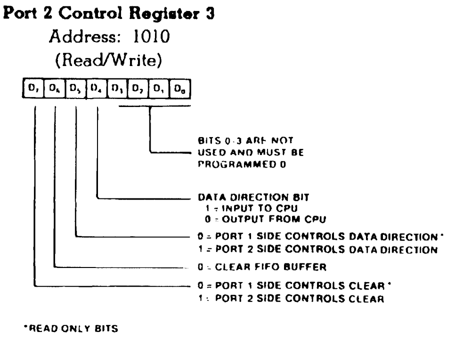
Figure 67 : Non-Z-BUS to 2-Wire Handshake.

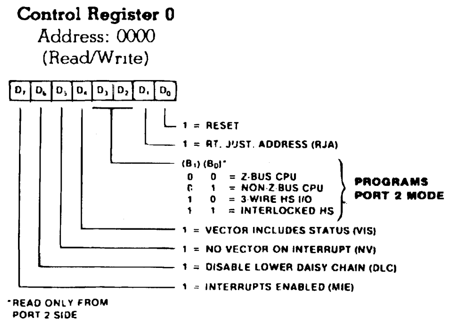


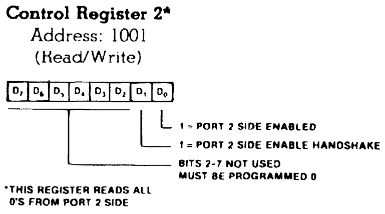
REGISTERS

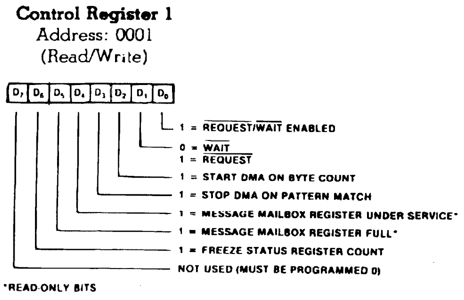
Figure 68 : Control Registers.


Port 1 Control Register 3
Address: 1010
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
PORT 2 SIDE-INPUT LINE (PIN 33)**
PORT 2 SIDE-OUTPUT LINE (PIN 32)**
NOT USED (MUST BE PROGRAMMED 0)
PORT 2 SIDE-OUTPUT LINE (PIN 30)**
DATA DIRECTION BIT
1 = INPUT TO CPU
0 = OUTPUT FROM CPU
0 = PORT 1 SIDE CONTROLS DATA DIRECTION
1 = PORT 2 SIDE CONTROLS
0 = CLEAR FIFO BUFFER
0 = PORT 1 SIDE CONTROLS CLEAR
1 = PORT 2 SIDE CONTROLS
**ONLY WHEN PORT 2 IS AN I/O PORT OTHERWISE THIS BIT RETURNS 0


Port 2 Control Register 3
Address: 1010
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
BITS 0-3 ARE NOT USED AND MUST BE PROGRAMMED 0
DATA DIRECTION BIT
1 = INPUT TO CPU
0 = OUTPUT FROM CPU
0 = PORT 1 SIDE CONTROLS DATA DIRECTION*
1 = PORT 2 SIDE CONTROLS DATA DIRECTION
0 = CLEAR FIFO BUFFER
0 = PORT 1 SIDE CONTROLS CLEAR*
1 = PORT 2 SIDE CONTROLS CLEAR
*READ ONLY BITS

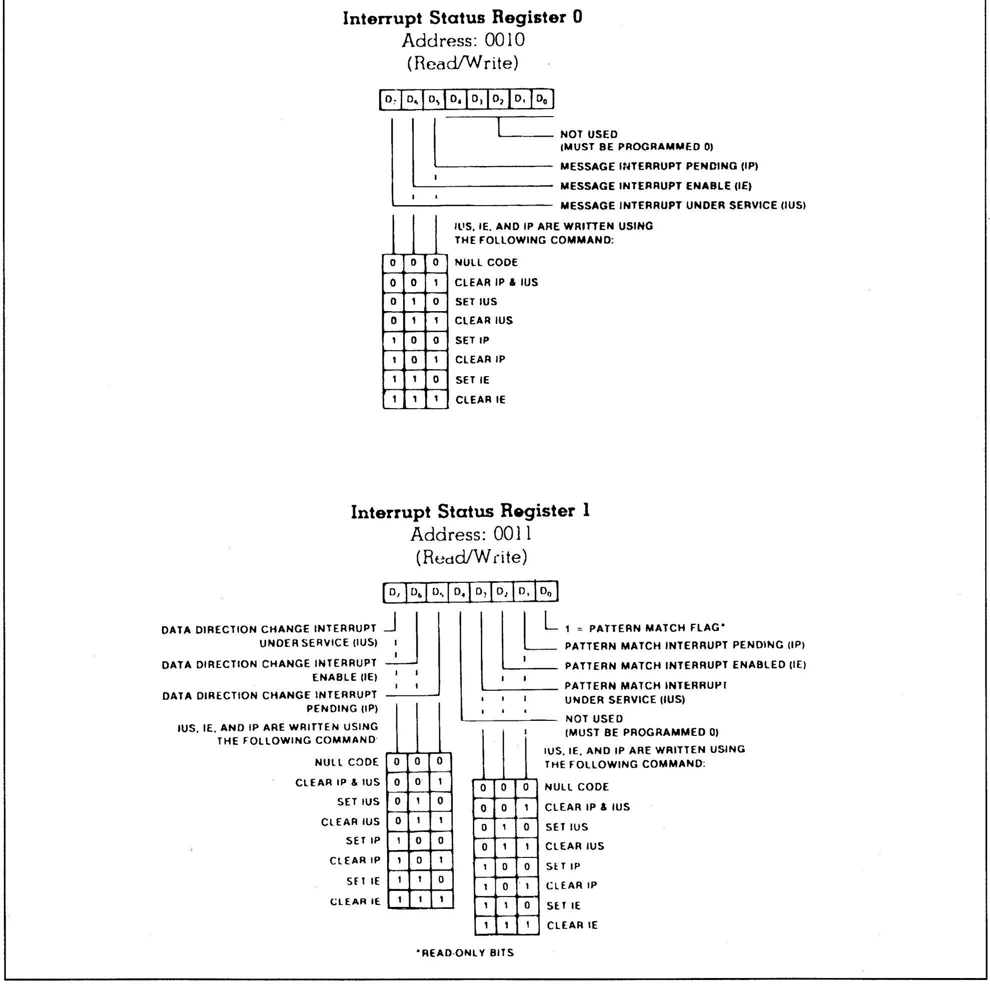

Control Register 0
Address: 0000
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
1 = RESET
1 = RT. JUST. ADDRESS (RJA)
(B1) (B0)*
0 0 = Z-BUS CPU
0 1 = NON-Z-BUS CPU
1 0 = 3-WIRE HS I/O
1 1 = INTERLOCKED HS
PROGRAMS PORT 2 MODE
1 = VECTOR INCLUDES STATUS (VIS)
1 = NO VECTOR ON INTERRUPT (NV)
1 = DISABLE LOWER DAISY CHAIN (DLC)
1 = INTERRUPTS ENABLED (MIE)
*READ ONLY FROM PORT 2 SIDE


Control Register 2*
Address: 1001
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
1 = PORT 2 SIDE ENABLED
1 = PORT 2 SIDE ENABLE HANDSHAKE
BITS 2-7 NOT USED MUST BE PROGRAMMED 0
*THIS REGISTER READS ALL 0'S FROM PORT 2 SIDE


Control Register 1
Address: 0001
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
1 = REQUEST/WAIT ENABLED
0 = WAIT
1 = REQUEST
1 = START DMA ON BYTE COUNT
1 = STOP DMA ON PATTERN MATCH
1 = MESSAGE MAILBOX REGISTER UNDER SERVICE*
1 = MESSAGE MAILBOX REGISTER FULL*
1 = FREEZE STATUS REGISTER COUNT
NOT USED (MUST BE PROGRAMMED 0)
*READ ONLY BITS

REGISTERS (continued)

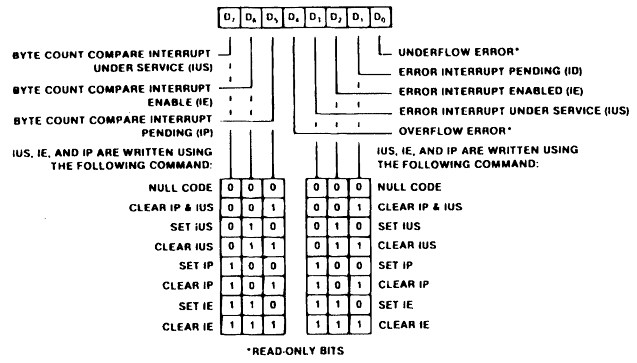
Figure 69 : Interrupt Status Registers.


Interrupt Status Register 0
Address: 0010
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
NOT USED (MUST BE PROGRAMMED 0)
MESSAGE INTERRUPT PENDING (IP)
MESSAGE INTERRUPT ENABLE (IE)
MESSAGE INTERRUPT UNDER SERVICE (IUS)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING COMMAND:
0 0 0 NULL CODE
0 0 1 CLEAR IP & IUS
0 1 0 SET IUS
0 1 1 CLEAR IUS
1 0 0 SET IP
1 0 1 CLEAR IP
1 1 0 SET IE
1 1 1 CLEAR IE
Interrupt Status Register 1
Address: 0011
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
DATA DIRECTION CHANGE INTERRUPT UNDER SERVICE (IUS)
DATA DIRECTION CHANGE INTERRUPT ENABLE (IE)
DATA DIRECTION CHANGE INTERRUPT PENDING (IP)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING COMMAND
1 = PATTERN MATCH FLAG*
PATTERN MATCH INTERRUPT PENDING (IP)
PATTERN MATCH INTERRUPT ENABLED (IE)
PATTERN MATCH INTERRUPT UNDER SERVICE (IUS)
NOT USED (MUST BE PROGRAMMED 0)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING COMMAND:
0 0 0 NULL CODE
0 0 1 CLEAR IP & IUS
0 1 0 SET IUS
0 1 1 CLEAR IUS
1 0 0 SET IP
1 0 1 CLEAR IP
1 1 0 SET IE
1 1 1 CLEAR IE
*READ-ONLY BITS

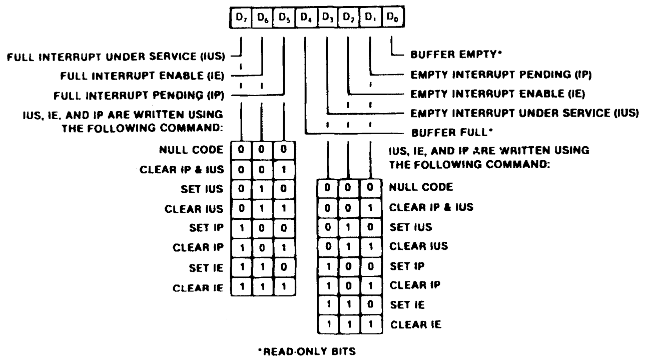
REGISTERS (continued)

Figure 70 : Interrupt Status Registers (continued).

Interrupt Status Register 2
Address: 0100
(Read/Write)


D7 D6 D5 D4 D3 D2 D1 D0
BYTE COUNT COMPARE INTERRUPT UNDER SERVICE (IUS)
BYTE COUNT COMPARE INTERRUPT ENABLE (IE)
BYTE COUNT COMPARE INTERRUPT PENDING (IP)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING COMMAND:
NULL CODE 0 0 0
CLEAR IP & IUS 0 0 1
SET IUS 0 1 0
CLEAR IUS 0 1 1
SET IP 1 0 0
CLEAR IP 1 0 1
SET IE 1 1 0
CLEAR IE 1 1 1
UNDERFLOW ERROR*
ERROR INTERRUPT PENDING (ID)
ERROR INTERRUPT ENABLED (IE)
ERROR INTERRUPT UNDER SERVICE (IUS)
OVERFLOW ERROR*
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING COMMAND:
0 0 0 NULL CODE
0 0 1 CLEAR IP & IUS
0 1 0 SET IUS
0 1 1 CLEAR IUS
1 0 0 SET IP
1 0 1 CLEAR IP
1 1 0 SET IE
1 1 1 CLEAR IE
*READ-ONLY BITS

Interrupt Status Register 3
Address: 0101
(Read/Write)



REGISTERS (continued)

Figure 71 : Count Register.

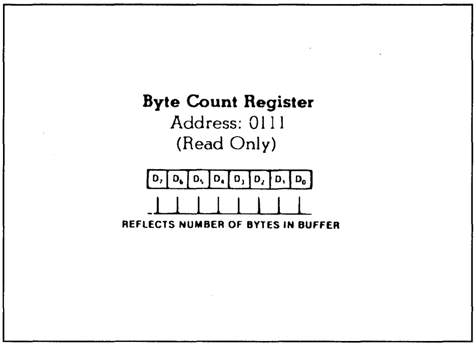


Figure 72 : Interrupt Vector Register.

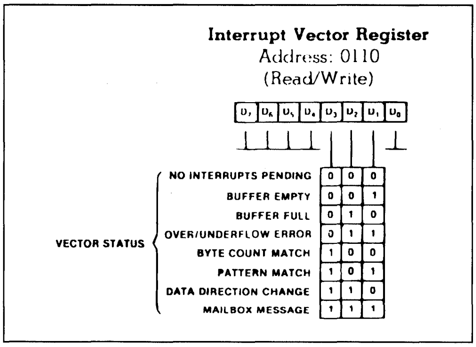


Figure 73 : Pattern Match Register.

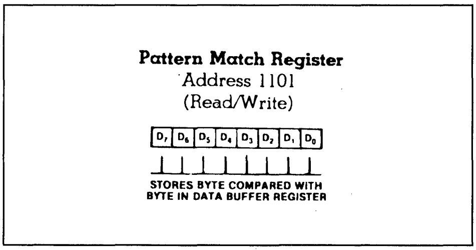


Figure 74 : Pattern Mask Register.

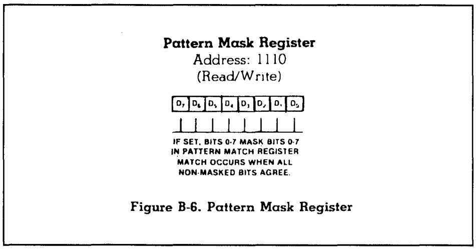

Pattern Mask Register
Address: 1110
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
IF SET, BITS 0-7 MASK BITS 0-7 IN PATTERN MATCH REGISTER MATCH OCCURS WHEN ALL NON-MASKED BITS AGREE.
Figure B-6. Pattern Mask Register

Figure 75 : Data Buffer Register.

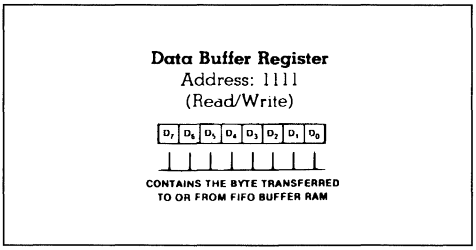


Figure 76 : Byte Count Comparation Register.

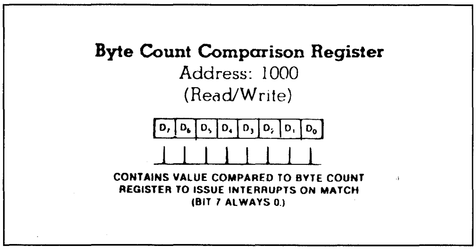

Byte Count Comparison Register
Address: 1000
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
CONTAINS VALUE COMPARED TO BYTE COUNT REGISTER TO ISSUE INTERRUPTS ON MATCH (BIT 7 ALWAYS 0.)

Figure 77 : Message Out Register.

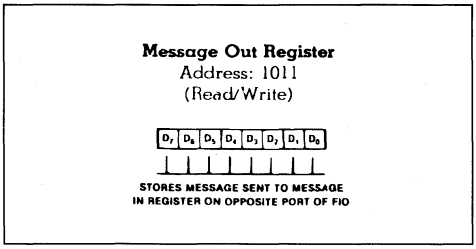
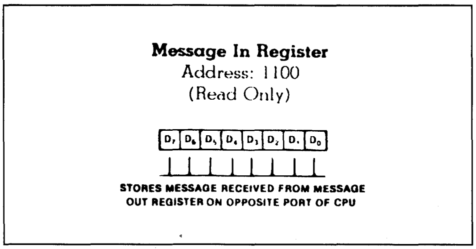


Figure 78 : Message In Register.


Message In Register
Address: 1100
(Read Only)
D7 D6 D5 D4 D3 D2 D1 D0
STORES MESSAGE RECEIVED FROM MESSAGE OUT REGISTER ON OPPOSITE PORT OF CPU

ELECTRICAL INFORMATION

ABSOLUTE MAXIMUM RATING

Symbol	Parameter	Test Conditions	Unit
V_I	Voltages on All Inputs and Outputs with Respect to GND	– 0.3 to + 7.0	V
T_A	Operating Ambient Temperature	0 to + 70	°C
T_{stg}	Storage Temperature	– 65 to + 150	°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only ; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows :

- $+4.75\ V \leq VCC \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

Figure 79 : Standard Test Conditions.

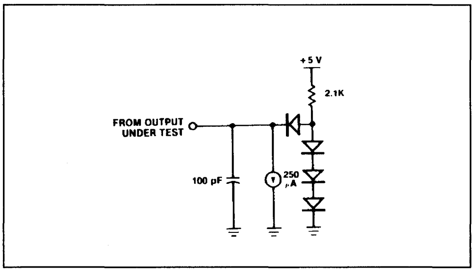
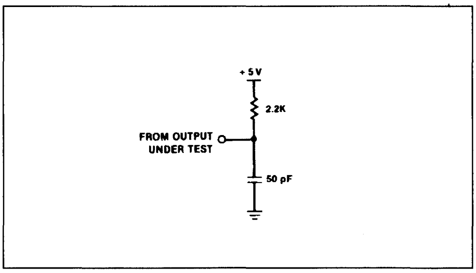


Figure 80 : Open-Drain Test Load.



DC CHARACTERISTICS

Symbol	Parameter	Test Conditions	Min.	Max.	Unit
V_{IH}	Input High Voltage		2.0	V_{CC}– + 0.3	V
V_{IL}	Input Low Voltage		– 0.3	0.8	V
V_{OH}	Output High Voltage	I_{OH} = – 250 mA	2.4		V
V_{OL}	Output Low Voltage	I_{OL} = + 2.0 mA I_{OL} = 3.2 mA		0.4 0.5	V V
I_{IL}	Input Leakage	0.4 = V_{IN} = + 2.4 V		± 10	µA
I_{IL} (M_0, M_1)	Mode Pin Leakage	0.4 = V_{IN} = + 2.4 V		± 100	µA
I_{OL}	Output Leakage	0.4 = V_{OUT} = + 2.4 V		± 10	µA
I_{CC}	V_{CC} Supply Current			250	mA

V_{CC} = 5 V ± 5 % unless otherwise specified, over specified temperature range.

CAPACITANCE

Symbol	Parameter	Test Conditions	Min.	Max.	Unit
C_{IN}	Input Capacitance	Unmeasured Pins Returned to Ground		10	pF
C_{OUT}	Output Capacitance			15	pF
$C_{I/O}$	Bidirectional Capacitance			20	pF
Inputs					
tr	Any Input Rise Time			100	ns
tf	Any Input Fall Time			100	ns

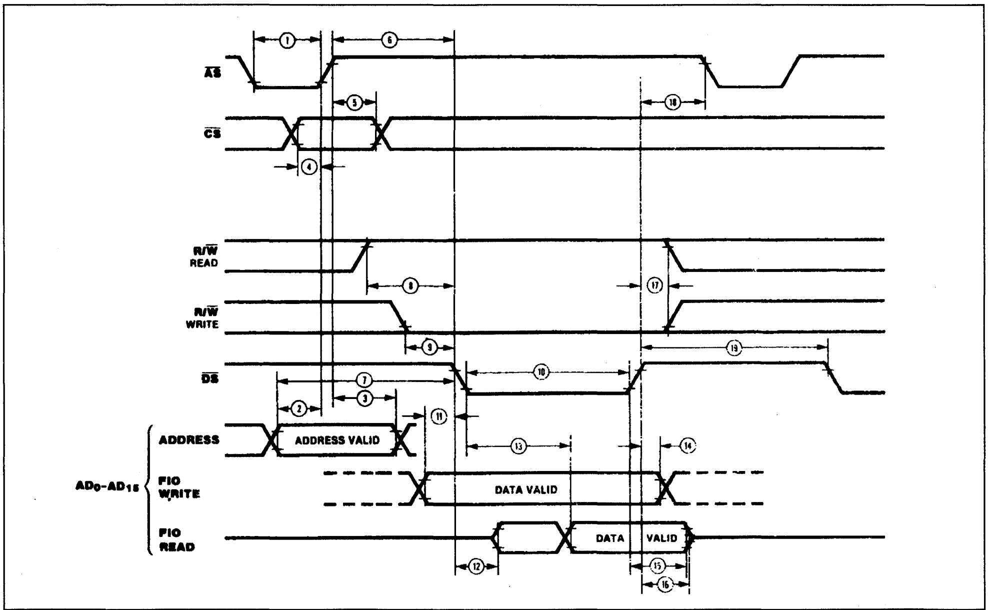
f = 1 MHz, over specified temperature range.

AC CHARACTERISTICS

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	TwAS	$\overline{AS}$ Low Width	70		50		1
2	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	30		10		1
3	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		30		1
4	TsCSO(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Setup Time	0		0		1
5	ThCSO(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Hold Time	60		40		1
6	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60		40		1
7	TsA(DS)	Address to $\overline{DS}$ ↓ (with $\overline{AS}$ ↑ to $\overline{DS}$ ↓ = 60 ns)	120		100		
8	TsRWR(DS)	R/$\overline{W}$ (read) to $\overline{DS}$ ↓ Setup Time	100		80		
9	TsRWW(DS)	R/$\overline{W}$ (write) to $\overline{DS}$ ↓ Setup Time	0		0		
10	TwDS	$\overline{DS}$ Low Width	390		250		
11	TsDW(DSf)	Write Data to $\overline{DS}$ ↓ Setup Time	30		20		
12	TdDS(DRV)	$\overline{DS}$ (read) ↓ to Address Data Bus Driven	0		0		
13	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		250		180	
14	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		20		
15	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data not Valid Delay	0		0		
16	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70		45	2
17	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	55		40		
18	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		25		
19	Trc	Valid Access Recovery Time	1000		650		3

Notes : 1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5 V from steady state with minimum ac load and maximum dc load.
3. This is a delay from $\overline{DS}$ of one FIO access to $\overline{DS}$ of another FIO access (either read or write).
* All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0". All timings are preliminary and subject to change.
† Units in nanoseconds (ns).

Figure 81 : Z-BUS CPU Interface Timing.

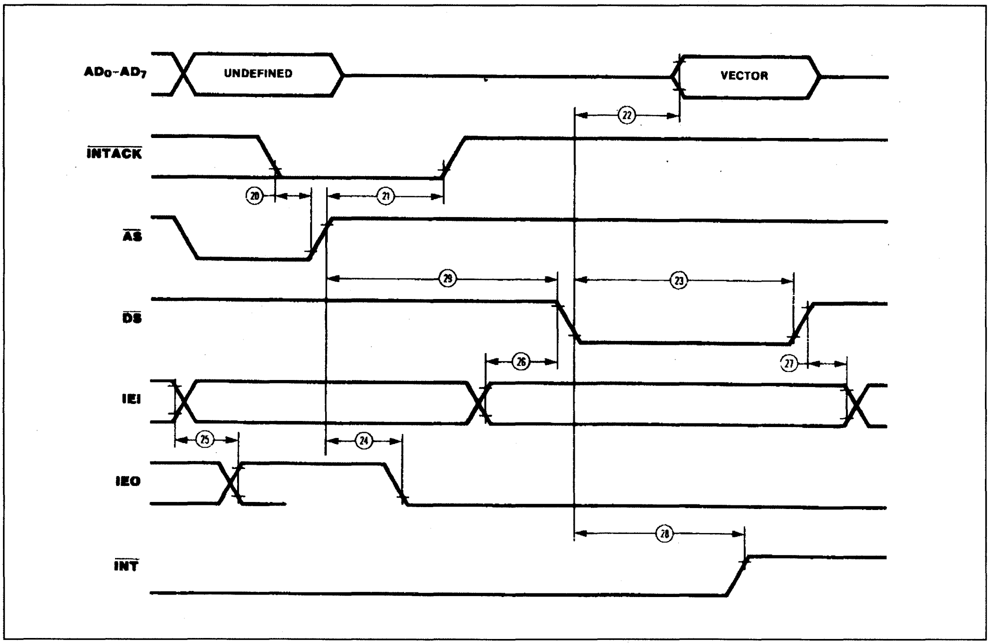


AC CHARACTERISTICS (Continued)

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
20	TsIA(AS)	$\overline{\text{INTACK}}$ to $\overline{\text{AS}}$ ↑ Setup Time	0		0		
21	ThIA(AS)	$\overline{\text{INTACK}}$ to $\overline{\text{AS}}$ ↑ Hold Time	250		250		
22	TsDSA(DR)	$\overline{\text{DS}}$ (acknowledge)↓ to Read Data Valid Delay		250		180	
23	TwDSA	$\overline{\text{DS}}$ (acknowledge) Low Width	390		250		
24	TdAS(IEO)	$\overline{\text{AS}}$ ↑ to IEO ↓ Delay ($\overline{\text{INTACK}}$ cycle)		350		250	4
25	TdIEI(IEO)	IEI to IEO Delay		150		100	4
26	TsIEI(DSA)	IEI to $\overline{\text{DS}}$ (acknowledge) ↓ Setup Time	100		70		
27	ThIEI(DSA)	IEI to $\overline{\text{DS}}$ (acknowledge) ↑ Hold Time	50		30		4
28	TdDS(INT)	$\overline{\text{DS}}$ ($\overline{\text{INTACK}}$ cycle) to $\overline{\text{INT}}$ Delay		900		800	
29	TdDCST	Interrupt Daisy Chain Settle Time					4

Notes : 4. The parameters for the devices in any particular daisy chain must meet the following constraint : the delay from $\overline{\text{AS}}$ to $\overline{\text{DS}}$ must be greater than the sum of TdAS(IEO) for the highest priority peripheral, TsIEI(DSA) for the lowest priority peripheral and TdIEI(IEO) for each peripheral, separating them in the chain.
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

Figure 82 : Z-BUS CPU Interrupt Acknowledge Timing.



AC CHARACTERISTICS (Continued)

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
30	TdMW(INT)	Message Write to $\overline{\text{INT}}$ Delay		1		1	5
31	TdDC(INT)	Data Direction Change to $\overline{\text{INT}}$ Delay		1		1	6
32	TdPMW(INT)	Pattern Match to $\overline{\text{INT}}$ Delay (write case)		1		1	
33	TdPMR(INT)	Pattern Match (read case) to $\overline{\text{INT}}$ Delay		1		1	
34	TdSC(INT)	Status Compare to $\overline{\text{INT}}$ Delay		1		1	6
35	TdER(INT)	Error to $\overline{\text{INT}}$ Delay		1		1	
36	TdEM(INT)	Empty to $\overline{\text{INT}}$ Delay		1		1	6
37	TdFL(INT)	Full to $\overline{\text{INT}}$ Delay		1		1	6
38	TdAS(INT)	$\overline{\text{AS}}$ to $\overline{\text{INT}}$ Delay					

Notes : 5. Write is from the other side of FIO.
6. Write can be from either side, depending on programming of FIO.
* Timing are preliminary and subject to change.
† Units equal to AS Cycles + ns.

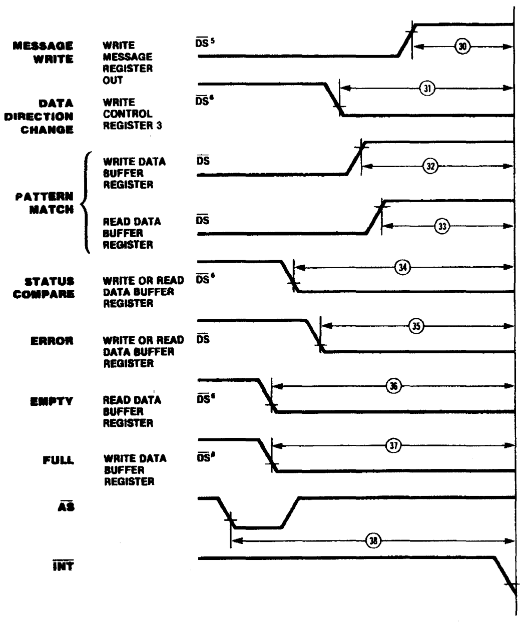

MESSAGE WRITE
WRITE MESSAGE REGISTER OUT
DS 5
30
31
DATA DIRECTION CHANGE
WRITE CONTROL REGISTER 3
DS 6
PATTERN MATCH
WRITE DATA BUFFER REGISTER
DS
32
READ DATA BUFFER REGISTER
DS
33
34
STATUS COMPARE
WRITE OR READ DATA BUFFER REGISTER
DS 6
35
ERROR
WRITE OR READ DATA BUFFER REGISTER
DS
36
EMPTY
READ DATA BUFFER REGISTER
DS 6
37
FULL
WRITE DATA BUFFER REGISTER
DS 6
AS
38
INT

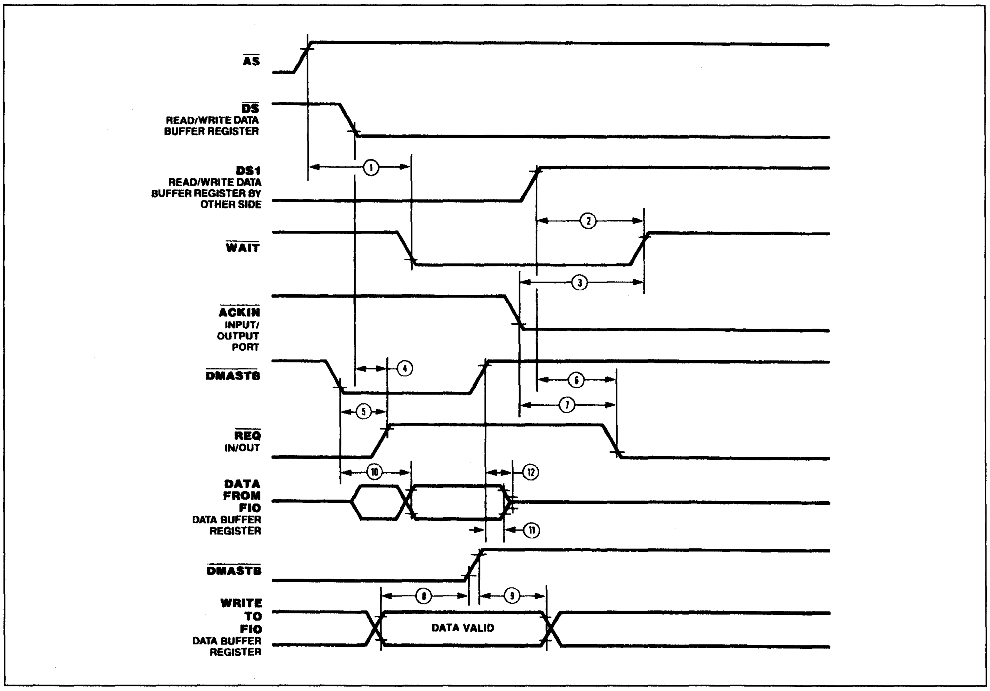
Figure 83 : Z-BUS Interrupt Timing.

AC CHARACTERISTICS

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	TdDS(WAIT)	$\overline{AS}$ ↑ to $\overline{WAIT}$ ↓ Delay		190		160	
2	TdDSI(WAIT)	$\overline{DSI}$ ↑ to $\overline{WAIT}$ ↑ Delay		1000		1000	
3	TdACK(WAIT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ ↑ Delay		1000		1000	1
4	TdDS(REQ)	$\overline{DS}$ ↓ to $\overline{REQ}$ ↑ Delay		350		300	
5	TdDMA(REQ)	$\overline{DMASTB}$ ↓ to $\overline{REQ}$ ↑ Delay		350		300	
6	TdDSI(REQ)	$\overline{DSI}$ ↑ to $\overline{REQ}$ ↓ Delay		1000		1000	
7	TdACK(REQ)	$\overline{ACKIN}$ ↓ to $\overline{REQ}$ ↓ Delay		1000		1000	
8	TdSU(DMA)	Data Setup Time to $\overline{DMASTB}$	200		150		
9	TdH(DMA)	Data Hold Time to $\overline{DMASTB}$	30		20		
10	TdDMA(DR)	$\overline{DMASTB}$ ↓ to Valid Data		150		100	
11	TdDMA(DRH)	$\overline{DMASTB}$ ↑ to Data not Valid	0		0		
12	TdDMA(DR2)	$\overline{DMASTB}$ ↑ to Data Bus Float		70		45	

Notes : 1. The delay is from $\overline{DAV}$ for 3-Wire Input Handshake. The delay is from DAC for 3-Wire Handshake.
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

Figure 84 : Z-BUS Request/Wait Timing.

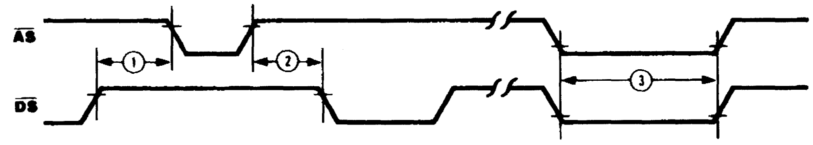

AS
DS READ/WRITE DATA BUFFER REGISTER
DS1 READ/WRITE DATA BUFFER REGISTER BY OTHER SIDE
WAIT
ACKIN INPUT/ OUTPUT PORT
DMASTB
REQ IN/OUT
DATA FROM FIO DATA BUFFER REGISTER
DMASTB
WRITE TO FIO DATA BUFFER REGISTER
DATA VALID
1
2
3
4
5
6
7
8
9
10
11
12

AC CHARACTERISTICS

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	TdDSQ(AS)	Delay from $\overline{DS}$ ↑ to $\overline{AS}$ ↓ for no Reset	40		20		
2	TdASQ(DS)	Delay for $\overline{AS}$ ↑ to $\overline{DS}$ ↓ for no Reset	50		30		
3	Tw(AS + DS)	Minimum Width of $\overline{AS}$ and $\overline{DS}$ Both Low for Reset	500		350		1

Notes : 1. Internal circuitry allows for the reset provided by the Z8 (DS held Low while $\overline{AS}$ pulses) to be sufficient.
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

Figure 85 : Z-BUS Reset Timing.



AC CHARACTERISTICS

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	TsA(RD)	Address Setup to $\overline{RD}$ ↓	80		80		1
2	TsA(WR)	Address Setup to $\overline{WR}$ ↓	80		80		
3	ThA(RD)	Address Hold Time to $\overline{RD}$ ↑	0		0		1
4	ThA(WR)	Address Hold Time to $\overline{WR}$ ↑	0		0		
5	TsCEl(RD)	$\overline{CE}$ Low Setup Time to $\overline{RD}$	0		0		1
6	TsCEl(WR)	$\overline{CE}$ Low Setup Time to $\overline{WR}$	0		0		
7	ThCEl(RD)	$\overline{CE}$ Low Hold Time to $\overline{RD}$	0		0		1
8	ThCEl(WR)	$\overline{CE}$ Low Hold Time to $\overline{WR}$	0		0		
9	TsCEh(RD)	$\overline{CE}$ High Setup Time to $\overline{RD}$	100		70		1
10	TsCEh(WR)	$\overline{CE}$ High Setup Time to $\overline{WR}$	100		70		
11	TwRDl	$\overline{RD}$ Low Width	390		250		
12	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		0		
13	TdRDf(DR)	$\overline{RD}$ ↓ to Valid Data Delay		250		180	
14	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data not Valid Delay	0		0		
15	TdRD(DRz)	$\overline{RD}$ ↑ to Data Bus Float		70		45	2
16	TwWRl	$\overline{WR}$ Low Width	390		250		
17	TsDW(WR)	Data Setup Time to $\overline{WR}$	0		0		
19	Trc(WR)	Write Valid Access Recovery Time	1000		650		
20	Trc(RD)	Read Valid Access Recovery Time	1000 +WR_p		650 +WR_p		3

Notes : 1. Parameter does not apply to Interrupt Acknowledge Transactions.
2. Float delay is measured to the time the output has changed 0.5 V from steady state with minimum ac load and maximum dc load.
3. Recovery time equal to Trc(WR) + write pulse width of the opposite side.
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

Figure 86 : Non-Z-BUS CPU Interface Timing.

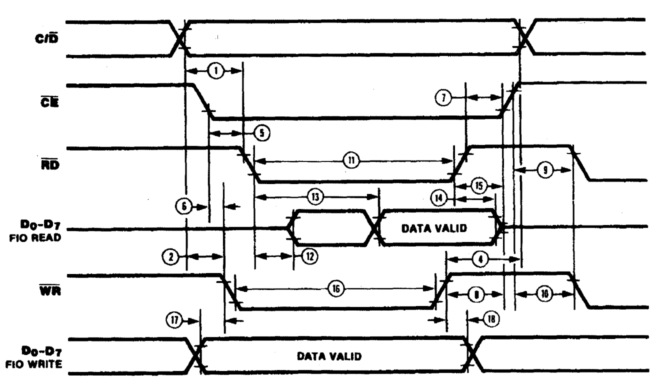
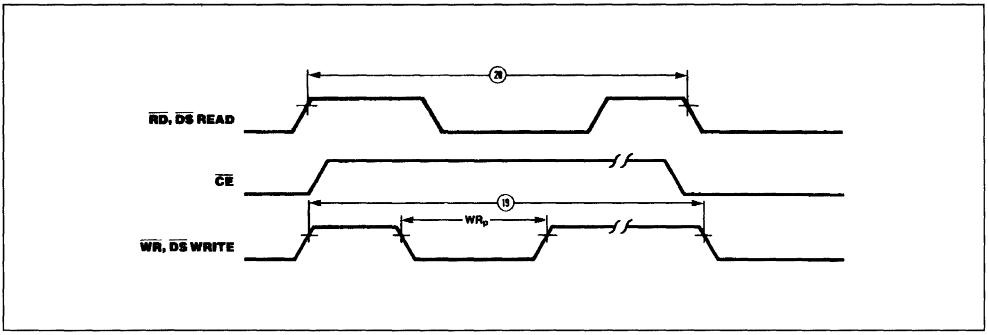


Figure 87 : Z-BUS/Non-Z-BUS Recovery Time.

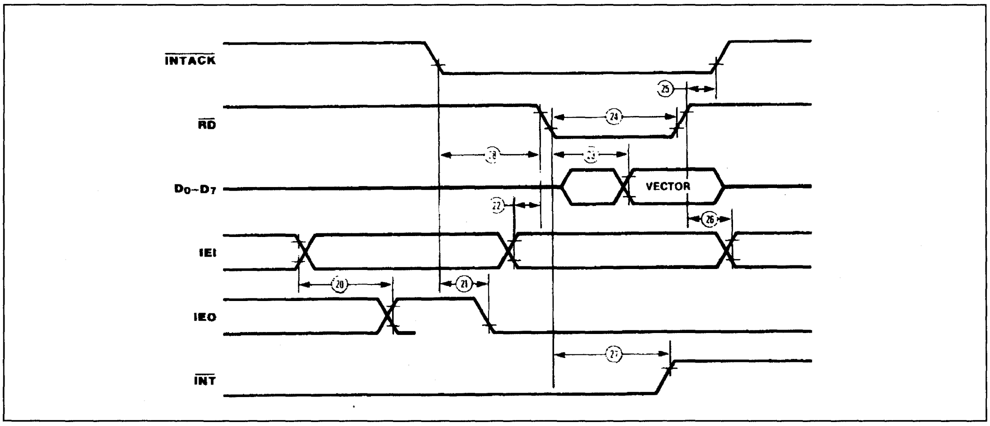


AC CHARACTERISTICS

N°	Symbol	Parameter	4 MHz Min.	4 MHz Max.	6 MHz Min.	6 MHz Max.	Notes*†
20	TdIEI(IEO)	IEI to IEO Delay		150		100	4
21	TdI(IEO)	$\overline{\text{INTACK}}$ ↓ to IEO ↓ Delay		350		250	4
22	TsIEI(RDA)	IEI Setup Time to $\overline{\text{RD}}$ (acknowledge)	100		70		4
23	TdRD(DR)	$\overline{\text{RD}}$ ↓ to Vector Valid Delay		250		180	
24	TwRDl(IA)	Read Low Width (interrupt acknowledge)	390		250		
25	ThIA(RD)	$\overline{\text{INTACK}}$ ↑ to $\overline{\text{RD}}$ ↑ Hold Time	30		20		
26	ThIEI(RD)	IEI Hold Time to $\overline{\text{RD}}$ ↑	20		10		
27	TdRD(INT)	$\overline{\text{RD}}$ ↓ to $\overline{\text{INT}}$ ↑ Delay		900		800	
28	TdDCST	Interrupt Daisy Chain Settle Time	350		250		4

Notes : 4. The parameter for the devices in any particular daisy chain must meet the following constraint : the delay from $\overline{\text{INTACK}}$ ↓ to $\overline{\text{RD}}$ ↓ must be greater than the sum of TdI(IEO) for the highest priority peripheral, TsIEI(RD) for the lowest, priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.
† Units in nanoseconds (ns).
* Timings are preliminary and subject to change.

Figure 88 : Non-Z-BUS Interrupt Acknowledge Timing.

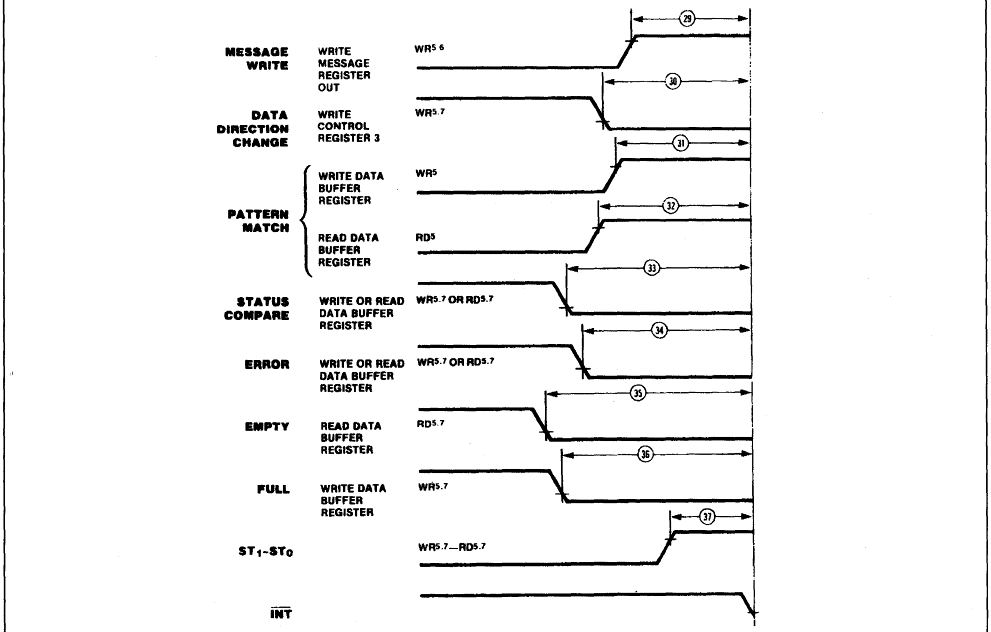


AC CHARACTERISTICS

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
29	TdMW(INT)	Message Write to $\overline{INT}$ Delay					5,6
30	TdDC(INT)	Data Direction Change to $\overline{INT}$ Delay					5,7
31	TdPMW(INT)	Pattern Match (write case) to $\overline{INT}$ Delay					5
32	TdPMR(INT)	Pattern Match (read case) to $\overline{INT}$ Delay					5
33	TdSC(INT)	Status Compare to $\overline{INT}$ Delay					5,7
34	TdER(INT)	Error to $\overline{INT}$ Delay					5,7
35	TdEM(INT)	Empty to $\overline{INT}$ Delay					5,7
36	TdFL(INT)	Full to $\overline{INT}$ Delay					5,7
37	TdSO(INT)	State 0 to $\overline{INT}$ Delay					

Notes : 5. Delay number is valid for State 0 only.
6. Write is from other side of FIO.
7. Write can be from either side, depending on programming of FIO.
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

Figure 89 : FIO Non-Z-BUS Interrupt Timing.

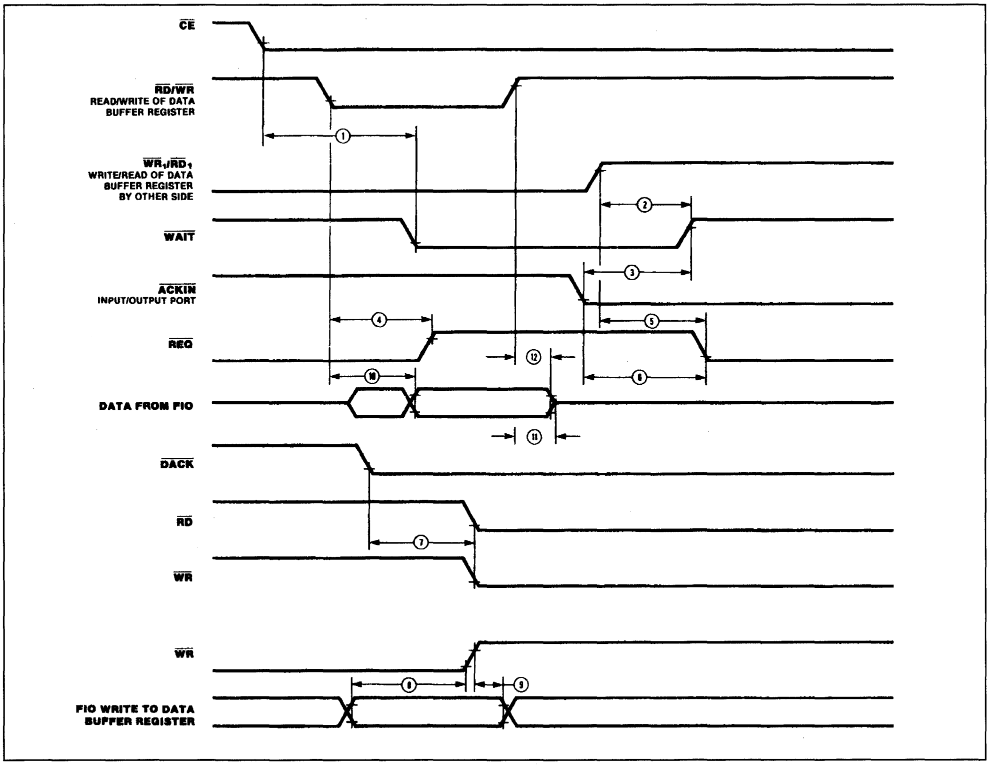

MESSAGE WRITE
WRITE MESSAGE REGISTER OUT
WR5 6
29
30
DATA DIRECTION CHANGE
WRITE CONTROL REGISTER 3
WR5.7
31
PATTERN MATCH
WRITE DATA BUFFER REGISTER
WR5
32
READ DATA BUFFER REGISTER
RD5
33
STATUS COMPARE
WRITE OR READ DATA BUFFER REGISTER
WR5.7 OR RD5.7
34
ERROR
WRITE OR READ DATA BUFFER REGISTER
WR5.7 OR RD5.7
35
EMPTY
READ DATA BUFFER REGISTER
RD5.7
36
FULL
WRITE DATA BUFFER REGISTER
WR5.7
37
ST1-ST0
WR5.7—RD5.7
INT

AC CHARACTERISTICS

N°	Symbol	Parameter	4 MHz Min.	4 MHz Max.	6 MHz Min.	6 MHz Max.	Notes*†
1	TdCE(WT)	$\overline{CE}$ ↓ to $\overline{WAIT}$ Active		200		170	
2	TdRDI(WT)	$\overline{RDI}$ ↑ or $\overline{WRI}$ ↑ to $\overline{WAIT}$ Inactive		1000		1000	
3	TdACK(WT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ Inactive		1000		1000	1
4	TdRD(REQ)	$\overline{RD}$ ↓ or $\overline{WR}$ ↓ to $\overline{REQ}$ Inactive		350		300	
5	TdRDI(REQ)	$\overline{RDI}$ ↑ or $\overline{WRI}$ ↑ to $\overline{REQ}$ Active		1000		1000	
6	TdACK(REQ)	$\overline{ACKIN}$ ↓ to $\overline{REQ}$ Active		1000		1000	
7	TdDAC(RD)	$\overline{DACK}$ ↓ to $\overline{RD}$ ↓ or $\overline{WR}$ ↓	100		80		
8	TSU(WR)	Data Setup Time to $\overline{WR}$	200				
9	Th(WR)	Data Hold Time to $\overline{WR}$	30			20	
10	TdDMA	$\overline{RD}$ ↓ to Valid Data		150		100	2
11	TdDMA(DRH)	$\overline{RD}$ ↑ to Data not Valid	0		0		2
12	TdDMA(DRZ)	$\overline{RD}$ ↑ to Data Bus Float		70		45	2

Notes : 1. The delay is from $\overline{DAV}$ ↓ for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Output Handshake.
2. Only when $\overline{DACK}$ is active.
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

Figure 90 : Non-Z-BUS Request/Wait Timing.

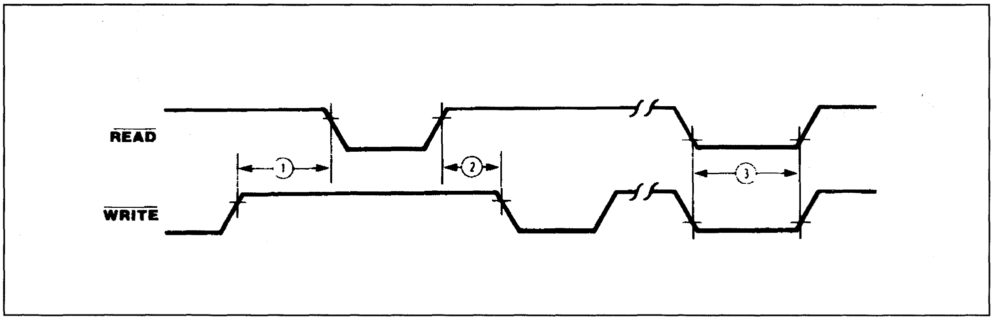

CE
RD/WR READ/WRITE OF DATA BUFFER REGISTER
WR₁/RD₁ WRITE/READ OF DATA BUFFER REGISTER BY OTHER SIDE
WAIT
ACKIN INPUT/OUTPUT PORT
REQ
DATA FROM FIO
DACK
RD
WR
WR
FIO WRITE TO DATA BUFFER REGISTER
1
2
3
4
5
6
7
8
9
10
11
12

AC CHARACTERISTICS

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	TdWR(RD)	Delay from $\overline{WR}$ ↑ to $\overline{RD}$ ↓	100		70		
2	TdRD(WR)	Delay from $\overline{RD}$ ↑ to $\overline{WR}$ ↓	100		70		
3	TwRD + WR	Width of $\overline{RD}$ and $\overline{WR}$, both Low for Reset	500		350		

Notes : * Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

Figure 91 : Non-Z-BUS Reset Timing.

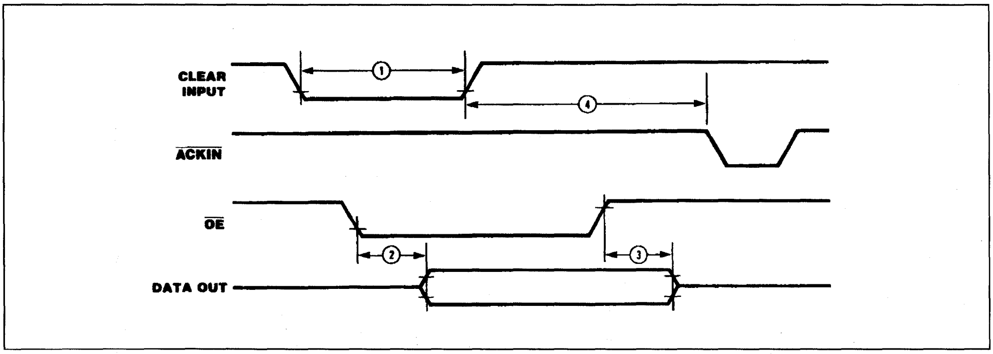


AC CHARACTERISTICS

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	TwCLR	Width of Clear to Reset FIFO	700		700		
2	TdOE(DO)	$\overline{OE}$ ↓ to Data Bus Driven		210		210	
3	TdOE(DRZ)	$\overline{OE}$ ↑ to Data Bus Float		150		150	
4	TdCLR(ACK)	$\overline{CLEAR}$ ↑ to $\overline{ACKIN}$ ↓	800		800		

Notes : * Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

Figure 92 : Port 2 Side Operation.



SGS-THOMSON MICROELECTRONICS

AC CHARACTERISTICS

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	TsDI(ACK)	Data Input to $\overline{\text{ACKIN}}$ ↓ to Setup Time	50		50		
2	TdACKf(RFD)	$\overline{\text{ACKIN}}$ ↓ to RFD ↓ Delay	0	500	0	500	
3	TdRFDr(ACK)	RFD ↑ to $\overline{\text{ACKIN}}$ ↓ Delay	0		0		
4	TsDO(DAV)	Data Out to $\overline{\text{DAV}}$ ↓ Setup Time	50		25		
5	TdDAVf(ACK)	$\overline{\text{DAV}}$ ↓ to $\overline{\text{ACKIN}}$ ↓ Delay	0		0		
6	ThDO(ACK)	Data Out to $\overline{\text{ACKIN}}$ Hold Time	50		50		
7	TdACK(DAV)	$\overline{\text{ACKIN}}$ ↓ to $\overline{\text{DAV}}$ ↑ Delay	0	500	0	500	
8	ThDI(RFD)	Data Input to RFD ↓ Hold Time	0		0		
9	TdRFDf(ACK)	RFD ↓ to $\overline{\text{ACKIN}}$ ↑ Delay	0		0		
10	TdACKr(RFD)	$\overline{\text{ACKIN}}$ ↑ ($\overline{\text{DAV}}$ ↑) to RFD ↑ Delay-interlocked and 3-wire Handshake	0	400	0	400	
11	TdDAVr(ACK)	$\overline{\text{DAV}}$ ↑ to $\overline{\text{ACKIN}}$ ↑ (RFD ↑)	0		0		
12	TdACKr(DAV)	$\overline{\text{ACKIN}}$ ↑ to $\overline{\text{DAV}}$ ↓	0	800	0	800	
13	TdACKf(empty)	$\overline{\text{ACKIN}}$ ↓ to Empty	0		0		
14	TdACKf(full)	$\overline{\text{ACKIN}}$ ↓ to Full	0		0		
15	TcACK	$\overline{\text{ACKIN}}$ Cycle Time	1		1		1

Notes : * Timings are preliminary and subject to change.
† Units in nanoseconds (ns), except as noted.
1. Units in microseconds.

Figure 93 : 2-Wire Handshake (port 2 side only) Output.

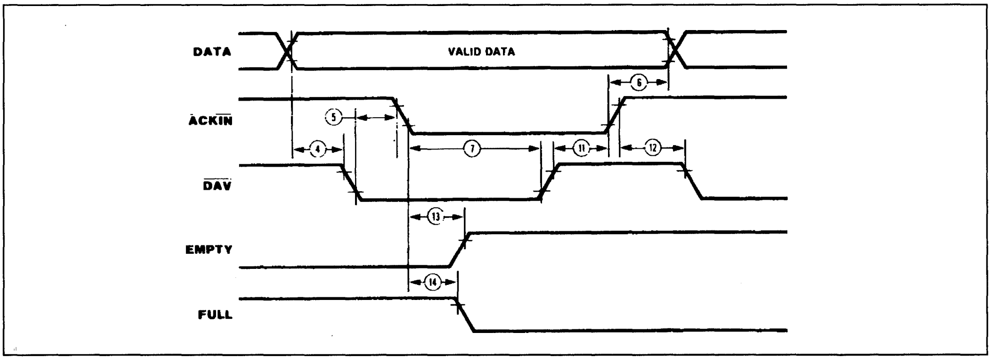
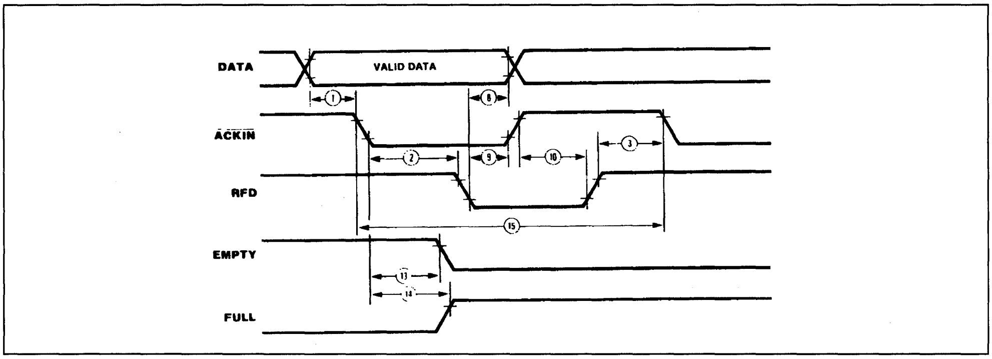


Figure 94 : 2-Wire Handshake (port 2 side only) Input.



AC CHARACTERISTICS (Continued)

N°.	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	TsDI(DAV)	Data Input to $\overline{DAV}$ ↓ Setup Time	50		50		
2	TdDAVIf(RFD)	$\overline{DAV}$ ↓ to RFD ↓ Delay	0	500	0	500	
3	TdDAVf(DAC)	$\overline{DAV}$ ↓ to $\overline{DAC}$ ↑ Delay	0	500	0	500	
4	ThDI(DAC)	Data in to DAC ↑ Hold Time	0		0		
5	TdDACIr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay	0		0		
6	TdDAVIr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay	0	500	0	500	
7	TdDAVIr(RFD)	$\overline{DAV}$ ↑ to RFD ↑ Delay	0	500	0	500	
8	TdRFDI(DAV)	RFD ↑ to $\overline{DAV}$ ↓ Delay	0		0		
9	TsDO(DAC)	Data Out to $\overline{DAV}$ ↓					
10	TdDAVOf(RFD)	$\overline{DAV}$ ↓ to RFD ↓ Delay	0		0		
11	TdDAVOf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay	0		0		
12	ThDO(DAC)	Data Out to DAC ↑ Hold Time					
13	TdDACOr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay		400		400	
14	TdDAVOr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay	0		0		
15	TdDAVOr(RFD)	$\overline{DAV}$ ↑ to RFD ↑ Delay	0		0		
16	TdRFDO(DAV)	RFD ↑ to $\overline{DAV}$ ↓ Delay	0	800	0	800	

Notes : * Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

Figure 95 : 3-Wire Handshake Input.

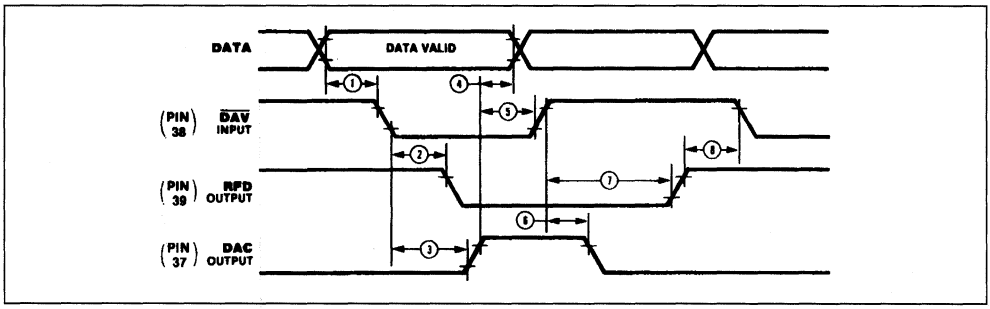
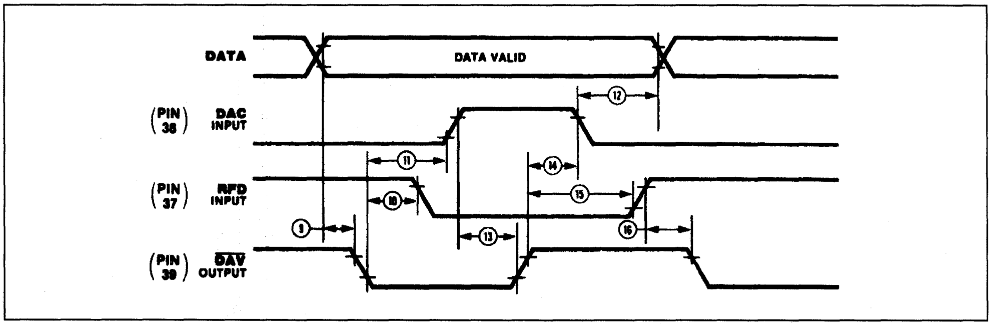


Figure 96 : 3-Wire Handshake Output.



ORDERING INFORMATION

Type	Package	Temp. Range	Clock
Z8038B1V	PDIP-40	0 to + 70 °C	4 MHz
Z8038B6V	PDIP-40	– 40 to + 85 °C	4 MHz
Z8038C1V	PLCC44	0 to + 70 °C	4 MHz
Z8038C6V	PLCC44	– 40 to + 85 °C	4 MHz
Z8038D1N	CDIP-40	0 to + 70 °C	4 MHz
Z8038D6N	CDIP-40	– 40 to + 85 °C	4 MHz
Z8038D2N	CDIP-40	– 55 to + 125 °C	4 MHz
Z8038AB1V	PDIP-40	0 to + 70 °C	6 MHz
Z8038AB6V	PDIP-40	– 40 to + 85 °C	6 MHz
Z8038AC1V	PLCC44	0 to + 70 °C	6 MHz
Z8038AC6V	PLCC44	– 40 to + 85 °C	6 MHz
Z8038AD1N	CDIP-40	0 to + 70 °C	6 MHz
Z8038AD6N	CDIP-40	– 40 to + 85 °C	6 MHz
Z8038AD2N	CDIP-40	– 55 to + 125 °C	6 MHz

Note : PDIP = Plastic DIP ; CDIP = Ceramic Multilayer DIP ; PLCC = Plastic Leaded Chip Carrier.

SERIAL COMMUNICATIONS CONTROLLER

Thank you for your interest in the SCC, one of the most versatile and most popular Serial Data Communications ICs. This document is intended to provide answers to all technical questions about the Z8530 Serial Communications Controller. Please ready this Preface where we try to anticipate your questions.

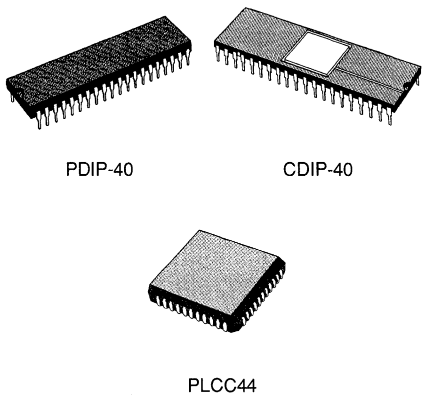
- If you are new to serial data communications, you will need additional tutorial information. Of the many introductory texts on this subject, *Technical Aspects of Data Communications* by John E. McNamara, published by Digital Press (DEC) 1982, is one of the best.
- If you have designed with simpler UARTs and USARTs, and HDLC/SDLC devices, the SCC offers you far greater flexibility, but also requires an in-depth study and understanding of the impact and the use of its many powerful features. This manual contains important information.
- If you are familiar with the Z80-SIO, you will feel right at home with the SCC, for it is really a functionally enhanced superset of the Z80-SIO.

Most users read only chapters that are of interest to them. If you are designing the microcomputer hardware structure using the SCC as a peripheral, you will want to read the Initialisation Worksheet and Interrupt Routine Sections.

If you are programming a system using the SCC, you will be more interested, on the Initialization Worksheet Section.

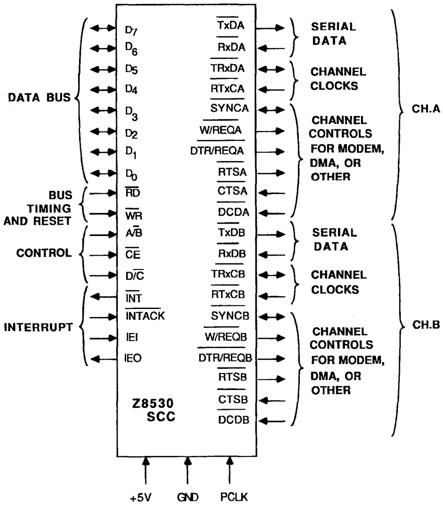
Points To Watch Out For :

1. Follow the worksheet for initialization (page). Unexplainable operations may occur if this procedure is not followed.
2. Watch out for Write Recovery time violation (Interfacing Section). Both the CPU clock rate and the SCC clock rate will affect the Write Recovery time.
3. Ensure Mode bits are not changed when writing Commands. (Register Overview page 75). Each Mode bit affects only one function and a Command bit entry requires a rewrite of the entire register ; therefore, care must be taken to insure the integrity of the Mode bits whenever a new command is issued.
4. Data must be valid prior to falling edge of $\overline{WR}$ or $\overline{DS}$.
5. If not used, $\overline{INTACK}$ should be tied high.



(Ordering Information at the end of the datasheet)

Figure 1 : Logic Functions.



CAPABILITIES

- Two independent full-duplex channels.
- Synchronous/Isosynchronous data rates :
  - Up to 1/4 of the PCLK (i.e., 1 Mbit/sec. maximum data rate with 4 MHz PCLK. Using external phase-lock loop.
  - Up to 375 Kbit/sec. with a 6 MHz clock rate. Up to 250 Kbit/sec. with a 4 MHz clock rate (FM encoding using digital phase-locked loop).
  - Up to 187.5 Kbit/sec. with a 6 MHz clock rate Up to 125 Kbit/sec. with a 4 MHz clock rate (NRZI encoding using digital phase-locked loop).
- Asynchronous capabilities :
  - 5, 6, 7, or 8 bits per character
  - 1, 1-1/2, or 2 stop bits
  - Odd or even parity
  - Times 1, 16, 32, or 64 clock modes
  - Break generation and detection
  - Parity, overrun and framing error detection.
- Byte-oriented synchronous capabilities :
  - Internal or external character synchronization
  - 1 or 2 sync characters (6 or 8 bits/character) in separate registers
  - Automatic Cyclic redundancy check (CRC) generation/detection.
- SDLC/HDLC capabilities :
  - Abort sequence generation and checking
  - Automatic zero insertion and deletion
  - Automatic flag insertion between messages
  - Address field recognition
  - I-field residue handling
  - CRC generation/detection
  - SDLC loop mode with EOP recognition/loop entry and exit.
- Receiver data registers quadruply buffered. Transmitter data registered double buffered.
- NRZ, NRZI, or FM encoding/decoding.
- Baud-rate generator in each channel.
- Digital phase-locked loop for clock recovery.
- Crystal oscillator.

GENERAL DESCRIPTION

The SCC Serial Communications Controller is a dual-channel, multiprotocol data communications peripheral designed for use with 8-bit and 16-bit microprocessors. The SCC functions as a serial-to-parallel, parallel-to-serial converter/controller. The SCC can be software-configured to satisfy a wide variety of serial communications applications. The device contains a variety of new, sophisticated internal functions including on-chip baud rate generators, digital phase-lock loops, and crystal oscillators, which dramatically reduce the need for external logic.

The SCC handles asynchronous formats, Synchronous byte-oriented protocols such as IBM Bisync, and Synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (telecommunications, cassette, diskette, tape drivers, etc.).

The device can generate and check CRC codes in any Synchronous mode and can be programmed to check data integrity in various modes. The SCC also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

With access to 14 Write registers and 7 Read registers per channel, the user can configure the SCC so that it can handle all asynchronous formats regardless of data size, number of stop bits, or parity requirements. The SCC accommodates all synchronous formats including character, byte, and bit-oriented protocols.

Within each operating mode, the SCC also allows for protocol variations by checking odd or even parity bits, character insertion or deletion, CRC generation and checking/break and abort generation and detection, and many other protocol-dependent features.

The SCC Z8530 is designed for non-multiplexed buses and is easily interfaced to CPUs such as the 8080, Z80, 6800, 68000 and *Multibus.

GENERAL DESCRIPTION (cont'd)

Figure 2 and Figure 5 show block diagrams of the SCC. Received data enters the receive data pins and follows one of several data paths, depending on the state of the control logic. The contents of the registers and the state of the external control pins establish the internal control logic. Transmitted data follows a similar pattern of control, register, and external pin definition.

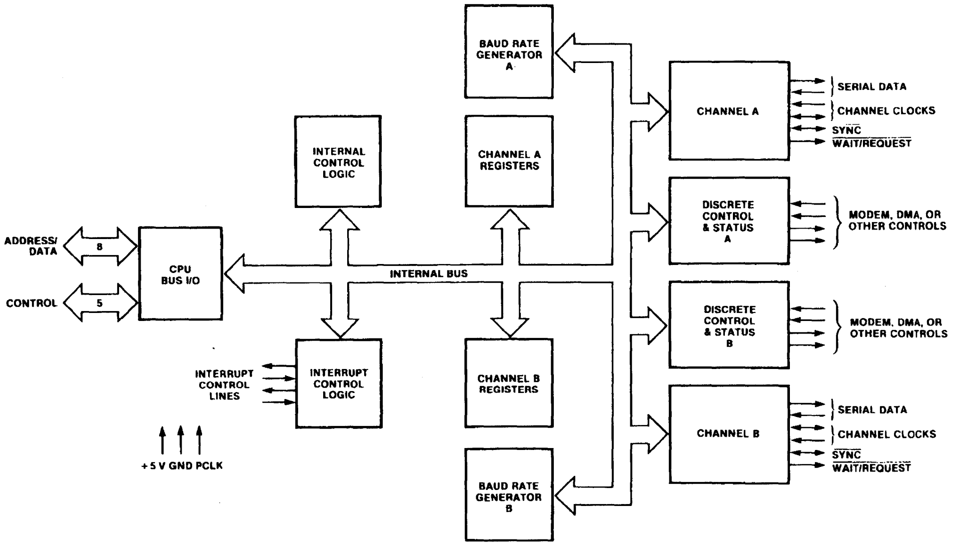
PIN DESCRIPTIONS

The SCC pins are divided into seven functional groups : Address/Data, Bus Timing and Reset, Device Control, Interrupt, Serial Data (both channels), Peripheral Control (both channels), and Clocks (both channels). Figures 3 and 4 show the Pin Configuration in both the proposed packages, Dual in Line and Chip Carrier.

The Address/Data group consists of the bidirectional lines used to transfer data between the CPU and the SCC. The direction of these lines depends on whether the SCC is selected and whether the operation is a Read or a Write.

The Timing and Control groups designate the type of transaction to occur and when this transaction will occur. The Interrupt group provides inputs and outputs to conform to the bus specifications for handling and prioritizing interrupts. The remaining groups are divided into Channel A and Channel B groups for serial data (transmit or receive), periphe-

Figure 2 : SCC Block Diagram.



PIN DESCRIPTION (cont'd)

ral control (such as DMA or modem), and the input and output lines for the receive and transmit clocks.

Here below are described the pin functions of the Z8530 Serial Communications Controller.

A/B. Channel A/Channel B Select (input, Channel A active HIGH). This signal selects the channel in which the Read or Write operation occurs.

CE. Chip Enable (input, active LOW). This signal selects the SCC for operation. It must remain active throughout the bus transaction.

D0-D7. Data Lines (bidirectional, 3-state). These I/O lines carry data or control information to and from the SCC.

D/C. Data/Control (input, Data active HIGH). This signal defines the type of information transfer performed by the SCC : data or control.

RD. Read (input, Active LOW). This signal indicates a Read operation and when the SCC is selected, enables the SCC bus drivers. During the interrupt acknowledge cycle, this signal gates the interrupt vector onto the bus if the SCC is the highest priority device requesting an interrupt.

WR. Write (input, active LOW). When the SCC is selected, this signal indicates a Write operation. The coincidence of RD and WR is interpreted as a Reset.

CTSA, CTSB. Clear to Send (inputs, active LOW). If these pins are programmed as auto enables, a LOW on these inputs enables the respective transmitters. If not programmed as auto enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The SCC detects transitions on these inputs and can interrupt the CPU on either logic level transitions.

DCDA, DCDB. Data Carrier Detect (inputs, active LOW). These pins function as receiver enables if they are programmed as auto enable bits ; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accomodate slow rise-time signals. The SCC detects transitions on these pins and can interrupt the CPU on either logic level transitions.

DTR/ REQA, DTR/ REQB. Data Carrier Detect (inputs, active LOW). These pins function as receiver enables if they are programmed into the DTR bit. They can also be used as general-purpose outputs (transmit) or as request lines for the DMA controller. The SCC allows full duplex DMA transfers.

IEI. Interrupt Enable In (input, active HIGH). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A HIGH on IEI indicates that no other higher priority

Figure 3 : DIP Pin Connections.

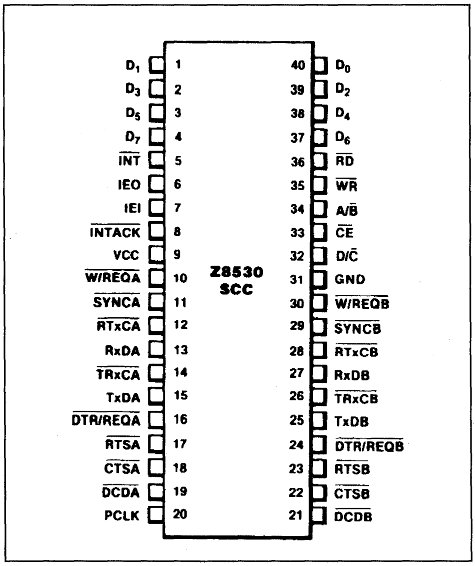
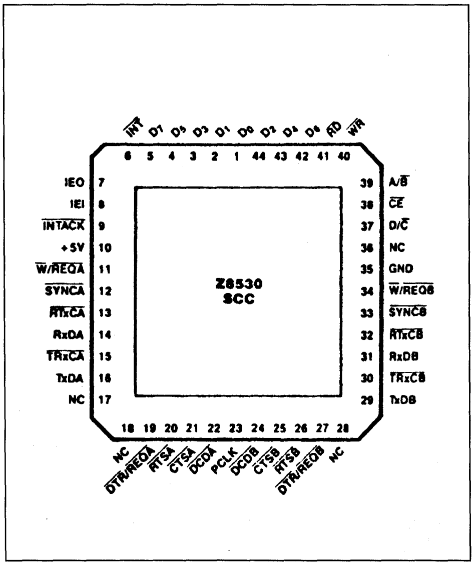


Figure 4 : Chip Carrier Pin Connection.



PIN DESCRIPTION (cont'd)

device has an Interrupt Under Service (IUS) or is requesting an interrupt.

IEO. Interrupt Enable Out (output, active HIGH). IEO is HIGH only if IEI is HIGH and the CPU is not servicing an SCC or SCC interrupt, or the controller is not requesting an interrupt (interrupt acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{\textbf{INTACK}}$. Interrupt Acknowledge (input, active LOW). This signal indicates an active interrupt acknowledge cycle. During this cycle, the interrupt daisy chain settles. When $\overline{\text{RD}}$ or $\overline{\text{DS}}$ becomes active, the SCC places an interrupt vector on the data bus (if IEI is HIGH). $\overline{\text{INTACK}}$ is latched by the rising edge of $\overline{\text{AS}}$ or PCLK.

$\overline{\textbf{INT}}$. Interrupt Request (output, open-drain, active LOW). This signal is activated when the SCC is requesting an interrupt.

PCLK. Clock (input). This is the master clock used to synchronize internal signals. PCLK is not required to have any phase relationship with the master system clock. PCLK is a TTL level signal.

$\overline{\textbf{RTSA}}$, $\overline{\textbf{RTSB}}$. Request to Send (outputs, active LOW). When the Request to Send (RTS) bit in Write Register 5 (Figure 48) is set, the $\overline{\text{RTS}}$ signal goes LOW. When the RTS bit is reset in the Asynchronous mode and auto enables is on, the signal goes HIGH after the transmitter is empty. In Synchronous mode or in Asynchronous mode with auto enables off, the $\overline{\text{RTS}}$ pins strictly follow the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{\textbf{RTxCA}}$, $\overline{\textbf{RTxCB}}$. Receive/ Transmit Clocks (inputs, active LOW). The functions of these pins are under program control. In each channel, $\overline{\text{RTxC}}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the digital phase-locked loop (refer to Section 4 for bit configurations). This pins can also be programmed for use the respective $\overline{\text{SYNC}}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in asynchronous modes.

RxDA, RxDB. Receive Data (inputs, active HIGH). These input signals receive serial data at standard TTL levels.

$\overline{\textbf{SYNCA}}$, $\overline{\textbf{SYNCB}}$. Synchronization (inputs/outputs, active LOW). These pins can act as either inputs, outputs, or as part of the crystal oscillator circuit. In the Asynchronous Receive mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. In this mode, transitions on these lines affect the state of the Sync/ Hunt status bits in Read Register 0 (Figure 59), but have no other function.

In External Synchronization mode with the crystal oscillator not selected, these lines also act as inputs. In this mode, $\overline{\text{SYNC}}$ must be driven LOW two receive clock cycles after the last bit in the sync character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of $\overline{\text{SYNC}}$.

In the Internal Synchronization mode, (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the receive clock cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync character is recognized (regardless of character boundaries). In SDLC mode, these pins act as outputs and are valid on receipt of a flag.

$\overline{\textbf{TRxCA}}$, $\overline{\textbf{TRxCB}}$. Transmit/ Receive Clocks (inputs or outputs, active LOW). The functions of these pins are under program control. $\overline{\text{TRxC}}$ may supply the receive clock or the transmit clock in the Input mode or supply the output of the digital phase-locked loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode. (Refer to Section 4 for bit configuration).

TxDA, TxDB. Transmit Data (outputs, active HIGH). This output signal transmits serial data at standard TTL levels.

$\textbf{W}/\overline{\textbf{REQA}}$, $\overline{\textbf{W}}/\overline{\textbf{REQB}}$. Wait/ Request (outputs, open drain when programmed for Wait function, driven HIGH or LOW when programmed for a Request function). These dual-purpose outputs can be programmed as Request (receive) lines for a DMA controller or as Wait lines to synchronize the CPU to the SCC data rate. The reset state is Wait. The SCC allows full duplex DMA transfer.

OVERVIEW

The SCC internal structure provides all the interrupt and control logic necessary to interface with non-multiplexed bus. Interface logic is also provided to monitor modem or peripheral control inputs and outputs. All of the control signals are general purpose and can be applied to various peripheral devices as well as used for modem control.

The center for data activity revolves around the internal read and write registers. The programming of these registers provides the SCC with functional "personality" ; i.e., register values can be assigned before or during program sequencing to determine how the SCC will establish a given communication protocol.

Register Functions

All modes of communication are established by the bit values of the write registers. As data is received or transmitted, read register values may change. These changed values can promote software action or internal hardware action for further register changes.

The register set for each channel includes 14 write registers and seven read registers. Ten write registers are used for control, two for sync character generation, and two for baud rate generation. In addition there are two write registers which are shared by both channels ; one is the interrupt vector register and one is the master interrupt control and reset register. Four read registers indicate status information, two are for baud rate generation, and one for the receive buffer. In addition there are two read registers which are shared by both channels ; one for the interrupt pending bits and one for interrupt vector.

Table 1 lists the assigned functions for each read and write register. The SCC contains only one WR2 (interrupt vector) and one WR9 (master interrupt control). Both registers are accessed and shared by either channel. Chapter 7 provides a detailed bit legend and description of each register.

Table 1 : Register Set.

	Read Register Functions
RR0	Transmitt/ Receive buffer status, and External status
RR1	Special Receive Condition status, residue codes, error conditions
RR2	Modified (Channel B only) interrupt vector and Unmodified interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive buffer
RR10	Miscellaneous XMTR, RCVR status parameters
RR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant
RR15	External / Status interrupt control information
	Write Register Functions
WR0	Command Register, CRC initialization resets for various modes
WR1	Interrupt conditions, Wait / DMA request control
WR2	Interrupt vector (access through either channel)
WR3	Receive / Control parameters, number of bits per character, RxCRC enable
WR4	Transmit / Receive miscellaneous parameters and codes, clock rate, number of sync characters, stop bits, parity
WR5	Transmit parameters and control, number of Tx bits per character, TxCRC enable
WR6	Sync character (1 st byte) or SDLC flag
WR8	Transmitt buffer
WR9	Master interrupt control and reset (accessed through either channel), reset bits, control interrupt daisy chain
WR10	Miscellaneous transmitter/receiver control bits, NRZI, NRZ, FM encoding, CRC reset
WR11	Clock mode control, source of Rx and Tx clocks
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits : baud rate generator, Phase-Locked Loop control, auto echo, local loopback
WR14	External/ Status interrupt control information-control external conditions causing interrupts

OVERVIEW (cont'd)

Data Paths

Figure 6 illustrates the data paths involved in the six major areas of the SCC :

- Transmitter
- Receiver
- Baud rate generator
- DPLL
- Clocking options
- Data encoding

All communication modes are established by programming the write registers. As data is received or transmitted, read register values may change, altering the direction of the data path. These changed values can promote software action or internal hardware action for further register changes.

Transmitter. The transmitter has an 8-bit Transmit Data register (WR8) loaded from the internal data bus and a Transmit Shift register loaded from either WR6, WR7, or the Transmit Data register. In byte-oriented modes, WR6 and WR7 can be programmed with sync characters. In Monosync mode, an 8-bit or 6-bit sync character is used (WR6), whereas a 16-bit sync character is used (WR6 and WR7) in Bisync mode. In bit-oriented synchronous modes, the flag contained in WR7 is loaded into the Transmit Shift register at the beginning and end of a message.

If asynchronous data is processed, WR6 and WR7 are not used and the Transmit Shift register is formatted with start and stop bits shifted out to the transmit multiplexer at the selected clock rate. Synchronous data (except SDLC/HDLC) is shifted to the CRC generator as well as to the transmit multiplexer.

SDLC/HDLC data is shifted to the CRC Generator and out through the zero insertion logic (which is disabled while the flags are being sent). A "0" is inserted in all address, control, information, and frame check fields following five contiguous "1s" in the data stream. The result of the CRC generator for SDLC data is also routed through the zero insertion logic and then to the transmit multiplexer.

Receiver. The receiver has a three deep 8-bit Data FIFO (paired with an 8-bit Error FIFO), and an 8-bit shift register. This arrangement creates a 3-byte delay time, which allows the CPU time to service an interrupt at the beginning of a block of high-speed data. With each Receive Data FIFO, the error FIFO stores parity and framing errors and other types of status information. The error FIFO is readable in Read Register 1.

Incoming data is routed through one of several paths depending on the mode and character length. In Asynchronous mode, serial data enters the 3-bit delay (Figure 5) if the character length of seven or eight bits is selected. If a character length of five or six bits is selected, data enters the receive shift register directly.

In synchronous modes, the data path is determined by the phase of the receive process currently in operation. A synchronous receive operation begins with a hunt phase in which a bit pattern that matches the programmed sync characters (6-bit, 8-bit, or 16-bit is searched).

The incoming data then passes through the Sync register and is compared to a sync character stored in WR6 or WR7 (depending on which mode it is in). The monosync mode matches the sync character programmed in WR7 and the character assembled in the Receive Sync register to establish synchronization.

Synchronization is achieved differently in the Bisync mode. Incoming data is shifted to the Receive Shift register while the next eight bits of the message are assembled in the Receive Sync register. If these two characters match the programmed characters in WR6 and WR7, synchronization is established. Incoming data can then bypass the Receive Sync register and enter the 3-bit delay directly.

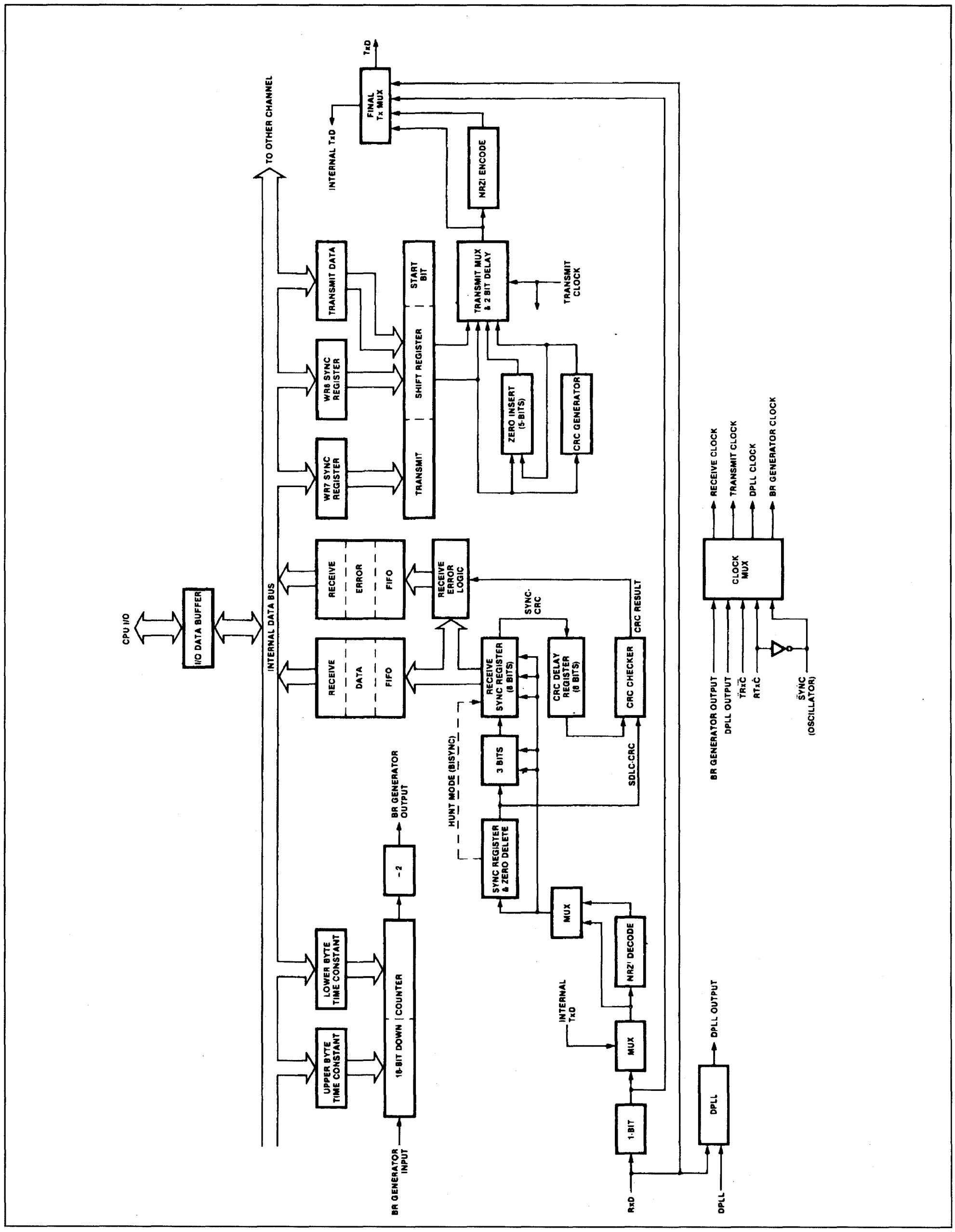
The SDLC mode of operation uses the Receive Sync register to monitor the receive data stream and to perform zero deletion when necessary ; i.e., when five continuous "1s" are received, the sixth bit is inspected and deleted from the data stream if it is "0". The seventh bit is inspected only if the sixth bit equals one. If the seventh bit is "0", a flag sequence has been received and the receiver is synchronized to that flag. If the seventh bit is a "1", an abort or an EOP (End Off Poll) is recognized, depending on the selection of either the normal SDLC mode or SDLC Loop mode.

The same path is taken by incoming data for both SDLC modes. The reformatted data enters the 3-bit delay and is transferred to the Receive Shift register. The SDLC receive operation begins in the hunt phase by attempting to match the assembled character in the Receive Shift Register with the flag pattern in WR7. Then the flag character is recognized, subsequent data is routed through the same path, regardless of character length.

Either the CRC - 16 or CRC - SDLC cyclic redundancy check (CRC) polynomial can be used for both Monosync and Bisync modes, but only the CRC - SDLC polynomial is used for SDLC operation. The data path taken for each mode is also different.

OVERVIEW (cont'd)

Figure 5 : Data Paths.



OVERVIEW (cont'd)

Bisync protocol is a byte-oriented operation that requires the CPU to decide whether or not a data character is to be included in CRC calculation. An 8-bit delay in all synchronous modes except SDLC is allowed for this process. In SDLC mode, all bytes are included in the CRC calculation.

Baud Rate Generator. Each channel in the SCC contains a programmable baud rate gene-rator. Each generator consists of two 8-bit, time-constant registers forming a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output that makes the output a square wave. On start-up, the flip-flop on the output is set High so that it starts in a known state, the value in the time-constant register is again loaded into the counter, and the counter begins counting down. When a count of zero is reached, the output of the baud rate generator toggles, the value in the time-constant register is loaded into the counter, and the process starts over. The time constant can be changed at any time, but the new value does not take effect until the next load of the counter.

No attempt is made to synchronize the loading of a new time constant with the clock used to drive the generator. When the time constant is to be changed, the generator should be stopped by writing to an enable bit in WR14. This ensures the loading of the correct time constant.

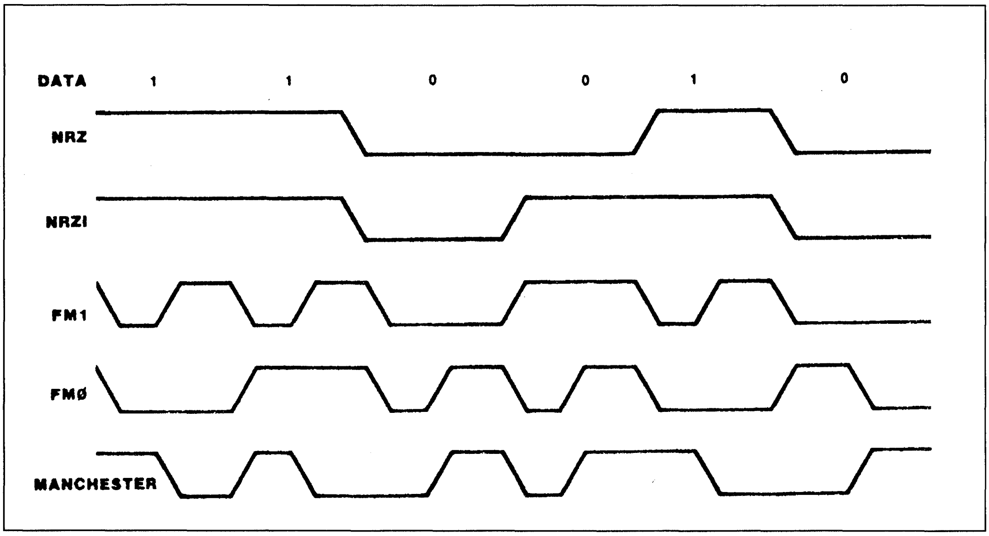
If neither the transmit clock nor the receive clock are programmed to come from the $\overline{TRXC}$ pin, the output of the baud rate generator may be made available for external use on the $\overline{TRXC}$ pin.

Digital Phase-locked Loop (DPLL). The SCC contains a digital phase-locked loop that can be used to recover clock information from a data stream with NRZI or FM coding. The DPLL is driven by a clock nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a receive clock for the data. This clock can then be used as the SCC receive clock, the transmit clock, or both.

Clocking Options. The SCC can select several clock sources for internal and external use. Write Register 11 is the Clock Mode Control register for both the receive and transmit clocks. It determines the type of signal on the $\overline{SYNC}$ and $\overline{RTxC}$ pins and the direction of the $\overline{TRxC}$ pin.

Write Register 11 also controls the output of the baud rate generator, the DPLL output, and the selection of either a TT1 or XTAL output for the $\overline{RTxC}$ pin.

Figure 6 : Data Encoding Methods.



OVERVIEW (cont'd)

Data Encoding. Figure 6 illustrates the four encoding methods used by the SCC. In NRZ encoding, a "1" is represented by a High level and a "0" is represented by a Low level. In NRZI encoding, a "1" is represented by no change in level and a "0" is represented by a change in level. In FM1 (more properly, biphase mark), a transition occurs at the beginning of every bit cell. A "1" is represented by an additional transition at the center of the bit cell and a "0" is represented by the absence of a tran-sition at the center of the bit cell. In FM0 (more properly, biphase space), a transition occurs at the beginning of every bit cell. A "0" is represented by an additional transition at the center of the bit cell and a "1" is represented by the absence of a transition at the center of the bit cell.

In addition to these four methods, the SCC can be used to decode Manchester (biphase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is Low to High, the bit is "0". If the transition is High to Low, the bit is "1".

Data Communications Capabilities

SCC logic handles all asynchronous, byte-oriented synchronous, and bit-oriented synchronous modes of operation. The following section briefly describes asynchronous, synchronous, and SDLC modes of communication.

Asynchronous. Figure 7 represents a typical asynchronous message format using one start bit, seven data bits, one parity bit, and one stop bit. A start bit is a High-to-Low transition detected by an asynchronous receiver and is actually an information bit notifying the receiver of an incoming message.

The start bit also initiates a clock circuit to provide latching pulses during expected data bit intervals. The parity bit is provided for error checking. The parity bit is calculated in both the receiver and the transmitter ; the two results are compared to ensure that the expected and the actual bit values match. The stop bit returns the message unit to the quiescent marking state ; i.e., a constant high state condition lasts until the next High-to-Low start bit indicates an incoming data byte. During reception, the start and stop bits are stripped away and checked for errors, leaving only the working data for CPU interaction. The number of selected bits for each asynchronous function may differ between the transmitter and the receiver.

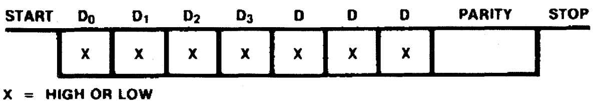
Monosync Mode. Monosync and Bisync modes require clocking information to be transmitted along with the data either by a method of encoding data that contains clocking information, or by a modem that encodes or decodes clock information in the modulation process.

Start and stop bits are not required in synchronous modes. All bits are used to transmit data. This eliminates the "waste" characteristic of asynchronous communication.

Figure 8 shows the character format for synchronous transmission. For example, bits 1-8 might be one character and bits 9-13 part of another character ; or bit 1 might be part of one character, bits 2-9 part of a second character, and bits 10-13 part of a third character. The framing (where each character begins) of each character is accomplished by defining a synchronization character, commonly called a "sync character".

The CPU places the receiver in Hunt mode whenever transmission begins (or whenever a data dropout has occurred and the hardware determines that resynchronization is necessary). In Hunt mode, the receiver shifts a bit into the Receive Shift register and compares the contents of the Receive Shift register and with the sync character (stored in another register), repeating the process until a match occurs. When a match occurs, the receiver begins transferring bytes to the receive FIFO.

Figure 7 : Asynchronous Message Format.



OVERVIEW (cont'd)

Bisynchronous Mode. The Bisync mode of operation (Figure 9) is similar to the Monosync mode, except that two sync characters are provided instead of one. Bisync attemps a more structured approach to synchronization through the use of special characters as message "headers" or "trailers". A detailed description of IBM's Bisync can be found in McNamara's Book (See Preface).

External Sync Mode. External Sync mode (Figure 10) eliminates the use of sync characters in the serial data stream by providing an external sync signal to mark the beginning of a data field ; i.e., an external input pin (Sync) waits for an active state change to indicate the beginning of an information field.

SDLC Mode. Synchronous Data Link Control mode (SDLC) uses synchronization characters similar to Bisync and Monosync modes (such as flags and pad characters), but it is a bit-oriented protocol instead of byte-oriented protocol.

Any data communication link involves at least two stations. The station that is responsible for the data link and issues the commands to control the link is called the "primary station". The other station is a "secondary station". Not all information transfers need to be initiated by a primary station. In SDLC mode, a secondary station can be the initiator.

The basic format for SDLC is a "frame" (Figure 11).The information field is not restricted in format or content and can be of any reasonable length (including zero). Its maximum length is that which can be expected to arrive at the receiver error-free most of time. Hence, the determination of maximum length is a function of communication channel error rate.

The two flags that delineate the SDLC frame serve as reference points when positioning the address and control fields, and they initiate the transmission error check. The ending flag indicates to the receiving station that the 16 bits just received constitute the frame check. The ending flag could be followed by another frame, another flag, or an idle. This means that when two frames fallow one another, the intervening flag may simultaneously be the ending flag of the first frame and the begin-ning flag of the next frame. Since the SDLC mode does not use characters of defined length, but rather works on a bit-by-bit basis, the 01111110 (7EH) flag can be recognized at any time.

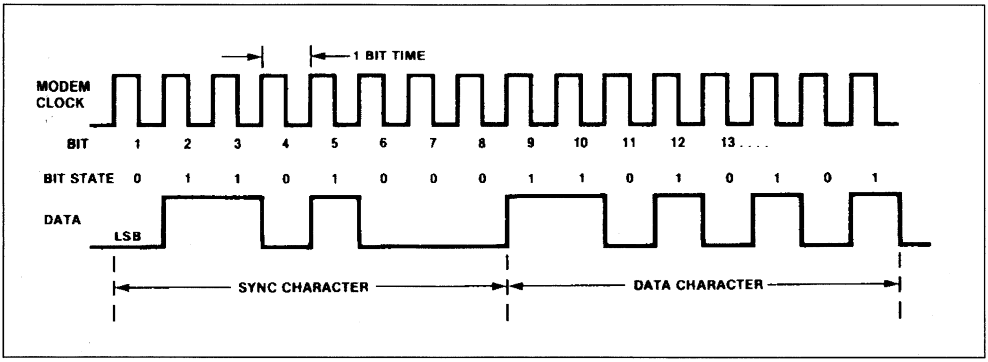
To ensure that the flag is not sent accidentally, SDLC procedures require a binary "0" to be inserted by the transmitter after the transmission of any five contiguous "1s". The receiver then removes the "0" following a received succession of five "1s". Inserted and removed "0s" are not included in the CRC calculation.

The address field is 8 bits long and designates the number of secondary station to which the commands or data from the primary station are sent.The control field is eight bits long and is used to initiate all SDLC activities.

The SCC can also serve the High-level synchronous Data Link Communication (HDLC) protocol, which is identical to SDLC except for differences in framing.

SDLC Loop Mode. The SCC supports SDLC Loop mode in addition to normal SDLC. SDLC Loop mode is very similar to normal SDLC but is usually used in application where a point-to-point network is not ap-

Figure 8 : Monosync Data Character Format.


1 BIT TIME
MODEM CLOCK
BIT 1 2 3 4 5 6 7 8 9 10 11 12 13....
BIT STATE 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1
DATA
LSB
SYNC CHARACTER
DATA CHARACTER

OVERVIEW (cont'd)

propriate (for example, Point-Of-Sale terminals). In an SDLC Loop there is a primary station, called the controller, that manages the message traffic flow on the loop, and there are any number of secondary stations.

A secondary station in an SDLC loop is always listening to the messages being sent around the loop, and must pass these messages to the rest of the loop by retransmitting them with a one-bit-time delay. The secondary station can only place its own message on the loop at specific times. The controller signals that secondary stations may transmit messages by sending a special character, called an EOP (End of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages this bit pattern is unique and thus is easily recognized.

When a secondary station has a message to transmit and it recognizes an EOP on the line , the first thing that it does is to change the last 1 or the EOP to a "0" before transmitting it. This turns the EOP into a Flag sequence. The secondary station now places its message on the loop and terminates its message with an EOP. Any secondary stations further down the loop with messages to transmit can then append its message to the message of the first secondary station by the same process. All secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop, except upon recognizing an EOP.

There are also restrictions as to when and how a secondary station physically becomes part of the loop. A secondary station that has just powered up must monitor the loop, without the one-bit-time delay, until it recognizes an EOP. When an EOP is recognized the one-bit-time delay is switched on. This does not disturb the loop because the line is marking idle between the time that the controller sends the EOP and the time that it receives the EOP back. The secondary station that has gone on-loop cannot place a message on the loop until the next time that an EOP is issued by the controller. A secondary station goes off-loop in a similar manner. When given a command to go off-loop, the secondary station waits until the next EOP to remove the one-bit-time delay.

To operate the SCC in SDLC Loop mode, the SCC must first be programmed just as if normal SDLC were to be used. Loop mode is then selected by writing the appropriate control word in WR10. The SCC

Figure 9 : Bisynchronous Message Format.

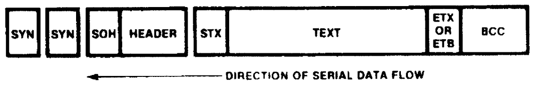


Figure 10 : External Sync Format.

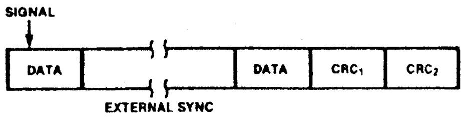
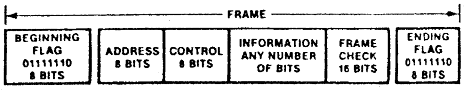


Figure 11 : SDLC Message Format.


FRAME
BEGINNING FLAG 01111110 8 BITS
ADDRESS 8 BITS
CONTROL 8 BITS
INFORMATION ANY NUMBER OF BITS
FRAME CHECK 16 BITS
ENDING FLAG 01111110 8 BITS

OVERVIEW (cont'd)

is now waiting for the EOP so that it can go on loop. While waiting for the EOP, the SCC ties TxD to RxD with only the internal gate delays in the signal path. When the first EOP is recognized by the SCC, the Break/Abort/EOP bit is set in RR0, generating an External/Status interrupt (if so enabled). At the same time, the On-Loop bit in RR10 is set to indicate that the SCC is indeed on-loop, and a one-bit time delay is inserted in the TxD to the RxD patch.

The SCC is now on-loop but cannot transmit a message until a flag and the next EOP are received. The requirement that a flag be received ensures that the SCC cannot erroneously send messages until the controller ends the current polling sequence and starts another one.

A secondary station on the loop is prohibited from transmitting a message during a polling sequence unless it captures the line at the moment the EOP passes by. The SCC does this automatically. If the CPU in the secondary station with SCC needs to transmit a message, the Go-Active-On-Poll bit in WR10 must be set. If this bit is set when the EOP is detected, the SCC changes the EOP to a flag and starts sending another flag. The EOP is reported in the Break/Abort/EOP bit in RR0 and the CPU should write its data bytes to the SCC, just as in normal SDLC frame transmission. When the frame is complete and CRC has been sent, the SCC closes with a flag and reverts to One-Bit-Delay mode. The last zero of the flag, along with the marking line echoed from the RxD pin, form an EOP for secondary stations further down the loop. If the Go-Active-On-Poll bit is not set at the time the EOP passes by, the SCC cannot send a message until a flag (terminating the current polling sequence) and another EOP are received. While the SCC is actually transmitting a message, the loop-sending bit in R10 is set to indicate this.

If SDLC loop is de-selected, the SCC is designed to exit from the loop gracefully . When SDLC Loop mode is de-selected by writing to WR10, the SCC waits until the next polling cycle to remove the on-bit time delay. If a polling cycle is in progress at the time the command is written, the SCC finishes sending any message that it may be transmitting, ends with an EOP, and disconnects TxD from RxD. If no message was in progress, the SCC immediately disconnects TxD from RxD. To ensure proper loop operation after the SCC goes off the loop, and until the external relays take the SCC completely out of the loop, the SCC should be programmed for Mark idle instead of Flag idle. When the SCC goes off the loop, the On-Loop bit is reset.

The SCC allows the user the option of using NRZI in SDLC Loop mode by programming WR20 appropriately. With NRZI encoding, the outputs of secondary stations in the loop may be inverted from their inputs because of messages that they have transmitted. Removing the stations from the loop (removing the one-bit time delay) may cause problems further down the loop because of extraneous transitions on the line. The SCC avoids this problem by making transparent adjustements at the end of each frame it sends in response to an EOP. A response frame from the SCC is terminated by a flag and an EOP. Normally, the flag and the EOP share a zero, but if such sharing would cause the RxD and TxD pins to be of opposite polarity after the EOP, the SCC adds another zero between the flag and the EOP. This causes an extra line transition so that RxD and TxD are identical after the EOP is sent. This extra zero is completely transparent because it only means that the flag and the EOP no longer share a zero. All that a proper loop exit needs, therefore, is the removal of the one-bit time delay.

I/O Capabilities.

The SCC can work with three basic forms of I/O operations : polling, interrupts, and block transfer. All three I/O types involve register manipulation during initialization and data transfer.

Polling. During a polling sequence, the status of Read Register 0 is examined in each channel. This register indicates whether or not a receive or transmit data transfer is needed and whether or not any special conditions are present, e.g., errors.

This method of I/O transfer avoids interrupts. All interrupt functions must be disabled in order to operate the device in a polled environment. With no interrupts enabled, this mode of operation must initiate a read cycle of Read Register 0 to detect an incoming character before jumping to a data handler routine.

Interrupts. The SCC provides interrupt capability through the use of pins and a hardware scheme that enhances the maximum speed of serial data. Whenever the interrupt ($\overline{INT}$) pin is active, the SCC is ready to transfer data.
Read and write registers are programmed so that an interrupt vector points to an interrupt service routine. The interrupt vector can also be modified to reflect various status conditions. Therefore, as many as eight different interrupt routines can be referenced.

Transmit interrupts, receive interrupts, and external/status interrupts are the main sources of interrupts. Each interrupt source is enabled under pro-

OVERVIEW (cont'd)

gram control, with channel A having a higher priority than channel B and with receive, transmit, and external/status interrupts prioritized respectively within each channel.

Block Transfers. The SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{W/REQ}$ output in conjunction with the Wait/Request bits in Write Register 1.

The $\overline{W/REQ}$ output can be defined by software as a $\overline{WAIT}$ line in the CPU Block Transfer mode or as a $\overline{REQUEST}$ line in the DMA Block Transfer mode.

To a DMA controller, the SCC REQUEST output indicates that the SCC is ready to transfer data to or from memory. To the CPU, the $\overline{WAIT}$ output indicates that the SCC is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

INTERFACING THE SCC

This section covers the details of interfacing the Z8530 to a system. The general timing requirements for the device are described in the hardware information section.

Interfacing the Z8530

Two control signals, $\overline{RD}$ and $\overline{WR}$, are used by the Z8530 to time bus transactions. In addition, four other control signals, $\overline{CE}$, D/ $\overline{C}$, A/ $\overline{B}$ and $\overline{INTACK}$, are used to control the type of bus transaction that will occur. A bus transaction start when the addresses on D/ $\overline{C}$ and A/ $\overline{B}$ are asserted before $\overline{RD}$ or $\overline{WR}$ fall. The coincidence of $\overline{CE}$ and $\overline{RD}$ or $\overline{CE}$ and $\overline{WR}$ latches the state of D/ $\overline{C}$ and A/ $\overline{B}$ and starts the internal operation. The $\overline{INTACK}$ signal must have been previously sampled High by a rising edge of PCLK for a read or write cycle to occur. In addition to sampling $\overline{INTACK}$, PCLK is used by the interrupt section to set the IP bits. The Z8530 generates internal control signals in response to a register access. Since $\overline{RD}$ and $\overline{WR}$ have no phase relationship with PCLK, the circuitry generating these internal control signals provides time for metastable conditions to disappear. This results in a recovery time related to PCLK. This recovery time applies only between transactions involving the Z8530, and any interventing transactions are ignored. This recovery time is four PCLK cycles, measured from the falling edge of $\overline{RD}$ or $\overline{WR}$ in the case of a read or write of any register.

Z8530 Read Cycle Timing. The Read cycle timing for the Z8530 is shown in Figure 12. The address on A / $\overline{B}$ and D/ $\overline{C}$ is latched by the coincidence of $\overline{RD}$ and $\overline{CE}$ active. $\overline{CE}$ must remain LOW and $\overline{INTACK}$ must remain HIGH throughout the cycle. The Z8530 bus drivers are enabled while $\overline{CE}$ and $\overline{RD}$ are both LOW. A read with D/ $\overline{C}$ HIGH does not disturb the state of the pointers and a read cycle with D/ $\overline{C}$ LOW resets the pointers to zero after the internal operation is complete.

Z8530 Write Cycle Timing. The Write cycle timing for the Z8530 is shown in Figure 13. The address on A / $\overline{B}$ and D/ $\overline{C}$, as well as the data on D0-D7, is latched by the coincidence of $\overline{WR}$ and $\overline{CE}$ active. $\overline{CE}$ must remain LOW and $\overline{INTACK}$ must remain HIGH throughout the cycle. A write cycle with D/ $\overline{C}$ HIGH does not disturb the state of pointers and a write cycle with D/ $\overline{C}$ LOW resets the pointers to zero after the internal operation is complete.

Z8530 Interrupt Acknowledge Cycle Timing. The interrupt Acknowledge cycle timing for the Z8530 is shown in Figure 14. The state of $\overline{INTACK}$ is latched by the rising edge of PCLK. While $\overline{INTACK}$ is LOW, the state of A/ $\overline{B}$, $\overline{CE}$, D/ $\overline{C}$, and $\overline{WR}$ are ignored. Between the time $\overline{INTACK}$ is first sampled LOW and the time $\overline{RD}$ falls, the internal and external IEI/ IEO daisy chains settle ; this is A.C. parameter #38 TdIAi(RD). If there is an interrupt pending in the Z8530, and IEI is HIGH when RD falls, the Interrupt Acknowledge cycle was intended for the Z8530. This being the case, the Z8530 sets the appropriate Interrupt Under Service latch, and places an interrupt vector on D0-D7. If the falling edge of $\overline{RD}$ sets an IUS bit in the Z8530, the $\overline{INT}$ pin goes active in response to the falling edge. Note that there should be only one RD per Acknowledge cycle. Another important fact is that the IP bits in the Z8530 are updated by PCLK divided by two, and this clock to update IPs is stopped while the pointers point to RR2 and RR3. This prevents data changing during a read, but will delay interrupt requests if the pointers are left pointing at these registers.

Z8530 Register Access. The registers in the Z8530 are accessed in a two-step process, using a Register Pointer to perform the addressing. To access a particular register, the pointer bits must be set by writing to WR0 the pointer bits may be written in either channel because only one set exists in the Z8530. After the pointer bits are set, the next read

INTERFACING THE SCC (cont'd)

or write cycle of the Z8530 having D/$\overline{C}$ LOW will access the desired register. At the conclusion of this read or write cycle the pointer bits are reset to "0s", so that the next control write will be to the pointers in WR0. A read or RR8 (the receive data buffer) or a write to WR8 (the transmit data buffer) may either be done in this fashion or by accessing the Z8530 having D/$\overline{C}$ pin HIGH. A read or write with D/$\overline{C}$ HIGH accesses the data registers directly, and independently, of the state of the pointer bits. This allows single-cycle access to the data registers and does not disturb the pointer bits. The fact that the pointer bits are reset to "0", unless explicitly set otherwise, means that WR0 and RR0 may also be accessed in a single cycle. That is, it is not necessary to write the pointer bits with "0" before accessing WR0 or RR0. There are three pointer bits in WR0, and these allow access to the registers with addresses 0 through 7. Note that a command may be written to WR0 at the same time that the pointer

Figure 12 : Z8530 Read Cycle Timing.

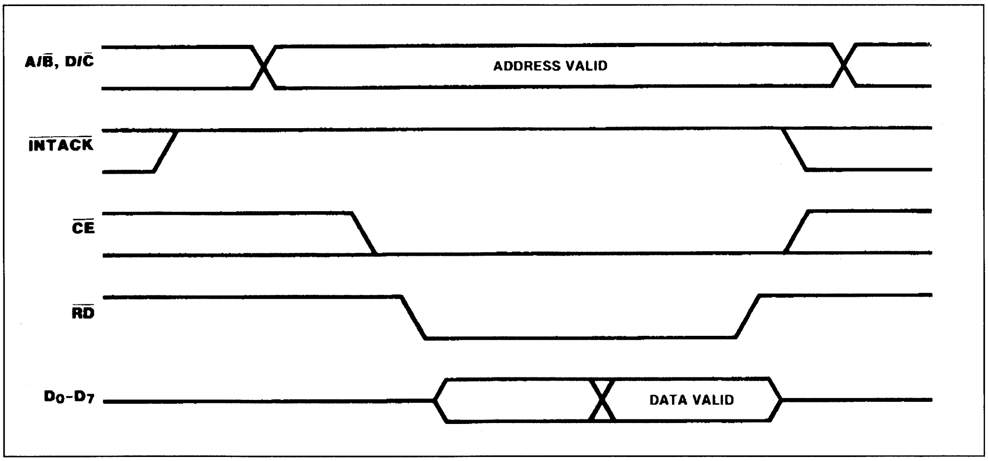
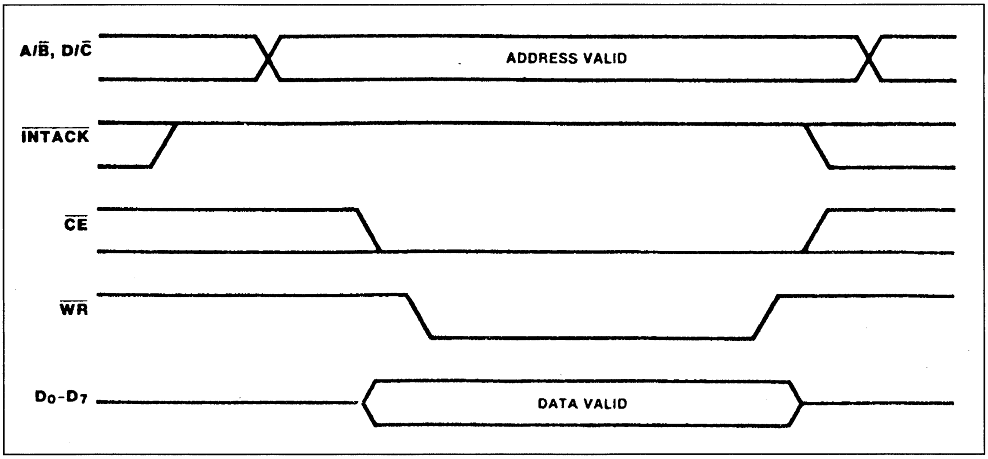
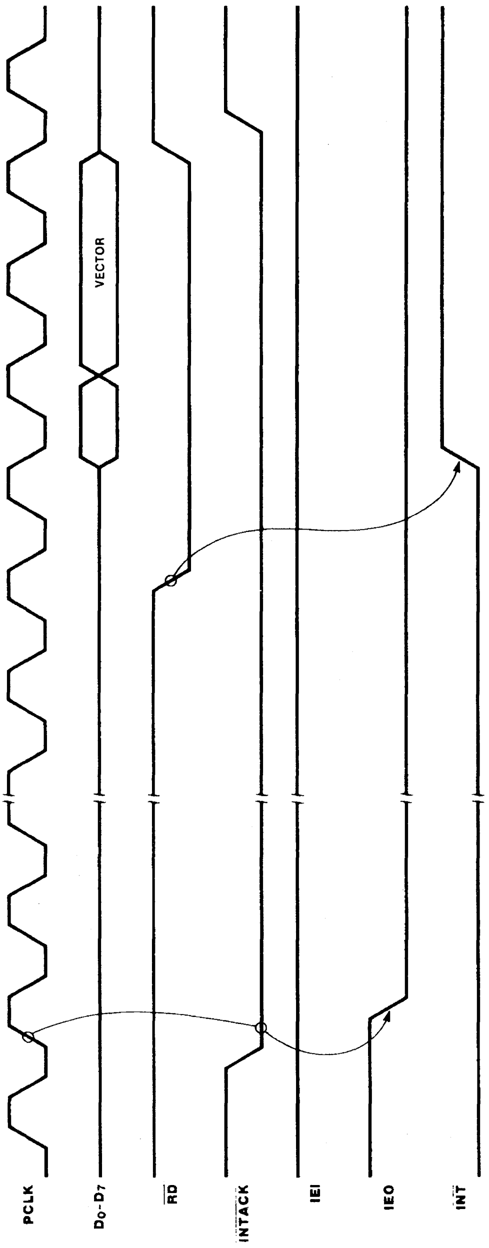


Figure 13 : Z8530 Write Cycle Timing.



INTERFACING THE SCC (cont'd)

Figure 14 : Z8530 Interrupt Acknowledge Details.



INTERFACING THE SCC (cont'd)

bits are written. To access the registers with addresses 8 through 15, a special command must accompany the pointer bits. This precludes concurrently issuing a command when pointing to these registers. The register map for the AmZ8530 is shown in Table 2. If for some reason, the state of the pointer bits is unknown they may be reset to "0" by performing a read cycle with the D/ $\overline{C}$ pin held LOW. Once the pointer bits have been set, the desired channel is selected by the state of the A/ $\overline{B}$ pin during the actual read or write of the desired register.

Z8530 Reset. The Z8530 may be reset by either hardware or software. Hardware reset occurs when $\overline{RD}$ and $\overline{WR}$ are both LOW, simultaneously, which is normally an illegal condition. As long as both $\overline{RD}$ and $\overline{WR}$ are LOW, the Z8530 recognizes the reset condition. Once this condition is removed, however, the reset condition is asserted internally for an additional four to five PCLK cycles. During this time any attempt to access the Z8530 will be ignored. The Z8530 has three software resets, encoded into command bits in WR9. There are two channel resets, which affect only one channel in the device and some of the bits in the write registers. The third command forces the same result as does a hardware reset. As in the case of a hardware reset, the Z8530 stretches the reset signal an additional four to five PCLK cycles beyond the ordinary valid access recovery time. The bits in WR9 may be written at the same time as the reset command because these bits are affected only by a hardware reset.

The reset values of the various registers are shown in Figure 15.

Table 2 : Z8530 Register Map.

A/B	PNT$_2$	PNT$_1$	PNT$_0$	WRITE	READ
0	0	0	0	WR0B	RR0B
0	0	0	1	WR1B	RR1B
0	0	1	0	WR2	RR2B
0	0	1	1	WR3B	RR3B
0	1	0	0	WR4B	(RR0B)
0	1	0	1	WR5B	(RR1B)
0	1	1	0	WR6B	(RR2B)
0	1	1	1	WR7B	(RR3B)
1	0	0	0	WR0A	RR0A
1	0	0	1	WR1A	RR1A
1	0	1	0	WR2	RR2A
1	0	1	1	WR3A	RR3A
1	1	0	0	WR4A	(RR0A)
1	1	0	1	WR5A	(RR1A)
1	1	1	0	WR6A	(RR2A)
1	1	1	1	WR7A	(RR3A)
With the Point High Command					
0	0	0	0	WR8B	RR8B
0	0	0	1	WR9	RR13B
0	0	1	0	WR10B	RR10B
0	0	1	1	WR11B	(RR15B)
0	1	0	0	WR12B	RR12B
0	1	0	1	WR13B	RR13B
0	1	1	0	WR14B	(RR10B)
0	1	1	1	WR15B	RR15B
1	0	0	0	WR8A	RR8A
1	0	0	1	WR9	(RR13A)
1	0	1	0	WR10A	RR10A
1	0	1	1	WR11A	(RR15A)
1	1	0	0	WR12A	RR12A
1	1	0	1	WR13A	RR13A
1	1	1	0	WR14A	RR14A
1	1	1	1	WR15A	RR15A

INTERFACING THE SCC (cont'd)

Figure 15 : Z8530 Register Reset Values.

HARDWARE RESET								CHANNEL RESET								
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WR0
0	0	.	0	0	.	0	0	0	0	.	0	0	.	0	0	WR1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	WR2
.	.	.	.	.	.	.	0	.	.	.	.	.	.	.	0	WR3
.	.	.	.	.	1	.	.	.	.	.	.	.	1	.	.	WR4
0	.	.	0	0	0	0	.	0	.	.	0	0	0	0	.	WR5
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	WR6
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	WR7
1	1	0	0	0	0	.	.	.	.	0	.	.	.	.	.	WR9
0	0	0	0	0	0	0	0	0	.	.	0	0	0	0	0	WR10
0	0	0	0	1	0	0	0	.	.	.	.	.	.	.	.	WR11
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	WR12
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	WR13
.	.	1	0	0	0	0	0	.	.	1	0	0	0	.	.	WR14
1	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	WR15
0	1	.	.	.	1	0	0	0	1	.	.	.	1	0	0	RR0
0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	RR1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RR3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RR10

SGS-THOMSON MICROELECTRONICS

I/O PROGRAMMING CAPABILITIES

Regardless of the version of the SCC, all communication modes can use a choice of polling, interrupt and block transfer. These modes must be selected by the user to select the proper hardware and software required to supply data at the rate required.

Polling

This is the simpliest mode to implement. The software must poll the SCC to determine when data is to be inputted or outputted from the SCC.In this mode, MIE (WR9 bit 3), and Wait/ DMA Request Enable (WR1 bit 7) are both reset to 0 to disable any requests. The software must then poll RR0 to determine the status of the receive buffer, transmit buffer and external status.

Interrupt Operations

The SCC, as a microprocessor peripheral, will request an interrupt only when it needs servicing. This allows the CPU to perform other operations while the SCC does not need service. The SCC has an internal priority resolution method to allow the highest priority interrupt to be serviced first.

The SCC is flexible with its interrupt method. The interrupt may be acknowledged with a vector transferred, acknowledged without a vector, or not acknowledged at all.

Interrupt Without Acknowledge. In this mode, the Interrupt Acknowledge signal does not have to be generated. This allows a simpler hardware design that does not have to meet the Interrupt acknowledge timing. Soon after the $\overline{\text{INT}}$ goes active, the interrupt controller will jump to the interrupt routine. In the interrupt routine, the code must read RR2 from Channel B to read the vector including status. When the vector is read from Channel B, it always includes the status regardless of the VIS bit (WR9 bit 0). The status given will decode the highest priority interrupt pending at the time it is read. The vector is not latched so that the next read could produce a different vector if another interrupt occurs. The register is disabled from change during the read operation to prevent an error if a higher interrupt occurs exactly during the read operation.

Once the status is read, the interrupt routine must decode the interrupt pending, and clear the condition. Removing the interrupt condition will clear the IP and bring $\overline{\text{INT}}$ inactive, as long as there are no other IP bits set. For example, writing a character to the transmit buffer will clear the transmit buffer empty IP.

When the interrupt IP, decoded from the status, is cleared RR2 can be read again. This allows the interrupt routine to clear all of the IP's within one interrupt request to the CPU.

Interrupt With Acknowledge. After the SCC brings $\overline{\text{INT}}$ active, the CPU must respond by bringing $\overline{\text{INTACK}}$ active. After enough time has elapsed to allow the daisy-chain to settle, the SCC will set the IUS bit for the highest priority IP. If the No Vector bit is not set (WR9 bit 1), the SCC will then place the interrupt vector on the data bus during a read. To speed the interrupt response time, the SCC can also modify 3 bits in the vector to indicate status. To include the status, the VIS bit (WR9 bit 0) must be set. The service routine must then clear the interrupting condition. For example, writing a character to the transmit buffer will clear the transmit buffer empty IP. After the interrupting condition is cleared, the routine can read RR3 to determine if any other IP's are set and clear them. At the end of the interrupt routine, a Reset IUS command (WR0) must be issued to unlock the daisy-chain and allow lower-priority interrupt requestes. This is the only way, short of a software or hardware reset, that an IUS bit may be reset.

If the No Vector bit (WR9 bit 1) is set, the SCC will not place the vector on the data bus. An interrupt controller must then vector the code to the interrupt routine. The interrupt routine must then read RR2 from Channel B to read the status. This is the same as the case of an interrupt without an acknowledge except the IUS is set and the vector will not change until the Reset IUS command in RR0 is issued.

Interrupt Sources. Each channel in the SCC contains 3 sources of interrupt, making a total of 6. These 3 sources of interrupts are the receiver, the transmitter, and External/ Status conditions. In addition, there are several conditions that may cause these interrupts.

The receive interrupt request may either be caused by a receive character available or a special condition. The receive character available interrupt is generated when a character is loaded into the FIFO and is ready to be read. The special conditions are receive FIFO overrun, CRC/ framming error. End of frame, and parity. The parity special condition can be included as a special condition or not depending on bit 2 WR1. The special condition status can be read from RR1.
The transmit interrupt request has only one source. It can only be set when the transmit buffer goes from

I/O PROGRAMMING CAPABILITIES (cont'd)

full to empty. Note that this means that the transmit interrupt will not be set until after the first character is written to the SCC.

The External/ status Interrupts have several sources which may be individually enabled in WR15. The sources are zero count, DCD, Sync/ Hunt, CTS, transmitter underrun/ EOM and Break/ Abort.

Each source of interrupt in the SCC has three control/status bits associated with it. There are Interrupt Enable (IE), Interrupt Pending (IP), and Interrupt Under Service (IUS) (see Figure 16). The IE bit is written by the processor and serves to control interrupt requests from the SCC. If the IE bit is set for a given source of interrupt, then that source may cause an interrupt request when all of the necessary conditions are met. If the IE bit is reset, no interrupt request will be generated by that source. The IE bits are write-only in the SCC. The IP bit for a given source of interrupt condition in the SCC and is reset directly by the processor, or indirectly by some action that the processor may take. If the corresponding IE bit is not set, the IP bits in the SCC are read-only in RR3A. The IUS bits are completely hidden from the processor's view. An IUS is set during an Interrupt Acknowledge cycle for the highest-priority IP. See Table 3 for the interrupt priority. IUS is used to control the operation of the interrupt daisy chain by masking lower-priority interrupts. At the end of an interrupt service routine, the processor must issue a Reset Highest IUS command in WR0 to allow lower-priority interrupts. This is the only way, short of a software or hardware reset, that an IUS bit may be reset.

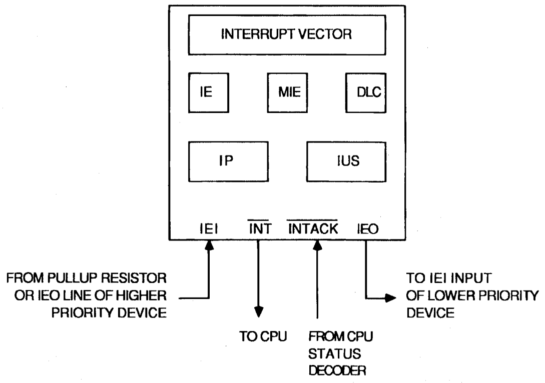
Table 3 : Interrupt Source Priority.

Receiver Channel A Transmit Channel A	High
External / Status Channel A	↓
Receiver Channel B Transmit Channel B	↓
External / Status Channel B	Low

Daisy-chain Priority Resolution. The six sources of interrupt in the SCC are prioritized in a fixed order via a daisy chain ; provision is made, via the IEI and IEO pins, for use of an external daisy chain as well. All Channel A interrupts are higher-priority than any Channel B interrupts, with the receiver, transmitter, and External/ Status interrupts prioritized in that order within each channel. The SCC requests an interrupt by pulling the $\overline{\text{INT}}$ pin Low from its open-drain state. This is controlled by the IP bits and the IEI input, among other things. A flowchart of the interrupt sequence for the SCC is shown in Figure 17. The internal daisy chain links the six sources of interrupt in a fixed order, chaining the IUS bits for each source. While an IUS is set, all lower-priority interrupt requests are masked off ; during an Interrupt Acknowledge cycle the IP bits are also gated into the daisy chain. This insures that the highest-priority IP will be selected to have its IUS set. The internal daisy chain may be controlled by the MIE bit in WR9. This bit, when reset, has the same effect as pulling the IEI pin Low, thus disabling all interrupt requests.

External Daisy Chain Operations. The SCC generates an interrupt request by pulling $\overline{\text{INT}}$ Low, but

Figure 16 : Peripheral Interrupt Structure.


INTERRUPT VECTOR
IE
MIE
DLC
IP
IUS
IEI
$\overline{\text{INT}}$
$\overline{\text{INTACK}}$
IEO
FROM PULLUP RESISTOR OR IEO LINE OF HIGHER PRIORITY DEVICE
TO CPU
FROM CPU STATUS DECODER
TO IEI INPUT OF LOWER PRIORITY DEVICE

I/O PROGRAMMING CAPABILITIES (cont'd)

only if such interrupt requests are enabled (IE is 1, MIE is 1), an IP is set without a higher-priority IUS being set, or no higher-priority IUS being set, or no higher-priority interrupt is being serviced (IEI is High), or no Interrupt Acknowledge transaction is taking place. It is not pulled Low by the SCC at this time, but instead continues to follow IEI until an Interrupt Acknowledge transaction occurs. Some time after $\overline{INT}$ has been pulled Low, the processor initiates an Interrupt Acknowledge transaction. Between the time the SCC recognizes that an Interrupt Acknowledge cycle is in progress and the time during the acknowledge that the processor requests an interrupt vector, the IEI/IEO daisy chain settles. Any peripheral in the daisy chain having an Interrupt Pending (IP is 1) or an Interrupt Under Service (IUS is 1) holds its IEO line Low and all others make IEO follow IEI.

When the processor requests an interrupt vector, only the highest-priority interrupt source with a pending interrupt (IP is 1) has its IEI input High, its IE bit set to "1", and its IUS bit set to "0". This is the interrupt source being acknowledged, and at this point it sets its IUS bit to "1". If its NV bit is "0", the SCC identifies itself by placing the interrupt vector from WR2 on the data bus. If the NV bit is "1", the SCC data bus remains floating, allowing external logic to supply a vector. If the VIS bit in the SCC is "1", the vector also contains status information, encoded as shown in Table 4, which further describes the nature of the SCC interrupt. If the VIS bit is "0", the vector held in WR2 is returned without modification. If the SCC is programmed to include status information in the vector, this status may be encoded and placed in either bits 1-3 or in bits 4-6. This operation is selected by programming the Status High/Status Low bit in WR9. At the end of the interrupt service routine, the processor should issue the Reset Highest IUS command to unlock the daisy chain and allow lower-priority interrupt requests. The IP is reset during the interrupt service routine either directly by command, or indirectly, through some action taken by the processor. The external daisy chain may be controlled by the DLC bit in WR9. This bit, when set, forces IEO Low, disabling all lower-priority devices.

Receive Interrupts. The Receive Interrupt mode is controlled by WR1 bits 4 and 3. These select one of the four interrupt modes. The four modes are, Interrupt disabled, Interrupt on first character or special conditions, Interrupt on all characters or special conditions, and Interrupt on special conditions.

Receive Interrupts Disabled. This mode prevents the receiver from requesting an interrupt. It is used in a polled environment where either the status bits in RR0 or the modified vector in RR2 (Channel B) is read. Although the receiver interrupts are disabled, the interrupt logic can still be used to provide status.

When these bits indicate that a received character has reached the top of the FIFO, the status in RR1 should be checked and then the data should be read. If status is to be checked, it must be done before the data is read, because the act of reading the data pops both the data and error FIFOs.

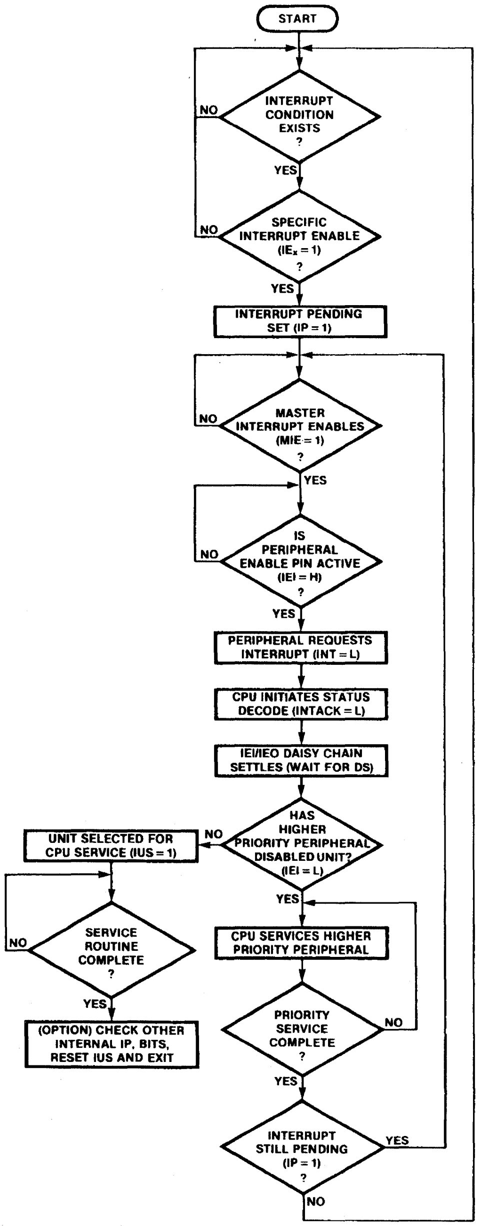
Table 4 : Interrupt Vector Modification.

V3	V2	V1	Status High / status Low = 0
V4	V5	V6	Status High / status Low = 1
0	0	0	Ch B Transmit Buffer Empty
0	0	1	Ch B External / Status Change
0	1	0	Ch B Receive Character Avail.
0	1	1	Ch B Special Receive Condition
1	0	0	Ch A Transmit Buffer Empty
1	0	1	Ch A External / status Change
1	1	0	Ch A Receive Character Avail.
1	1	1	Ch A Special Recieve Condition

Receive Interrupt On First Character Or Special Condition. This mode is designed for use with DMA transfers of the receive characters. After this mode is selected, the first character received, or the first character already stored in the FIFO, will set the receiver IP. This IP will be reset when this character is removed from the SCC. No further receive interrupts will occur until the processor issues an Enable Interrupt on Next Receive Character command in WR0 until a special receive condition occurs. The SCC recognizes several special receive conditions. A receive overrun (where a character in the FIFO is written over) is a special receive condition, as is a framing error in Asynchronous mode, or the end-of-frame condition in SDLC mode. In addition, if D2 of WR1 is set, any character with a parity error will generate a special receive condition interrupt. The correct sequence of events when using this mode is to first select the mode and wait for the receive character available interrupt. When the interrupt occurs the processor should read the character and then enable the DMA to transfer the remaining characters. A special receive condition interrupt may occur any time after the first character is received, but is guaranteed to occur after the character having the special condition has been read. The status is not lost in this case, however, because the FIFO is locked by the special condition. In the service routine the processor should read RR1 to obtain the status, and may read the data again if necessary. The FIFO is unlocked by issuing an Error Reset command in WR0. If the special condition was End-

I/O PROGRAMMING CAPABILITIES (cont'd)

Figure 17 : Interrupt Flowchart.


START
INTERRUPT CONDITION EXISTS ?
NO
YES
SPECIFIC INTERRUPT ENABLE (IEx = 1) ?
NO
YES
INTERRUPT PENDING SET (IP = 1)
MASTER INTERRUPT ENABLES (MIE = 1) ?
NO
YES
IS PERIPHERAL ENABLE PIN ACTIVE (IEI = H) ?
NO
YES
PERIPHERAL REQUESTS INTERRUPT (INT = L)
CPU INITIATES STATUS DECODE (INTACK = L)
IEI/IEO DAISY CHAIN SETTLES (WAIT FOR DS)
HAS HIGHER PRIORITY PERIPHERAL DISABLED UNIT? (IEI = L)
NO
YES
UNIT SELECTED FOR CPU SERVICE (IUS = 1)
SERVICE ROUTINE COMPLETE ?
NO
YES
(OPTION) CHECK OTHER INTERNAL IP, BITS, RESET IUS AND EXIT
CPU SERVICES HIGHER PRIORITY PERIPHERAL
PRIORITY SERVICE COMPLETE ?
NO
YES
INTERRUPT STILL PENDING (IP = 1) ?
YES
NO

I/O PROGRAMMING CAPABILITIES (cont'd)

of-Frame, the processor should now issue the Enable Interrupt on Next Receive Character command to prepare for the next frame. The first character interrupt and special condition interrupt are distinguished by the status included in the interrupt vector. In all other respects they are identical, including sharing the IP and IUS bits.

Interrupt On All Receive Characters Or Special Conditions. This mode is designed for an interrupt-driven system. In this mode the SCC will set the receiver IP on every received character, whether or not it has a special receive condition. This includes characters already in the FIFO when this mode is selected. In this mode of operation the IP is reset when the character is removed from the FIFO, so if the processor requires status for any character, this status must be read before the data is removed from the FIFO. The special receive conditions are identical to those previously mentioned, and as before, the only difference between a "receive character available" interrupt and a "special receive condition" interrupt is the status encoded in the vector. In this mode a special receive condition does not lock the receive data FIFO so that the service routine must read the status in RR1 before reading the data. At moderate to high data rates, where the interrupt overhead is significant, time can usually be saved by checking for another received character before exiting the service routine. This technique eliminates the Interrupt Acknowledge and the processor-state-saving time, but care must be exercised because this receive character must be checked for special receive conditions before it is removed from the SCC.

Receive Interrupt On Special Conditions. This mode is designed for use with DMA transfers of the receive characters. In this mode, only receive characters with special conditions will cause the receiver IP to be set. All other characters are assumed to be transferred via DMA. No special initialization sequence is needed in this mode. Usually the DMA is initialized and enabled, and then this mode is selected in the SCC. A special receive condition interrupt may occur at any time after this mode is selected but the logic guarantees that the interrupt will not occur until after the character with the special condition has been read from the SCC . The special condition locks the FIFO so that the status will be valid when read in the interrupt service routine, and it guarantees that the DMA will not transfer any characters until the special condition has been serviced. In the service routine the processor should read RR1 to obtain the status and unlock the FIFO by issuing an Error Reset command. DMA transfer of the receive characters will then resume.

Transmit Interrupts. Transmit interrupts are controlled by the Transmit Interrupt Enable bit (D1) in WR1. If the interrupt capabilities of the SCC are not required, polling may be used. This is selected by disabling the transmit interrupts and polling the Transmit Buffer Empty bit in RR0. When the Transmit Buffer Empty is set a character may be written to the SCC without fear of writing over previous data. Another way of polling the SCC is to enable the transmit interrupt and then reset the MIE bit in WR9. The processor may then poll the IP bits in RR3A to determine when the transmit buffer is empty. Transmit interrupts should also be disabled in the case of DMA transfer of the transmitted data.

While the transmit interrupts are enabled the SCC will set the transmit IP whenever the transmit buffer becomes empty. This means that the transmit buffer must have been full before the transmit IP can be set. Thus when the transmit interrupts are first enabled, the transmit IP will not be set until after the first character is written to the SCC. In synchronous modes one other condition can cause the transmit IP to be set. This occurs at the end of a transmission after CRC is sent. When the last bit of CRC has cleared the Transmit Shift register and the flag or sync character is loaded into the Transmit Shift register, the SCC will set the transmit IP. Data for the new frame or block to be transmitted may be written at this time. In this particular case the Transmit Buffer Empty bit in RR0 is not set ; only the transmit IP is set. If the transmit Buffer Empty bit is, in fact, set for the transmit interrupt that occurs immediately after CRC transmission, this indicates that data was written while CRC was being sent. This is an indication that the transmitter underflowed, without the CPU being aware of it. The transmit IP is reset either by writing data to the transmit buffer or by issuing the Reset Transmit IP command in WR0. Ordinarily the response to a transmit interrupt is to write more data to the SCC ; however, at the end of a frame or block of data where CRC is to be sent next, the Reset Transmit IP command should be issued in lieu of data.

External/Status Interrupts. There are several sources of External/Status interrupts, each of which may be individually enabled in WR15. The master enable for the External/Status interrupts is located in WR1 (D0). The individual enable bits in WR15 control whether or not latches will be present in the path from the source of interrupt to the status bit in RR0. If an individual enable bit in WR15 is set to "0" the latches are not present in the signal path and the value read in RR0 reflects the current status. An interrupt source whose individual enable in WR15 is "0" is not a source of External/Status interrupts

I/O PROGRAMMING CAPABILITIES (cont'd)

even though the External/Status Interrupt Enable bit is set. When an individual enable in WR15 is set to "1", the latch is present in the signal path. The latches for the sources of External/Status interrupts are not independent. Rather, they all close at the same time as a result of a state change by one of the sources of interrupt. Thus, a read of RR0 returns the current status for any bits whose individual enable is "0" and either the current state or the latched state of the remainder of the bits. To guarantee the current status the processor should issue a Reset External/Status Interrupts command in WR0 to open the latches. The External/Status IP is set by the closing of the latches and remains set as long as they are closed. If the master enable for the External/Status interrupts is not set, the IP will never be set, even though the latches may be present in the signal paths and working as described. Because the latches close on the current status but give no indication of change, the processor must maintain a copy of RR0 in memory. When the SCC generates an External/Status interrupt the processor should read RR0 and determine which condition changed state and take appropriate action. The copy of RR0 in memory must then updated and the Reset External/Status Interrupt command issued. Care must be taken in writing the interrupt service routine for the External/Status interrupts because it is possible for more than one status condition to change state at the same time. All of the latch bits in RR0 should be compared to the copy of RR0 in memory. If none have changed and the ZC interrupt is enabled, the Zero Count condition caused the interrupt.

The operation of the individual enable bits in WR15 for each of the six sources of External/Status interrupts is identical, but subtle differences exist in the operation of each source of interrupt. The six sources are Break/Abort, Underrun/EOM, CTS, DCD, Sync/Hunt and Zero Count. The Break/Abort, Underrun/EOM, and Zero Count conditions are internal to the SCC, while Sync/Hunt may be internal or external, and CTS and DCD are purely external signals. In the following discussions each source is assumed to be enabled, so that the latches are present, and the External/Status interrupts are enabled as a whole. Recall that the External/Status IP is set while the latches are closed and that the state of the signal is reflected immediately in RR0 if the latches are not present.

The Break/Abort status is used in asynchronous and SDLC modes but is always "0" in synchronous modes other than SDLC. In asynchronous modes this bit is set when a break sequence (null character plus framing error) is detected in the receive data stream, and remains set as long as "0s" continue to be received. This bit is reset when a "1" is received. A single null character is left in the receive FIFO each time that the break condition is terminated. This character should be read and discarded. In SDLC mode this bit is set by the detection of an abort sequence, which is seven or more contiguous "1s" in the receive data stream. The bit is reset when a "0" is received. A received abort forces the receiver into Hunt, which is also an external/status condition. Though these two bits change state at roughly the same time, one or two External/ Status interrupts may be generated as a result. The Break/Abort bit is unique in that both transitions are guaranteed to cause the latches to close, even if another External/ Status interrupt is pending at the time these transitions occur. This guarantees that a break or abort will be caught.

The Transmit Underrun/ EOM bit is used in synchronous modes to control the transmission of CRC. This bit is reset by issuing the Reset Transmit Underrun/ EOM command in WR0. However, this transition does not cause the latches to close ; this occurs only when the bit is set. To inform the processor of this fact, the SCC sets this bit when CRC is loaded into the Transmit Shift register. This bit will also be set if the processor issues the Send Abort command in WR0. The bit is always set in Asynchronous mode.

The CTS bit reports the state of the $\overline{CTS}$ input, and the DCD bit reports the status of the DCD input. Both bits latch on either input transition. In both cases, after the Reset External/ Status Interrupt command is issued, if the latches are closed, they remain closed if there is any odd number of transitions on an input ; they will be open if there is an even number of transi-tions on the input.

The Zero Count bit is set when the counter in the baud rate generator reaches a count of "0" and is reset when the counter is reloaded. The latches are closed only when this bit is set to "1", and the status in RR0 always reflects the current status.While the Zero Count IE bit in WR15 is reset this bit is forced to "0" .

There are a variety of ways in which the Sync/ Hunt may be set and reset, depending on the SCC's mode operation. In Asynchronous mode this bit reports the state of the $\overline{SYNC}$ pin, latching on both input transitions. The same is true of External Sync mode. However, if the crystal oscillator is enabled while in Asynchronous mode this bit will be forced to "0" and the latches will not be closed. Selecting the crystal option in External Sync mode is illegal, but the result will be the same. In Synchronous modes other than SDLC the Sync/Hunt reports the

I/O PROGRAMMING CAPABILITIES (cont'd)

Hunt state of the receiver. Hunt mode is entered when the processor issues the Enter Hunt command in WR3. This forces the receiver to search for a sync character match in the receive data stream. Because both transitions of the Hunt bit close the latches, issuing this command will cause an External/Status interrupt. The SCC resets this bit when character synchronization has been achieved, causing the latches to again be closed. In these synchronous modes the SCC will not reenter the Hunt mode automatically ; only the Enter Hunt command will set this bit. In SDLC mode this bit is also set by the Enter Hunt command, but the receiver will also automatically enter the Hunt mode if an Abort sequence is received. The receiver leaves Hunt upon receipt of a flag sequence. Both transitions of the Hunt bit will cause the latches to be closed. In SDLC mode the receiver will automatically synchronize on Flag characters. The receiver is in Hunt mode when it is enabled, so the Enter Hunt command will probably never be needed.

If careful attention is paid to details, the interrupt service routine for External/Status interrupts is straightforward. To determine which bit or bits changed state, the routine should first read RR0 and compare it to a copy from memory. For each changed bit the appropriate action should be taken and the copy in memory updated. The service routine should close with a Reset External/Status interrupts command to reopen the latches. The copy of RR0 in memory should always have the Zero Count bit set to "0", since this will be the state of the bit after the Reset External/Status interrupts command at the end of the service routine. When the processor issues the Reset Transmit Underrun/EOM latch command in WR0, the Transmit Underrun/EOM bit in the copy of RR0 in memory should be reset because this transition does not cause an interrupt.

Block Transfers

The SCC offers several alternatives for the block transfer of data. The various options are selected by WR1 (bits D7 through D5) and WR14 (bit D2). Each channel in the SCC has two pins which may be used to control the block transfer data. Both pins in each channel may be programmed to act as DMA Request signals, and one pin in each channel may be programmed to act as a Wait signal for the CPU. In either mode, it is advisable to select and enable the mode in two separate accesses of the appropriate register. The first access should select the mode and the second access should enable the function. This procedure prevents glitches on the output pins. Reset forces Wait mode, with W/ REQ open-drain.

Wait On Transmit. The Wait function on transmit is selected by setting both D6 and D5 to "0" and then enabling the function by setting D7 of WR1 to "1". In this mode the W/ REQ pin carries the WAIT signal, and is open-drain when inactive and Low when active. When the processor attempts to write to the transmit buffer when it is full, the SCC will assert WAIT until the buffer is empty. This allows the use of a block-move instruction to transfer the transmit data. In the Z8530, WAIT will go active in res-ponse to WR going active, but only if the data buf-fer is being accessed, either directly or via the pointers. The WAIT pin is released in response to the falling edge of PCLK. Details of the timing are shown in Figure 18.

Wait On Receive. The Wait function on receive is selected by setting D6 or WR1 to "0", D5 of WR1 to "1", and then enabling the function by set-tiing D7 of WR1 to "1". In this mode the W/ REQ pin carries the WAIT signal, and is open-drain when inactive and Low when active. When the processor attempts to read data from the receive FIFO when it is empty, the SCC will assert WAIT until a character has reached the top of the FIFO. This allows the use of a block-move instruction to transfer the receive data. In the Z8530, WAIT will go active in res-ponse to RD going active, but only if the receive data FIFO is being accessed, either directly or via the pointers. The WAIT pin is released in response to the falling edge of PCLK. Details of the timing are shown in Figure 19.

DMA Requests. The two DMA request pins W/ REQ and DTR/ REQ can be programmed to be used as DMA requests. The W/ REQ pin can be used as either a transmit or a receive request and the DTR/ REQ pin can only be used as a receive request. For full-duplex operation, the W/ REQ is, therefore, used for transmit and the DTR/ REQ is used for receive. These modes are described below.

DMA Request On Transmit (using W/REQ). The Request on Transmit function is selected by setting D6 of WR to "1", D5 of WR1 to "0", and then enabling the function by setting D7 of WR1 to "1". In this mode the W/ REQ pin carries the REQUEST signal, which is active Low. When this mode is selected, but not yet enabled, the W/ REQ is driven High. When the enable bit is set, REQUEST goes Low if the transmit buffer is empty at the time, or will remain High until the transmit buffer becomes empty. Note that the REQUEST pin will follow the state of the transmit buffer even though the transmitter is disabled. Thus, if the REQUEST is enabled, the DMA may write data to the SCC before the transmitter is enabled. This will not cause a problem in Asynchro-

I/O PROGRAMMING CAPABILITIES (cont'd)

nous mode but may cause problems in Synchronous mode because the SCC will send data in preference to flags or sync characters. It may also complicate the CRC initialization, which cannot be done until after the transmitter is enabled. With only one exception, the REQUEST pin directly follows the state of the transmit buffer in this mode. REQUEST goes Low when the transmit buffer empties and remains Low until the transmit buffer is filled. The SCC generates only one falling edge on REQUEST per character requested and the timing for this is shown in Figure 20. The one exception occurs in synchronous modes at the end of CRC transmission. At the end of CRC transmission, when the closing flag or sync character is loaded into the Transmit Shift register, REQUEST is pulsed High for one PCLK cycle. The DMA may use this falling edge on REQUEST to write the first character of the next frame or block to the SCC.

In the Z8530, REQUEST will go High in response to the falling edge of WR, but only when the appropriate transmit buffer in the SCC is accessed. This is shown in Figure 21.

DMA Request On Transmit (using DIR/ REQ). A second Request on Transmit function is available on the DTR/ REQ pin. This mode is selected by setting D2 of WR14 to "1". When this bit is set to "1", REQUEST goes Low if the transmit buf-fer is empty at the time, or will go High until the transmit buffer becomes empty. While D2 of WR14 is set to "0", the DTR/ REQ pin is DTR and follows the inverted state of D7 in WR5. This pin will be High after a channel or hardware reset and in the DTR mode. In the Request mode REQUEST will follow the state of the transmit buffer even though the transmitter is disabled. Thus if REQUEST is enabled before the transmitter is enabled, the DMA may write data to the SCC before the transmitter is enabled. This will not cause a problem in Asynchronous mode, but may cause problems in Synchronous mode because the SCC will send data in preference to flags or sync characters. It may also complicate the CRC initialization, which cannot be done until after the transmitter is enabled. With only one exception, the REQUEST pin directly follows the state of the transmit buffer in this mode. REQUEST goes Low when the transmit buffer empties and remains Low until the transmit buffer is filled. The SCC generates only one falling edge on REQUEST per character requested. The one exception occurs in synchronous modes at the end of CRC transmission. At the end of CRC transmission, when the closing flag or sync character is loaded into the Transmit Shift register, REQUEST is pulsed High for one PCLK cycle. The DMA may use this falling edge on REQUEST to write the first character of the next frame or block to the SCC. The Request signal on DTR/ REQ differs from the one on W/ REQ in that it does not go immediately High in response to the access which writes to the transmit buffer. This is because the registers in the SCC are not written during the actual access, but are delayed by some number of PCLK cycles. The Request signal on DTR/ REQ follows the state of the transmit buffer exactly while the Request signal on W/ REQ goes inactive in anticipation of the transmit buffer becoming full. The timing of the Request signal on both pins is shown in Figure 21.

DMA Request On Receive. The Request on Receive function is selected by setting D6 and D5 of WR1 to "1" and then enabling the function by setting D7 of WR1 to "1". In this mode the W/ REQ pin carries the REQUEST signal, which is active Low. When this mode is selected, but not yet enabled, the W/ REQ pin is driven High. When the enable bit is set REQUEST goes Low if the receive buffer contains a character at the time, or will remain High until a character enters the receive buffer. Note that the REQUEST pin will follow the state of the receive buffer even though the receiver is disabled. Thus, if the receiver is disabled and REQUEST is still enabled, the DMA will transfer the previously received data correctly. In this mode the REQUEST pin directly follows the state of the receive buffer with only one exception. REQUEST goes Low when a character enters the receive buffer and remains Low until this character is removed from the receive buffer. The SCC generates only one falling edge on REQUEST per character transfer requested and the timing for this is shown in Figure 22. The one exception occurs in the case of a special receive condition in the Receive Interrupt on First Character or Special Condition mode, or the Receive Interrupt on Special Condition Only mode. In the two interrupt modes any receive character with a special receive condition is locked at the top of the FIFO until an Error Reset command is issued. This character in the receive FIFO would ordinarily cause additional DMA Requests after the first time it is read. However, the logic in the SCC guarantees only one falling edge on REQUEST by holding REQUEST High from the time the character with the special receive condition is read, and the FIFO locked, until after the Error Reset command has been issued. Once the FIFO is unlocked by the Error Reset command, REQUEST again follows the state of the receive buffer. In the Z8530, REQUEST will go High in response to the falling edge of RD, but only when the appropriate receive buffer in the SCC is accessed. This is shown in Figure 23.

I/O PROGRAMMING CAPABILITIES (cont'd)

Figure 18 : Wait on Transmit.

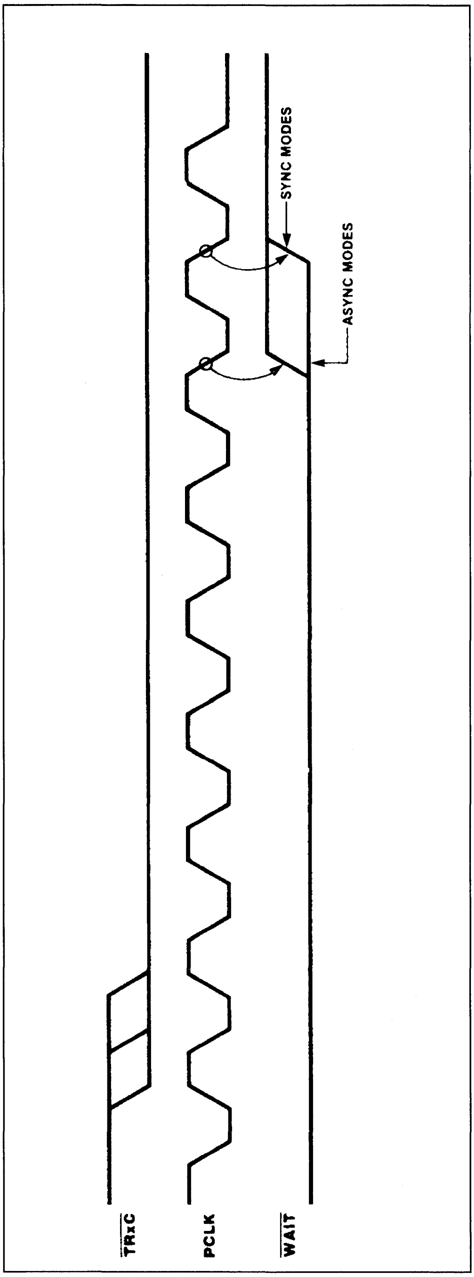
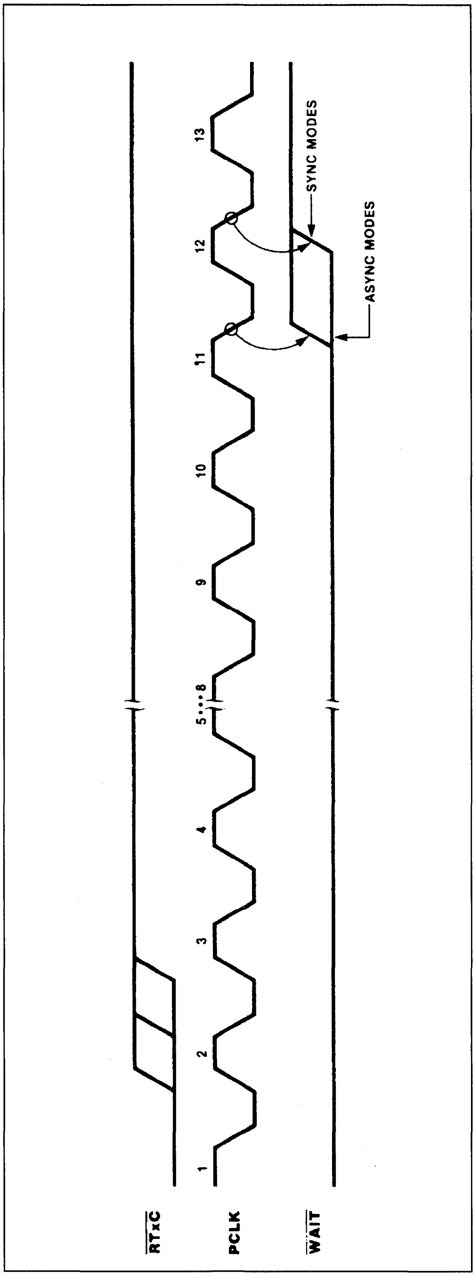


Figure 19 : Wait on Receive.



I/O PROGRAMMING CAPABILITIES (cont'd)

Figure 20 : Transmit Request Assertion.

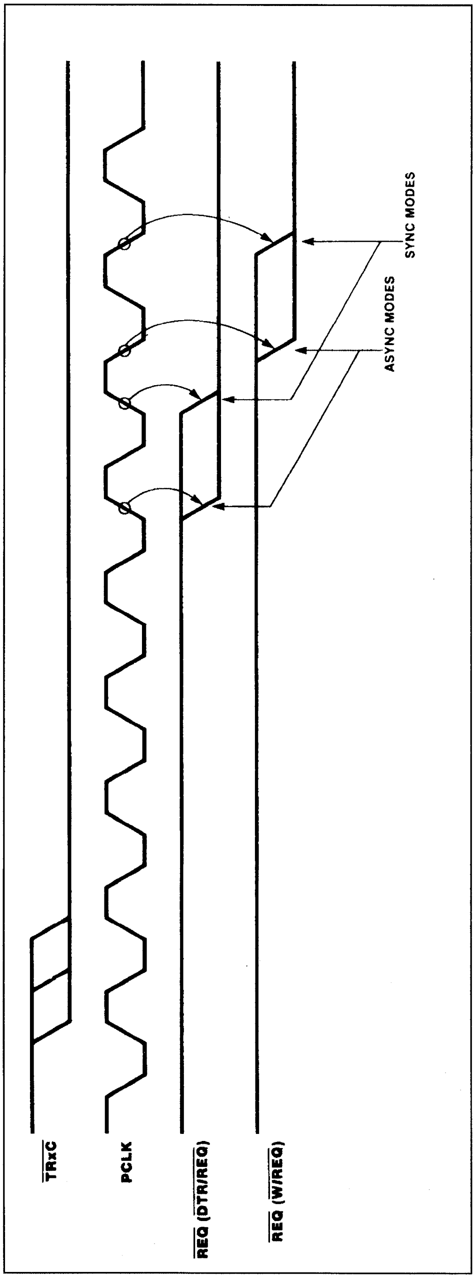
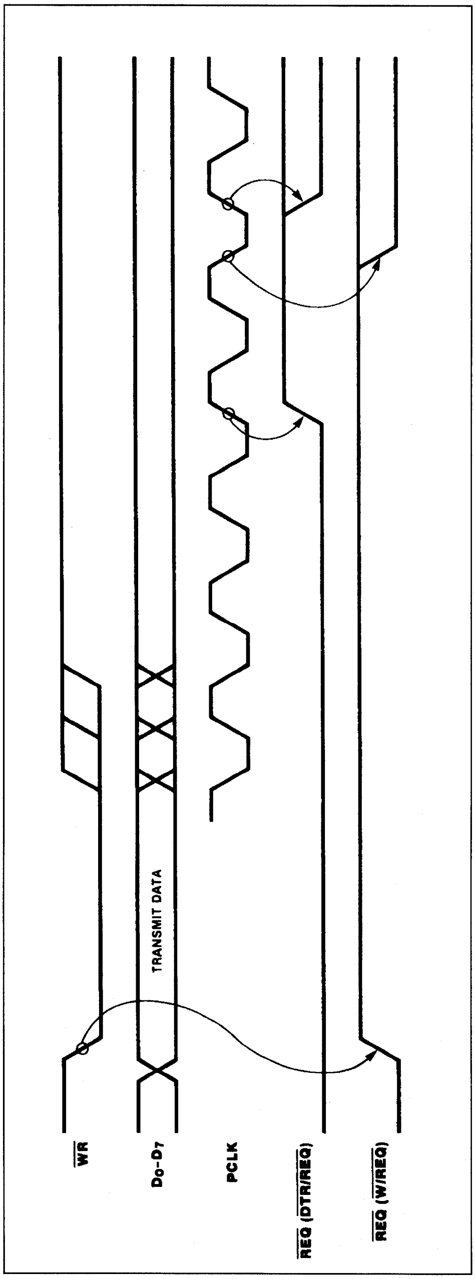

SYNC MODES
ASYNC MODES
$\overline{TRxC}$
PCLK
$\overline{REQ}$ ($\overline{DTR}$/REQ)
$\overline{REQ}$ ($\overline{W}$/REQ)

Figure 21 : Transmit Request Release.


TRANSMIT DATA
$\overline{WR}$
D_0-D_7
PCLK
$\overline{REQ}$ ($\overline{DTR}$/REQ)
$\overline{REQ}$ ($\overline{W}$/REQ)

SGS-THOMSON MICROELECTRONICS

I/O PROGRAMMING CAPABILITIES (cont'd)

Figure 22 : Receive Request Assertion.

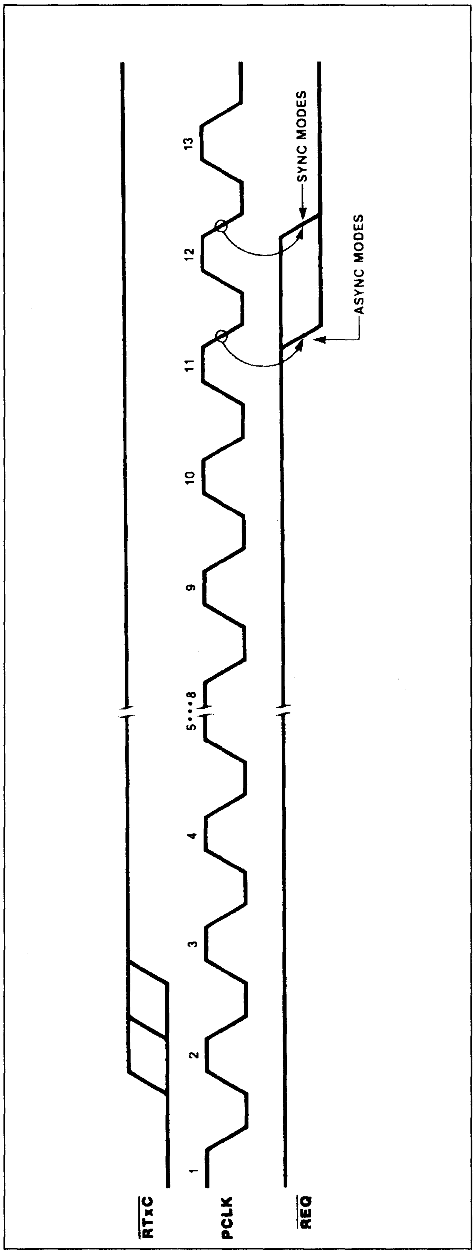
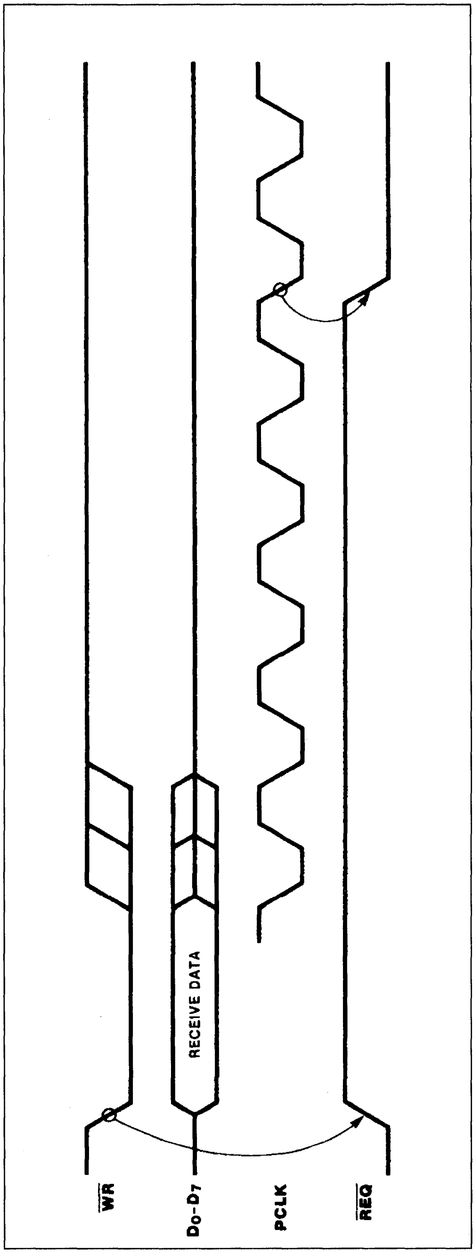


Figure 23 : Z8530 Receive Request Release.



PROGRAMMING DATA COMMUNICATION MODES

The SCC provides two independent full-duplex channels programmable for use in any common asynchronous or synchronous data communication protocol. These include asynchronous, synchronous byte-oriented protocols, monosync, IBM Bisync, and bit-oriented protocols such as HDLC and SDLC. This chapter is divided into 3 parts : Asynchronous, Synchronous, and SDLC.

Asynchronous Mode

The SCC supports Asynchronous mode with a number of programmable options including the number of bits per character, the number of stop bits, the clock factor, modem interface signals and break detect and generation. Asynchronous mode is selected by programming the desired number of stop bits in D_3 and D_2 of WR4. Programming these two bits with other than "00" places both the receiver and transmitter in Asynchronous mode. In this mode, the SCC ignores the state of bits D_4, D_3, D_2, and D_1 of WR3, bits D_5 and D_4 of WR4, bits D_2 and D_0 of WR5, all of WR6 and WR7 and all of WR10 except D_6 and D_5. Bits that are ignored may be programmed with "1" or "0" or not at all.

Asynchronous Receive. Asynchronous mode is selected by specifying the number of stop bits per character in WR4. This selection applies only the transmitter, however, as the receiver always checks for one stop bit. If after character assembly the receiver finds this stop bit to be a "0", the Framing Error bit in the receive error FIFO is set at the same time that the character is transferred to the receive data FIFO. This error bit accompanies the data to the top of the FIFO, where it generates a special receive condition. The Framing Error bit is not latched, and so must be read in RR1 before the accompanying data is read.

The number of bits per character is controlled by bits D7 and D6 of WR3. Five, six, seven, or eight bits per character may be selected via these two bits. Data is right-justifed with the unused bits set to "1s". An additional bit, carrying parity information may be selected by setting bit D0 of WR4 to "1". Note that this also enables parity for the transmitter. The parity sense is selected by bit D1 of WR4. If this bit is set to "1", the received character is checked for even parity, if set to "0", the received character is checked for odd parity. The additional parity bit per character is transferred to the receive data FIFO along with the data if the data plus parity is eight bits or less. The parity Error bit in the receive error FIFO may be programmed to cause a special receive condition interrupt by setting bit D2 of WR1 to "1". This error bit is latched and so will remain active, once set, until an Error Reset command has been issued. If interrupts are not used to transfer data, the Parity Error, Framing Error, and Overrun Error bits in RR1 should be checked before the data is removed from the receive data FIFO.

The break condition is continuous "0s", as opposed to the usual continuous ones during an idle. The SCC recognizes the Break condition upon seeing a null character (all "0s") plus a framing error. Upon recognizing this sequence the Break bit in RR0 will be set and will remain set until a "1" is received. At this point the break condition is no longer present. At the termination of a break the receive data FIFO contains a single null character, which should be read and discarded. The Framing Error bit will not be set for this character, but if odd parity has been selected, the Parity Error bit will be set. Caution should be exercised if the receive data line contains a switch that is not debounced to generate breaks. Switch bounce may cause multiple breaks, recognized by the SCC to be additional characters assembled in the receive data FIFO. It may also cause a receive overrun condition being latched.

The SCC may be programmed to accept a receive clock that is one, sixteen, thirty-two, or sixty-four times the data rate. This is selected by bits D7 and D6 in WR4. The 1X mode is used when bits are synchronized external to the receiver. The 1X mode is the only mode in which a data encoding method other than NRZ may be used. The clock factor is common to the receiver and transmitter.

The SCC provides up to three modem control signals associated with the receiver. The $\overline{\text{SYNC}}$ pin is a general-purpose input whose state is reported in the Sync/Hunt bit in RR0. If the crystal oscillator is enabled, this pin is not available and the Sync/Hunt bit is forced to "0". Otherwise, the SYNC pin may be used to carry the Ring Indicator signal. The $\overline{\text{DTR}}$/ $\overline{\text{REQ}}$ pin carries the inverted state of the DTR bit (D7) in WR5 unless this pin has been programmed to carry a DMA Request signal. The $\overline{\text{DCD}}$ pin is ordinarily a simple input to the $\overline{\text{DCD}}$ bit in RR0. However, if the Auto Enables mode is selected by setting D5 of WR3 to "1", this pin becomes an enable for the receiver. That is, if Auto Enables is on and the $\overline{\text{DCD}}$ pin is HIGH, the receiver is disabled. While the $\overline{\text{DCD}}$ pin is LOW, the receiver is enabled.

The initialization sequence for the receiver in Asynchronous mode is : WR4 first to select the mode,then WR3 and WR5 to select the various options. At this point, the other registers should be ini-

PROGRAMMING DATA COMMUNICATION MODES (cont'd)

tialized as necessary. When all of this is complete the receiver may be enabled by setting bit D_0 or WR3 to "1".

Asynchronous Transmit. Asynchronous mode is selected by specifying the number of stop bits per character in bits D_3 and D_2 of WR4. The three options available are one, one-and-a-half, or two stop bits per character. These two bits only select the number of stop bits for the transmitter, as the receiver always checks for one stop bit.

The number of bits per transmitted character is controlled both by Bits D_6 and D_5 in WR5 and the way the data is formatted within the transmit buffer. The bits in WR5 allow the option of five, six seven, or eight bits per character. When five bits per character is selected the data may be formatted before being written to the transmit buffer to allow transmission of from one to five bits per character.

This formatting is shown in Table 5.

Table 5 : Data Format - Five Bits or Less.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	1	1	0	0	0	0	One data bit
1	1	1	0	0	0	0	0	Two data bits
1	1	0	0	0	0	0	0	Three data bits
1	0	0	0	0	0	0	0	Four data bits
0	0	0	0	0	0	0	0	Five data bits

In all cases the data must be right-justified, with the unused bits being ignored except in the case of five bits per character. An additional bit, carrying parity information, may be automatically appended to every transmitted character by setting bit D0 of WR4 to "1". This bit is sent in addition to the number of bits specified in WR4 or by the data format. The parity sense is selected by bit D1 of WR4. If this bit is set to "1", the transmitter sends even parity, if set to "0", the parity is odd.

The transmitter may be programmed to send a Break by setting bit D4 of WR5 to "1". The transmitter will send continuous "0s" from the first transmit clock edge after this command is issued, until the first transmit clock edge after this bit is reset. The transmit clock edges referred to here are those that define transmitted bit cell boundaries.

An additional status bit for use in Asynchronous mode is available in bit D_0 or RR1. This bit, called All Sent, is set when the transmitter is completely empty and any previous data or stop bits have reached the TxD pin. The All Sent bit can be used by the processor as an indication that the transmitter may be safely disabled.

The SCC may be programmed to accept a transmit clock that is one, sixteen, thirty-two, or sixty-four time the data rate. This is selected by bits D_7 and D_6 of WR4, in common with the clock factor for the receiver. Note that the chosen clock factor may restrict the number of stop bits that may be transmitted. In particular, when the clock rate and data rate are identical, one-and-a-half stop bits are not allowed. If any length other than one stop bit is desired in the times one mode, only two stop bits may be used.

There are two modem control signals associated with the transmitter provided by the SCC. The $\overline{RTS}$ pin is a simple output that carries the inverted state of the RTS bit (D_1) in WR5, unless the Auto Enables bit (D_5) is set in WR3. When Auto Enables is set the RTS pin will immediately go LOW when the RTS bit is set. However, when the RTS bit is reset the $\overline{RTS}$ pin remains LOW until the transmitter is completely empty and the last stop bit has left the TxD pin. Thus the $\overline{RTS}$ pin may be used to disable external drivers for the transmit data. The $\overline{CTS}$ pin is ordinarily a simple input to the CTS bit in RR0. However, if Auto Enables mode is selected this pin becomes an enable for the transmitter. That is, if Auto Enables is on and the $\overline{CTS}$ pin is HIGH, the transmitter is disabled ; the transmitter is enabled while the $\overline{CTS}$ pin is LOW.

The initialization sequence for the transmitter in Asynchronous mode is : WR4 first to select the mode, then WR3 and WR5 to select the various options. At this point the other registers should be initialized as necessary. When all of this is complete, the transmitter may be enabled by setting bit D3 of WR5 to "1". Note that the transmitter and receiver may be initialized at the same time.

Synchronous Mode

In synchronous modes of operation a special bit pattern is used to provide character synchronization. The SCC offers several options to support Synchronous mode including various sync character lengths, the number of bits per data character, parity generation and checking, CRC generation and checking, as well as modem controls and a transmitter to receiver synchronization function. Synchronous mode is selected by programming bits D_3 and D_2 of WR4 with "Os". This selects Synchronous mode, as opposed to Asynchronous mode, but this selection is further modified by bits D_5 andD_7 of WR4 as well as bits D_1 and D_0 of WR10.

PROGRAMMING DATA COMMUNICATION MODES (cont'd)

The sync character or characters are written in WR6 and WR7. In all synchronous modes, except External Sync the state of bits D_7 and D_6 of WR4 are forced to "0" to select the times one clock mode. In External Sync mode these two bits must be programmed with "0s".

Synchronous Receive. The receiver in the SCC searches for character synchronization only while it is in Hunt mode. In this mode the receiver is idle except that it is searching the incoming data stream for a sync character match. The receiver is in Hunt mode when it is first enabled, and may be placed in Hunt mode by command from the processor. This is accomplished by issuing the Enter Hunt Mode command in WR3. This bit (D_4) is a command ; writing a "0" to it has no effect. The Hunt status of the receiver is reported by the Sync/ Hunt bit in RR0. Sync/Hunt is one of the possible sources of external/status interrupts, with both transitions causing an interrupt. This is true even if the Sync/ Hunt bit is set as a result of the processor issuing the Enter Hunt Mode command.

An 8-bit sync character is selected by setting bits D_5 and D_4 of WR4, as well as bit D0 of WR10, to "0". With this option the receiver searches the data stream for a match will the eight bits in WR7. The 6-bit sync option requires the same programming except that D0 of WR10 is set to "1" and the sync character is held in the high-order six bits of WR7. The SCC also allows the option of double length sync characters. This is selected by setting bit D_5 of WR4 to "0" and bit D_4 of WR4 to "1". The selection between 12 and 16 bits of sync character is controlled by bit D0 of WR10. A "0" selects 16 bits of sync character, while a "1" in this bit selects a 12-bit sync character. The arrangement of the sync character in WR6 and WR7 is shown in Figure 24. For those applications requiring any other sync character length, the SCC makes provision for an external circuit to provide a character synchronization signal on the $\overline{\text{SYNC}}$ pin. This mode is selected by setting bit D_5 and D_4 of WR4 to "1". In this mode the Sync/Hunt bit in RR0 reports the state of the $\overline{\text{SYNC}}$ pin but the receiver must still be placed in Hunt mode when the external logic is searching for a sync character match. When the receiver is in Hunt mode and the $\overline{\text{SYNC}}$ pin is driven LOW, two receive clock cycles after the last bit of the sync character is received, character assembly will begin on the rising edge of the receive clock immediately preceding the activation of $\overline{\text{SYNC}}$. This is shown in Figure 25. The receiver leaves Hunt mode when $\overline{\text{SYNC}}$ is driven LOW. In all cases except External Sync mode the $\overline{\text{SYNC}}$ pin is an output that is driven LOW by the SCC to signal that a sync character has been received. The $\overline{\text{SYNC}}$ pin is activated regardless of character boundaries so any external circuitry using it should only respond the $\overline{\text{SYNC}}$ pulse that occurs while the receiver is in Hunt mode. The timing for the $\overline{\text{SYNC}}$ signal is shown in Figure 26.

The number of bits per character is controlled by bits D_7 and D_6 of WR3. Five, six, seven, or eight bits per character may be selected via these two bits. The data is right-justified in the receive data buffer. The SCC merely takes a snapshot of the receive data stream at the appropriate times so the "unused" bits in the receive buffer are only the bits following the character in the data stream. An additional bit, carrying parity information, may be selected by setting bit D_0 of WR4 to "1". If this bit is set to "1", the received character is checked for even parity, if set to "0", the received character is checked for odd parity. The additional bit per character is visible in the receive data FIFO if the data plus parity is eight bits or less. The parity bit is not visible when there are eight data bits per character. The Parity Error bit in the receive error FIFO may be programmed to cause a Special Receive Condition interrupt by setting bit D_2 of WR1 to "1". This error bit is latched and so will remain active, once set, until an Error Reset command has been issued. If interrupts are not used to transfer data the Parity Error, CRC Error, and Overrun Error bits in RR1 should be checked before the data is removed from the receive data FIFO. The character lenght may be changed at any time before the new number of bits has been assembled by the receiver, but, care should be exercised as unexpected results may occur. A representative example, switching from five bits to eight bits and back to five bits is shown in Figure 27. It is sometimes desirable to prevent sync characters in the receive data stream from being transferred to the receive data FIFO. This function is available in the SCC by setting the Sync Character Load Inhibit bit (D1) in WR3 to "1". While this bit is set to "1", character about to be loaded into the receive data FIFO is compared with the contents of WR6. If all eight bits match the character, it is not loaded into the receive data FIFO. Because the comparison is across eight bits, this function works correctly only when the number of bits per character is the same as the sync character length. Thus it cannot be used with 12- or 16-bit sync characters. Both leading sync characters and sync characters embedded in the data will be properly removed in the case of an 8-bit sync character, but only the leading sync characters may be properly removed in the case of a 6-bit sync character. Care must be exercised in using this feature be-

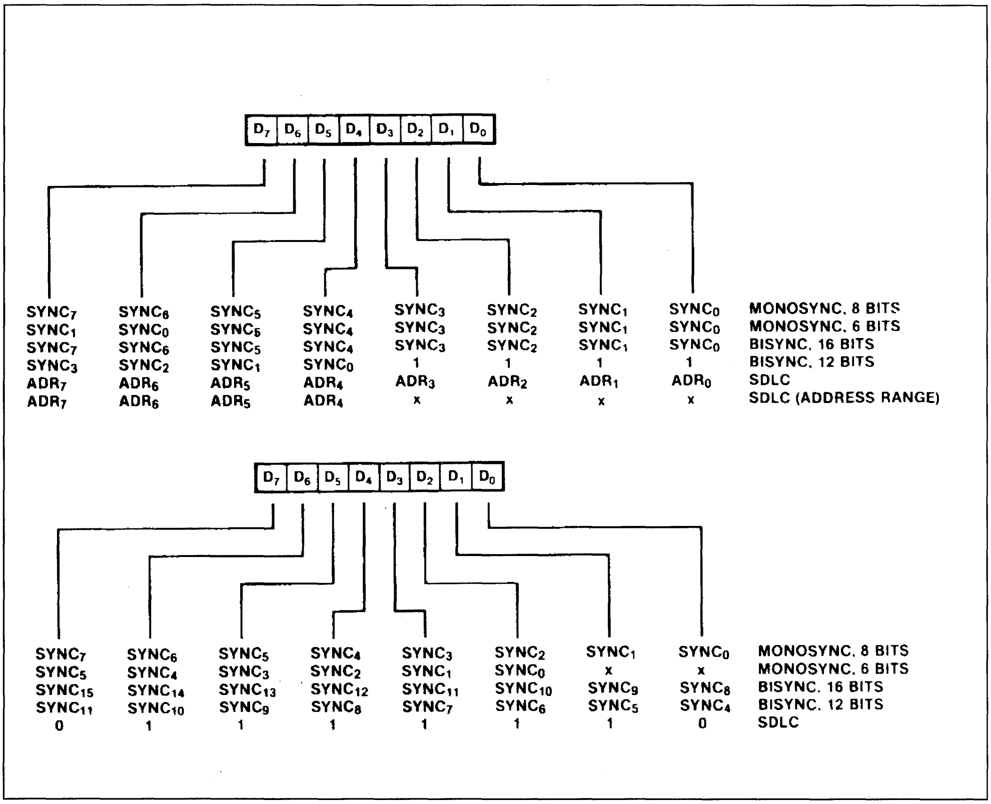
PROGRAMMING DATA COMMUNICATION MODES (cont'd)

cause sync characters not transferred to the receive data FIFO will automatically be excluded from CRC calculation. This works properly only in the 8-bit case.

Either of two CRC polynomials may be used in synchronous modes, selected by bit D_2 in WR5. If this bit is set to "1", the CRC-16 polynomial is used, if this bit is set to "0", the CRC-CCITT polynomial is used. This bit controls the polynomial selection for both the receiver and transmitter. The initial state of the generator and checker is controlled by bit D_7 of WR10. When this bit is set to "1", both the generator and checker will have an initial value of all ones, if this bit is set to "0", the initial values will be all "0s". The SCC presets the checker whenever the receiver is in Hunt mode so a CRC reset command is not strictly necessary. However, the CRC checker may be preset by issuing the Reset CRC Checker command in WR0. This command is encoded in bits D_7 and D_6 of WR0. If CRC is to be used the CRC checker must be enabled by setting bit D0 of WR3 to "1". If sync characters are being stripped from the data stream, this may be done at any time before the first non-sync character is received. If the sync strip feature is not being used, CRC must not be enabled until after the first data character has been transferred to the receive data FIFO. As previously mentioned, 8-bit sync characters stripped from the data stream are automatically excluded from CRC calculation.

Some synchronous protocols require that certain characters be excluded from CRC calculation. This is possible in the SCC because CRC calculation may be enabled and disabled on the fly. To give the processor sufficient time to decide whether or not a particular character should be included in the CRC calculation, the SCC contains an 8-bit time delay between the receive shift register and the CRC

Figure 24 : Sync Character Programming.



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
$SYNC_7$	$SYNC_6$	$SYNC_5$	$SYNC_4$	$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	MONOSYNC, 8 BITS
$SYNC_1$	$SYNC_0$	$SYNC_5$	$SYNC_4$	$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	MONOSYNC, 6 BITS
$SYNC_7$	$SYNC_6$	$SYNC_5$	$SYNC_4$	$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	BISYNC, 16 BITS
$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	1	1	1	1	BISYNC, 12 BITS
ADR_7	ADR_6	ADR_5	ADR_4	ADR_3	ADR_2	ADR_1	ADR_0	SDLC
ADR_7	ADR_6	ADR_5	ADR_4	x	x	x	x	SDLC (ADDRESS RANGE)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
$SYNC_7$	$SYNC_6$	$SYNC_5$	$SYNC_4$	$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	MONOSYNC, 8 BITS
$SYNC_5$	$SYNC_4$	$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	x	x	MONOSYNC, 6 BITS
$SYNC_{15}$	$SYNC_{14}$	$SYNC_{13}$	$SYNC_{12}$	$SYNC_{11}$	$SYNC_{10}$	$SYNC_9$	$SYNC_8$	BISYNC, 16 BITS
$SYNC_{11}$	$SYNC_{10}$	$SYNC_9$	$SYNC_8$	$SYNC_7$	$SYNC_6$	$SYNC_5$	$SYNC_4$	BISYNC, 12 BITS
0	1	1	1	1	1	1	0	SDLC

PROGRAMMING DATA COMMUNICATION MODES (cont'd)

Figure 25 : SYNC as an Output.

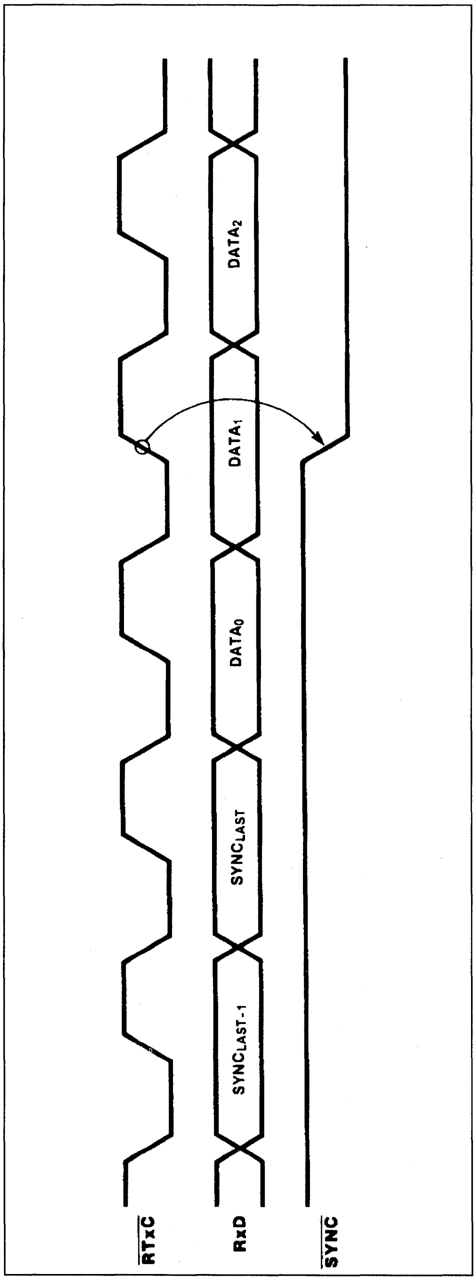
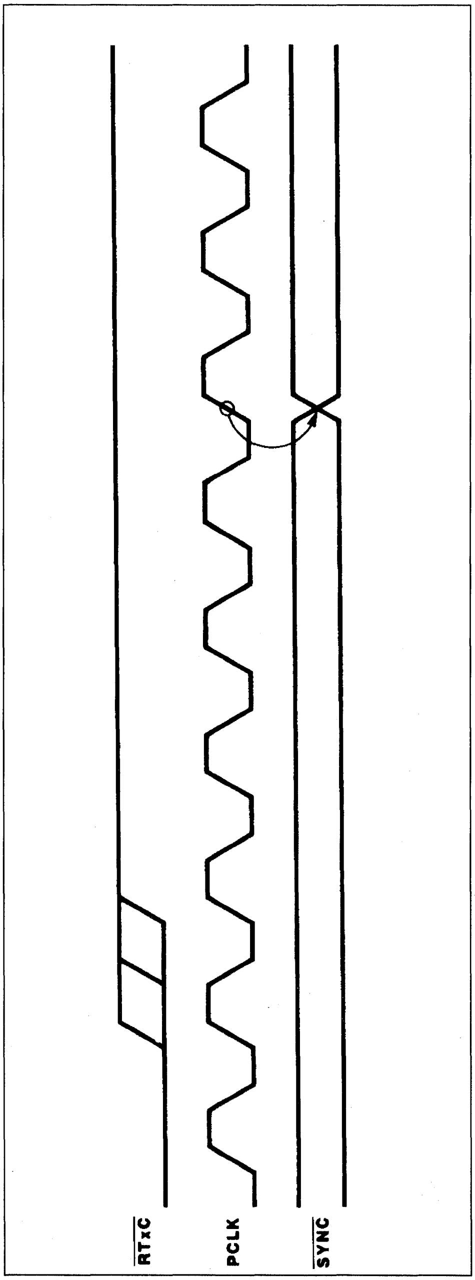


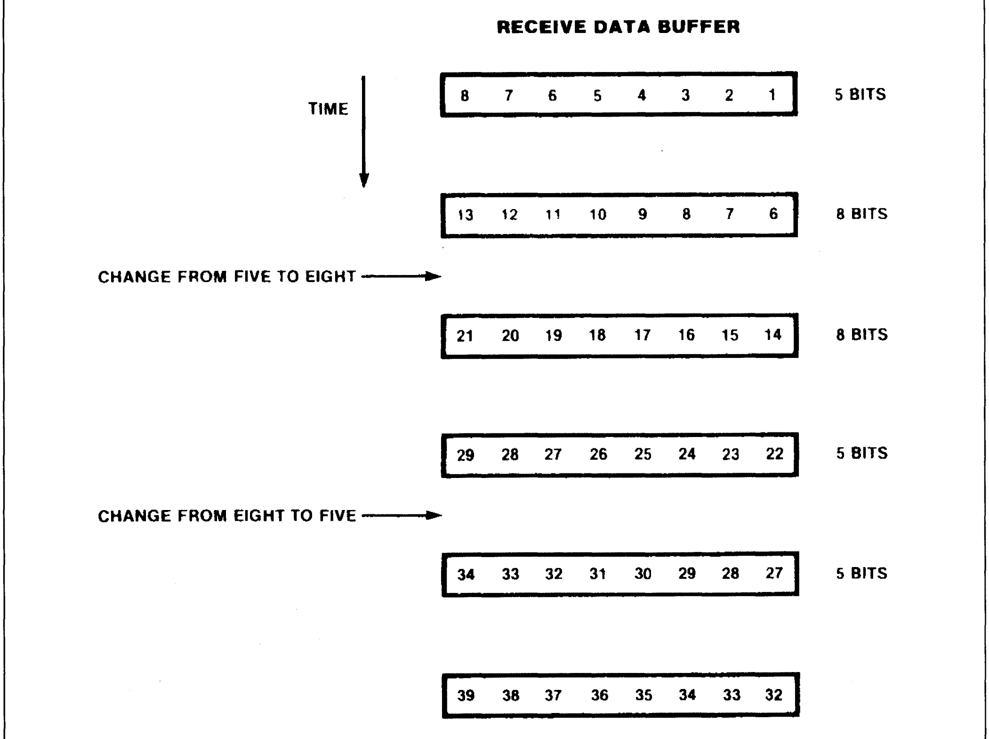
Figure 26 : SYNC as an Input.



PROGRAMMING DATA COMMUNICATION MODES (cont'd)

checker. The logic also guarantees that the calculation will only start or stop on a character boundary by delaying the enable or disable until the next character is loaded into the receive data FIFO. To understand how this works refer to Figure 28 and the following explanation. Consider a case where the SCC receives a sequence of eight bytes, called A, B, C, D, E, F, G and H with A receiver first. Now suppose that A is the sync character, that CRC is to be calculated on B, C, E, and F, and that F is the last byte of this message. Before A is received the receiver is in Hunt mode and the CRC is disabled. When A is in the receive shift register it is compared with the contents of WR7. Since A is the sync character, the bit patterns match and receiver leaves Hunt mode, but character A is not transferred to the receive data FIFO. The CRC remains disabled even though somewhere during the next eight-bit-time processor reads B and enables CRC. At the end of an eight-bit-time, B is in the 8-bit delay and C is in the receive shift register. At this point, B is loaded into the receive data FIFO. The CRC remains disabled even though somewhere during the next eight bit times the processor reads B and enables CRC. At the end of the eight-bit-time, B is in the 8-bit delay and C is in the receive shift register. Character C is loaded into the receive data FIFO and at the same time the CRC checker is enabled. During the next eight-bit-time, the processor reads C and leaves the CRC enabled. At the end of these eight-bit-times the SCC has calculated CRC on B, character C is the 8-bit delay and D is in the Receive Shift register. D is then loaded into the receive data buffer and at some point during the next eight-bit-time the processor reads D and disables CRC. At the end of these eight-bit-times CRC has been calculated on C, character D is in the 8-bit delay and E is in the Receive Shift register.

Figure 27 : Changing Character Length.



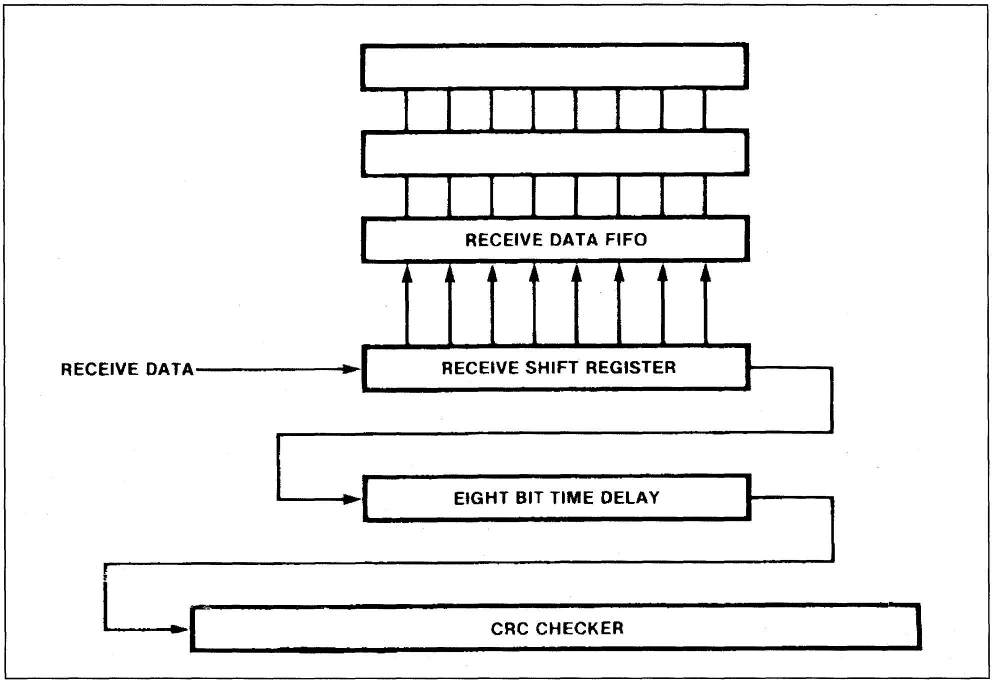
PROGRAMMING DATA COMMUNICATION MODES (cont'd)

Now E is loaded into the receiver data FIFO and, at the same time, the CRC is disabled. During the next eight-bit-times the processor reads E and enables the CRC. During this time E shifts into the 8-bit delay, F enters the Receive Shift register and CRC is not being calculated on D. After these eight-bit-times have elapsed, E is in the 8-bit delay, and F is in the Receive Shift register. Now F is transferred to the receive data FIFO and CRC is enabled. During the next eight-bit-times the processor reads F and leaves the CRC enabled. The processor is usually aware that this is the last character in the message and so prepares to check the result of the CRC computation. However, another sixteen bit-times are required before CRC has been calculated on all of character F. At the end of eight-bit-times F is in the 8-bit delay and G is in the Receive Shift register. At this time G is transferred to the receive data FIFO. Character G must be read and discarded by the processor. Eight bit times later H is transferred to the receive data FIFO also. The result of a CRC calculation is latched in the receive error FIFO at the same time as data is written to the receive data FIFO. Thus the CRC result through character F accompanies character H in the FIFO and will be valid in RR1 until character H is read from the receive data FIFO. The CRC checker may be disabled and reset at any time after character H is transferred to the receive data FIFO. Recall, however, that internally CRC will not be disabled until a character is loaded into the receive data FIFO so the reset command should not be issued until after this occurs. A better alternative is to place the receiver in Hunt mode, which automatically disables and resets the CRC checker.

Up to two modem control signals associated with the receiver are available is synchronous modes. The $\overline{\text{DTR}}/\overline{\text{REQ}}$ pin carries the inverted state of the DTR bit (D7) in WR5 unless this pin has been programmed to carry a DMA Request signal. The $\overline{\text{DCD}}$ pin is ordinarily a simple input to the DCD bit in RR0. However, if the Auto Enables mode is selected by setting D5 of WR3 to "1", this pin becomes an enable for the receiver. That is, if Auto Enables is on and the $\overline{\text{DCD}}$ pin is HIGH the receiver is disabled ; while the $\overline{\text{DCD}}$ pin is LOW the receiver is enabled.

The initialization sequence for the receiver in synchronous modes is WR4 first, to select the mode,

Figure 28 : Receive CRC Data Path.



PROGRAMMING DATA COMMUNICATION MODES (cont'd)

then WR10 to modify it if necessary, WR6 and WR7 to program the sync characters and then WR3 and WR5 to select the various options. At this point the other registers should be initialized as necessary. When all of this is complete the receiver is enabled by setting bit D of WR3 to "1".

Synchronous Transmit. Once Synchronous mode has been selected, any of three sync character lengths may be selected. An 8-bit sync character is selected by setting bits D_5 and D_4 or WR4, as well as bit D0 of WR10 to "0". With this option selected the transmitter sends the contents of WR6 when it has no data to send. The 6-bit sync option requires the same programming except that bit D0 of WR10 is set to "1" and only the least significant six bits of WR6 and used as a time fill. For a 16-bit sync character, set bit D_4 of WR4 to "1" and D_5 of WR4 and bit D_0 of WR10 to "0". In this mode the transmitter sends the concatenation of WR6 and WR7 as a time fill. Because the receiver requires that sync characters be left-justified in the registers, while the transmitter requires them to be right-justified, only the receiver will work with a 12-bit sync character. While the receiver is in External Sync mode the transmitter sync length may be six or eight bits, as selected by bit D0 of WR10.

The number of bits per transmitted character is controlled by bits D_6 and D_5 of WR5 and the way the data is formatted within the transmit buffer. The bits in WR5 allow the option of five, six, seven, or eight bits per character. When five bits per character is selected the data may be formatted before being written to the transmit buffer to allow transmisssion of from one to five bits per character. This formatting is shown in Table 5. In all cases the data must be right-justified, with the unused bits being ignored except in the case of five bits per character. An additional bit, carrying parity information, may be automatically appended to every transmitted character by setting bit D_0 of WR4 to "1". This parity bit is sent in addition to the number of bits specified in WR4 or by the data format. If this bit is set to "1", the transmitter will send even parity, if set to "0", the transmitted parity will be odd.

Either of two CRC polynomials may be used in synchronous modes, selected by bit D_2 in WR5. If this bit is set to "1", the CRC-16 polynomial is used and, if this bit is set to "0", the CRC-CCITT polynomial is used. This bit controls the selection for both the transmitter and receiver. The initial state of the generator and checker is controlled by bit D_7 of WR10. When this bit is set to "1", both the generator and checker will have an initial value of all ones, if this bit is set to "0", the initial values will be all zeros. The SCC does not automatically preset the CRC generator, so this must be done in software. This is accomplished by issuing the Reset Tx CRC Generator command, which is encoded in bits D_7 and D_6 of WR0. For proper results this command must be issued while the transmitter is enabled and sending sync characters. If CRC is to be used, the transmit CRC generator must be enabled by setting bit D0 of WR5 to "1". This bit may also be used to exclude certain characters from the CRC calculation. Sync characters are automatically excluded from the CRC calculation and any characters written as data may also be excluded from the calculation by using bit D_0 of WR5. Internally, the CRC is enabled or disabled for a particular character at the same time as the character is loaded from the transmit buffer to the Transmit Shift register. Thus, to exclude a character from CRC calculation bit, D_0 of WR5 should be set to "0" before the character is written to the transmit buffer. This guarantees that the internal disable will occur when the character moves from the buffer to the shift register. Once the buffer becomes empty, the Tx CRC Enable bit may be written for the next character.

Enabling the CRC generator is not sufficient to control the transmission of CRC. In the SCC this function is controlled by the Tx Underrun/EOM bit, which may be reset by the processor and set by the SCC. When the transmitter underruns (both the transmit buffer and Transmit Shift register are empty) the state of the Tx Underrun/EOM bit determines the action taken by the SCC. If the tx Underrun/EOM bit is set when the underrun occurs, the transmitter will send sync characters, if this bit is reset when the underrun occurs, the transmitter will send the accumulated CRC followed by sync characters. When the CRC is loaded into the transmit Shift register for transmission, the SCC will set the Tx Underrun/EOM bit to indicate this. This transition may be programmed to cause an external/status interrupt, or the Tx Underrun/EOM is available in RR0. The Reset Tx Underrun/EOM Latch command is encoded in bits D_7 and D_6 in WR0. For correct transmission of the CRC at the end of a block of data, this command must be issued after the first character is written to the SCC but before the transmitter underruns after the last character written to the SCC. The command is usually issued immediately after the first character is written to the SCC so that CRC will be sent if an underrun occurs inadvertently during the block of data.

In synchronous modes, if the transmitter is disabled during transmission of a character, that character

PROGRAMMING DATA COMMUNICATION MODES (cont'd)

will be sent completely. This applies to both data and sync characters. However, if the transmitter is disabled during the transmission of CRC, the 16-bit transmission will be completed, but the remaining bits will come from the SYNC registers rather than the remainder of the CRC.

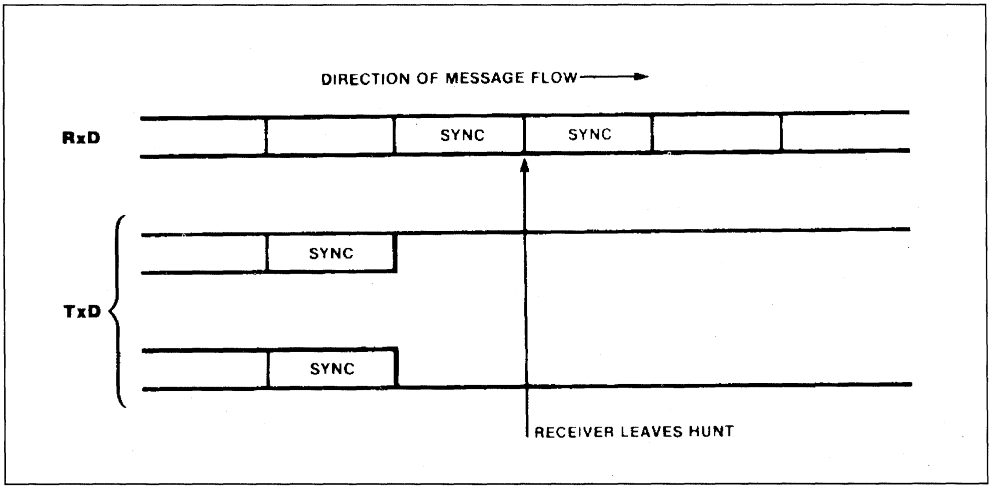
There are two modem control signals associated with the transmitter provided by the SCC. The $\overline{RTS}$ pin is a simple output that carries the inverted state of the RTS bit (D_1) in WR5. The $\overline{CTS}$ pin is ordinarily a simple input to the CTS bit in RR0. However, if Auto Enables mode is selected this pin becomes an enable for the transmitter. That is, if Auto Enables is on and the $\overline{CTS}$ pin is HIGH the transmitter is disabled. While the $\overline{CTS}$ pin is LOW, transmitter is enabled.

The initialization sequence for the transmitter in synchronous modes is : WR4 first, to select the mode, then WR10 to modify it if necessary, WR6 and WR7 to program the sync characters, then WR3 and WR5 to program the sync characters, and then WR3 and WR5 to select the various options. At this point, the other registers should be initialized as necessary. When all of this is complete the transmitter may be enabled by setting bit D_3 or WR5 to "1". Now that the transmitter is enabled the CRC generator may be initialized by issuing the Reset Tx CRC Generator command in WR0.

Transmitter To Receiver Synchronization. The SCC contains a transmitter-to-receiver synchronization function that may be used to guarantee that the character boundaries for the received and transmitted data are the same. In this mode the receiver is in Hunt and the transmitter is idle, sending either all "1s" or "0s". When the receiver recognizes a sync character, it leaves Hunt mode and one character time later the transmitter is enabled and begins sending sync characters. Beyond this point the receiver and transmitter are again completely independent, except that the character boudaries are now aligned. This is shown in Figure 29. There are several restrictions on the use of this feature in the SCC. First, it will only work with 6-bit, 8-bit or 16-bit sync characters, and the data character length for both the receiver and the transmitter must be six bits with a 6-bit sync character or eight bits with an 8-bit or 16-bit sync character. Of course, the receive and transmit clocks must have the same rate as well as the proper phase relation-ship.

A specific sequence of operations must be followed to synchronize the transmitter to the receiver. Both the receiver and transmitter must have been initialized for operation in Synchronous mode sometime in the past, although this initialization need not be redone each time the transmitter is synchronized to the receiver. The transmitter is disabled by setting bit D_3 of WR5 to "0". At this point the transmitter will send continuous "1s". If it is desired that continuous "1s". If it is desired that continuous "0s" be transmitted, the Send Break bit (D_4) in WR5 should be set to "1". The transmitter is now idling but must still be placed in the transmitter to receiver synchronization mode. This is accomplished by setting the Loop

Figure 29 : Transmitter to Receiver Synchronization.



PROGRAMMING DATA COMMUNICATION MODES (cont'd)

the processor should set the Go Active on Poll bit (D_4) in WR10. The final step is to force the receiver to search for sync characters. If the receiver is currently disabled the receiver will enter Hunt mode when it is enabled by setting bit D_0 of WR3 to "1". If the receiver is already enabled it may be placed in Hunt mode by setting bit D_4 of WR3 to "1". Once the receiver leaves Hunt mode the transmitter is activate on the following character boundary.

SDLC Mode

SDLC mode is useful in bit-oriented protocols. That is, protocols which use the technique of "0" insertion to achieve data transparency. In SDLC mode, frames of information are opened and closed by a unique bit pattern called a flag. The Flag character has a bit pattern of "01111110" and this sequence is unique because all data between the opening and closing flags is prohibited from having more than five consecutive "1s". The transmitter guarantees this by watching the transmit data stream and inserting a "0" after five consecutive ones, irrespective of character boundaries. In turn, the receiver searches the receive data stream for five consecutive "1s" and deletes the next bit if it is a "0". CRC may be used in SDLC mode but only with the CRC-CCITT polynomial, because the transmitter in the SCC automatically inverts the CRC before transmission, and the receiver - to compensate for this - checks the CRC result for the bit pattern "0001110100001111". This is consistent with bit-oriented protocols such as SDLC, HDLC, and ADCCP. There are two unique bit patterns in SDLC mode besides the flag sequence. They are the Abort and EOP (End of Poll) sequence. An Abort is a sequence of from seven to thirteen consecutive "1s" and is used to signal the premature termination of a frame. The EOP is the bit pattern "11111110", which is used in loop applications as a signal to a secondary station that it may begin transmission.

SDLC mode is selected by setting bit D_5 of WR4 to "1" and bits D_4, D_3, and D_2 of WR4 to "0". In addition, the flag sequence must be written to WR7. Additional control bits for SDLC mode are located in WR10.

SDLC Receive. The receiver in the SCC always searches the receive data stream for flag characters in SDLC mode. Ordinarily, the receiver transfers all received data between flags to the receive data FIFO. However, if the receiver is in Hunt mode no flag is received. The receiver is in Hunt mode when first enabled, or the receiver may be placed in Hunt mode by the processor issuing the Enter Hunt mode command in WR3. This bit (D_4) is a command, and writing a "0" to it has no effect. The Hunt status of the receiver is reported by the Sync/Hunt bit in RR0. Sync/Hunt is one of the possible sources of external/status interrupts, with both transitions causing an interrupt. This is true even if the Sync/Hunt bit is set as a result of the processor issuing the Enter Hunt mode command. The receiver will automatically enter Hunt mode if an abort is received. Because the receiver always searches the receive data stream for flags and automatically enter Hunt Mode when an abort is received, the receiver will always handle frames correctly, and the Enter Hunt Mode command should never to needed. The SCC will drive the $\overline{\text{SYNC}}$ pin LOW to signal that a flag has been recognized. The timing for the $\overline{\text{SYNC}}$ signal is shown in Figure 30.

The first byte in an SDLC frame is assumed by the SCC to be the address of the secondary station for which the frame is intended. The SCC provides several options for handling this address. If the Address Search Mode bit (D_2) in WR3 is set to "0", the address recognition logic is disabled and all received frames are transferred to the receive data FIFO. In this mode the software must perform any address recognition. If the Address Search Mode bit is

Table 6 : Residue Codes.

Residue Code			Bits in Previous Byte				Bits in Second Previous Byte				Bits in Third Previous Byte			
2	1	0	8 B/C	7 B/C	6 B/C	5 B/C	8 B/C	7 B/C	6 B/C	5 B/C	8 B/C	7 B/C	6 B/C	5 B/C
1	0	0	0	0	0	0	3	1	0	0	8	7	5	2
0	1	0	0	0	0	0	4	2	0	0	8	7	6	3
1	1	0	0	0	0	0	5	3	1	0	8	7	6	4
0	0	1	0	0	0	0	6	4	2	0	8	7	6	5
1	0	1	0	0	0	0	7	5	3	1	8	7	6	5
0	1	1	0	0	0	-	8	6	4	-	8	7	6	-
1	1	1	1	0	-	-	8	7	-	-	8	7	-	-
0	0	0	2	-	-	-	8	-	-	-	8	-	-	-

PROGRAMMING DATA COMMUNICATION MODES (cont'd)

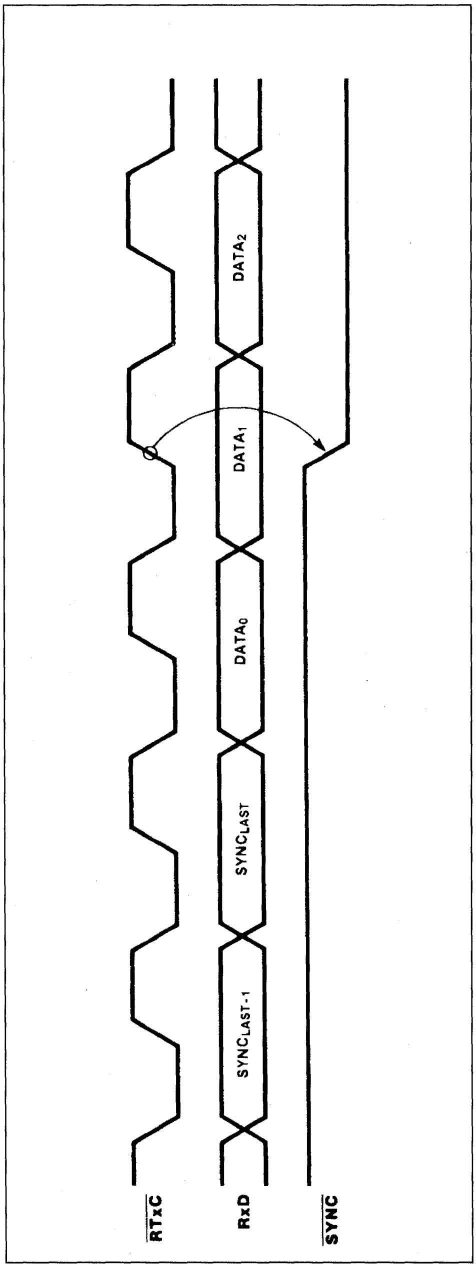
set to "1", only those frames whose address matches the address programmed in WR6 or the global address (all "1s") will be transferred to the receive data FIFO. The address comparison will be across all eight bits of WR6 if the Sync Character Load Inhibit bit (D_1) in WR3 is set to "0". The comparison may be modified so that only the four most significant bits of WR6 must match the received address. This mode is selected by setting the Sync Character Load Inhibit bit to "1". In this mode, however, the address field is still eight bits wide. The address field is transferred to the receive data FIFO in the same manner as data. It is not treated differently than data.

The number of bits per character is controlled by bits D_7 and D_6 of WR3. Five, six, seven, or eight bits per character may be selected via these two bits. The data is right-justified in the receive buffer. The SCC merely takes a snapshot of the receive data stream at the appropriate times, so the "unused" bits in the receive buffer are only the bits following the character in the data stream. An additional bit carrying parity information may be selected by setting bit D_6 of WR4 to "1". This also enables parity in the transmitter. The parity sense is selected by bit D_1 of WR4. Parity is not normally used in SDLC mode. The character length may be changed at any time before the new number of bits have been assembled by the receiver. Care should be exercised, however, as unexpected results may occur. A representative example, switching from five bits to eight bits and back to five bits is shown in Figure 31.

Most bit-oriented protocols allow an arbitrary number of bits between opening and closing Flags. The SCC allows for this by providing three bits of Residue Code in RR1 that indicate which bits in the last few bytes transferred from the receive data FIFO by the processor are actually valid data bits. The meaning of these three bits with each character length option is shown in Table 6. As indicated in the table, these bits allow the processor to determine those bits in the information (and not CRC) field. This allows transparent retransmission of the received frame. The Residue Code bits do not go through a FIFO so they change in RR1 when the last character of the frame is loaded into the receive data FIFO. If there are any characters already in the receive data FIFO the Residue Code will be updated before they are read by the processor. Thus these three bits off RR1 should be ignored by the processor unless the End of Frame bit in RR1 is set.

Only the CRC-CCITT polynomial may be used for CRC calculation in SDLC mode, although the gene-

Figure 30 : SYNC as an Output.



SGS-THOMSON MICROELECTRONICS

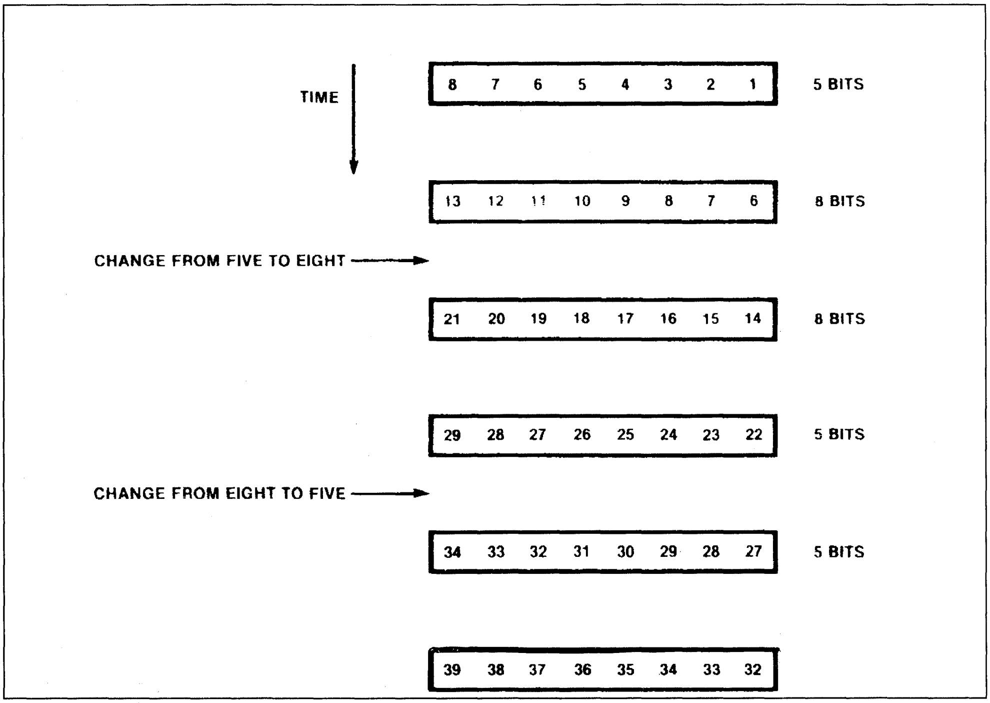
PROGRAMMING DATA COMMUNICATION MODES (cont'd)

rator and checker may be preset to all "1s" or all "0s". The CRC-CCITT polynomial is selected by setting D_2 of WR5 to "0", bit D_7 of WR10 controls the preset value. If this bit is set to "1", the generator and checker are preset to "1s", if this bit is reset, the generator and checker are preset to all "0s". The receiver expects the CRC to be inverted before transmission and so checks the CRC result against the value "0001110100001111". The SCC presets the CRC checker whenever the receiver is in Hunt mode or whenever a flag is received so a CRC reset command is not strictly necessary. However, the CRC checker may be preset by issuing the Reset CRC Checker command in WRO. The CRC checker is automatically enabled for all data between the opening and closing flags by the SCC in SDLC mode, and the Rx CRC Enable bit (D_3) in WR3 is ignored. The result of the CRC calculation for the entire frame is valid in RR1 only when accompagnied by the end of Frame bit being set in RR1. At all other times the CRC Error bit in RR1 should be ignored by the processor. Care must be exercised so that the processor does not attempt to use the CRC bytes that are transferred as data because not all of the bits are transferred properly. The last two bits of CRC are never transferred to the receive data FIFO and are not recoverable.

A frame is terminated by a closing flag. When the SCC recognizes this flag the contents of the Receive Shift register are transferred to the receive data FIFO, the Residue Code is latched, the CRC Error bit is latched in the status FIFO and the End Of Frame bit is set in the receive status FIFO. The End Of Frame bit, upon reaching the top of the FIFO, will cause a special receive condition. The processor may then read RR1 to determine the result of the CRC calculation as well as the Residue Code. If either the Rx Interrupt on Special Condition Only or the Rx Interrupt on First Character or Special Condition mode are selected, the processor must issue and Error Reset command in WR0 to unlock the receive FIFO.

In addition to searching the data stream for flags, the receiver in the SCC also watches for seven consecutive "1s", which is the abort condition. The

Figure 31 : Changing Character Length.



PROGRAMMING DATA COMMUNICATION MODES (cont'd)

presence of seven consecutive "1s" is reported in the Break/Abort bit in RR0. This is one of the possible external/status interrupts, so transitions of this status may be programmed to cause interrupts. Upon receipt of an abort the receiver is forced into Hunt mode, where it looks for flags. The Hunt status is also a possible external/status condition whose transition may be programmed to cause an interrupt. The transitions of these two bits occur very close together but either one or two external/status interrupts may result. The abort condition is terminated when a "0" is received, either by itself or as the leading "0" of a flag. The receiver does not leave Hunt mode until a flag has been received so two discrete external/status conditions will occur at the end of an abort. An abort received in the middle of a frame terminates the frame reception, but no in an orderly manner, because the character being assembled is lost.

Up to two modem control signals associated with the receiver are available in SDLC mode. The $\overline{\text{DTR}}/\overline{\text{REQ}}$ pin carries the inverted state of the DTR bit (D_7) in WR5 unless this pin has been programmed to carry a DMA Request signal. The DCD pin is ordinarily a simple input to the DCD bit in RR0. However, if the Auto Enables mode is selected by setting bit D5 of WR3 to "1", this pin becomes an enable for the receiver. That is, if Auto Enable is on and the $\overline{\text{DCD}}$ pin is HIGH the receiver is disabled. While the $\overline{\text{DCD}}$ pin is LOW, the receiver is enabled.

The initialization sequence for the receiver in SDLC mode is : WR4 first, to select the mode, then WR10 to modify it if necessary, WR6 to program the address, WR7 to program the flag and the WR3 and WR5 to select the various options. At this point the other registers should be initialized as necessary. When all of this is complete the receiver may be enabled by setting bit D_6 of WR3 to "1".

SDLC Transmit. Once SDLC mode has been selected, the flag must be written in WR7, to be used to open and close the transmitted frames. The SCC does not automatically send the address byte ; it merely encapsulates the data supplied by the processor with flags and CRC. Ordinarily, a frame will be terminated by the SCC with CRC and a flag but the SCC may be programmed to send an abort and a flag in place of the CRC. This option allows the SCC to abort a frame transmission in progress if the transmitter is accidentally allowed to underrun. This is controlled by the Abort/ $\overline{\text{Flag}}$ On Underrun bit (D_2) in WR10. When this bit is set to "1" the transmitter will send an abort and a flag in place of the CRC when an underrun occurs. The frame will be terminated normally, with CRC and a flag, if this bit is set to "0". The SCC is also able to send an abort by command of the processor. The Send Abort command, issued in WR0, will send eight consecutive "1s" and then the transmitter will idle. Since up to five consecutive "1s" may have been sent prior to the command being issued, a Send Abort will cause a sequence of from eight to thirteen "1s" to be transmitted. The Send Abort command also empties the transmit buffer register. The idle condition for the transmitter is continuous flags, but this is under program control. By setting the Mark/ $\overline{\text{Flag}}$ Idle bit (D_3) in WR10 to "1", the transmitter will send continuous "1s" in place of the idle flags. Note that the closing flag will be transmitted correctly even if this mode is selected. The Mark/ $\overline{\text{Flag}}$ Idle must be set to "0", allowing a flag to be transmitted, before data is written to the transmit buffer. Care must be exercised in doing this because the continuous "1s" are transmitted eight at a time, and all eight must leave the Transmit Shift register, so that flag may be loaded into it before the first data is written to the transmit buffer. When using the transmitter in SDLC mode, recall that all data passes through the zero inserter, which adds an extra five bit times of delay between the Transmit Shift register and the Transmit Data pin.

The number of bits per transmitted character is controlled by bits D_6 and D_5 of WR5 and the way the data is formatted within the transmit buffer. The bits in WR5 allow the option of five, six, seven, or eight bits per character. When five bits per character is selected, the data may be formatted before being written to the transmit buffer, to allow transmission of one to five bits per character. This formatting is shown in Table 4-1. In all cases the data must be right-justified, with the unused bits being ignored, except in the case of five bits per character.

An additional bit, carrying parity information, may be automatically appended to every transmitted character by setting bit D_6 of WR4 to "1". This bit is sent in addition to the number of bits specified in WR4 or by the data format. The parity sense is selected by bit D_1 of WR4. Parity is not normally used in SDLC mode. The character length may be changed on the fly, but the desired length must be selected before the character is loaded into the transmit shift register from the transmit buffer. The easiest way to ensure this is to write to WR5 to change the character length before writing the data to the transmit buffer.

Only the CRC-CCITT polynomial may be used in SDLC mode. This is selected by setting bit D_2 in WR5 to "0". This bit controls the selection for both

PROGRAMMING DATA COMMUNICATION MODES (cont'd)

the transmitter and receiver. The initial state of the generator and checker is controlled by bit D_7 of WR10. When this bit is set to "1", both the generator, and checker sill have an initial value of all "1s" and, if this bit is set to "0", the initial values will be all "0s". The SCC does not automatically preset the CRC generator so this must be done in software. This is accomplished by issuing the Reset Tx CRC generator command, which is encoded in bits D_7 and D_6 of WR0. For proper results, this command must be issued while the transmitter is enabled and idling. If CRC is to be used the transmit CRC generator must be enabled by setting bit D0 of WR5 to "1". CRC is normally calculated on all characters between opening and closing flags, so this bit is usually set to "1" at initialization and never changed.

Enabling the CRC generator is not sufficient to control the transmission of CRC. In the SCC this function is controlled by the Tx Underrun/ EOM bit, which may be reset by the processor and set by the SCC. When the transmitter underruns (both the transmit buffer and transmit shift register are empty) the state of the Tx Underrun EOM bit determines the action taken by the SCC. If the Tx Underrun/EOM bit is set to "1" when the underrun occurs, the transmitter will send flags ; if this bit is reset to "0" when the underrun occurs, the transmitter will send either the accumulated CRC followed by flags, or an abort followed by flags, depending the state of the Abort/ $\overline{\text{Flag}}$ on Underrun bit in the WR10. When the CRC or abort is loaded into the Transmit Shift register for transmission, the SCC will set the Tx Underrun/EOM bit to indicate this. This transition may be programmed to cause an external/status interrupt, or the Tx Underrun/ EOM bit is available in RR0. The Reset Tx Underrun/EOM Latch command is encoded in bits D_7 and D_6 of WR0.

For correct transmission of the CRC at the end of a frame, this command must be issued after the first character is written to the SCC but before the transmitter underruns after the last character written the SCC. The command is usually issued immediately after the first character is written to the SCC so that the abort or CRC is sent if an underrun occurs inadvertently. The Abort/ $\overline{\text{Flag}}$ on Underrun bit (D2) in WR10 is usually set to "1" at the same time as the Tx Underrun/EOM bit is reset so that an abort sill be sent if the transmitter underruns. The bit is then set to "0" near the end of the frame to allow the correct transmission of CRC.

In this paragraph the term "completely sent" means shifted out of the Transmit Shift register, not shifted out of the zero inserter, which is an additional five bit times of delay. In SDLC mode, if the transmitter is disabled during transmission of a character, that character will be "completely sent". This applies to both data and flags. However, if the transmitter is disabled during the transmission of CRC, the 16-bit transmission will be completed but the remaining bits will be from the Flag register rather than the remainder of the CRC.

There are two modem control signals associated with the transmitter provided by the SCC. The $\overline{\text{RTS}}$ pin is a simple output that carries the inverted state of the RTS bit (D_1) in WR5. The $\overline{\text{CTS}}$ pin is ordinarily a simple input to the $\overline{\text{CTS}}$ bit in RR0. However, if Auto Enables mode is selected, this pin becomes an enable for the transmitter. That is, if Auto Enables is on and the $\overline{\text{CTS}}$ pin is HIGH the transmitter is disabled. If the $\overline{\text{CTS}}$ pin is LOW, the transmitter is enabled.

The initialization sequence for the transmitter in SDLC mode is : WR4 first, to select the mode, then WR10 to modify it if necessary, WR7 to program the flag, and then WR3 and WR5 to select the various options. At this point the other registers should be initialized as necessary. When all of this is complete, the transmitter may be enabled by setting bit D_3 of WR5 to "1". Now that the transmittter is enabled, the CRC generator may be initialized by issuing the Reset Tx CRC Generator command in WR0.

SDLC Loop Mode. SDLC Loop mode is quite similar to SDLC mode except that two additional control bits are used. They are the Loop Mode bit (D_1) and the Go Active on Poll bit (D_4) in WR10. In addition to these two extra control bits, there are also two status bits in RR10. They are the On Loop bit (D_1) and the Loop Sending bit (D_4). Before Loop mode is selected both the receiver and transmitter must be completely initialized for SDLC operation. Once this is done, Loop mode is selected by setting bit D_1 of WR10 to "1". At this point the SCC connects TxD to RxD with only gate delays in the path. At the same time a flag is loaded into the Transmit Shift register, and is shifted to the end of the zero inserter, ready for transmission. The SCC will remain in this state until the Go Active on Poll bit (D4) in WR10 is set to "1". When this bit is set to "1" the receiver begins looking for a sequence of seven consecutive "1s", indicating either an EOP or an idle line. When the receiver detects this condition the Break/Abort bit in RR0 is set to "1" and a one-bit time delay is inserted in the path from RxD to TxD. The On Loop bit in RR10 is also set to "1" at this time, and the receiver enters the Hunt mode. The SCC cannot transmit on the loop until a flag is received, causing the receiver

PROGRAMMING DATA COMMUNICATION MODES (cont'd)

to leave Hunt mode, and another EOP (bit pattern " 11111110") is received. The SCC is now on the loop and capable of transmitting on the loop. As soon as this status is recognized by the processor, the Go Active On Poll bit in WR10 should be set to "0" to prevent the SCC from transmitting on the loop without the consent of the processor.

To transmit a message on the loop, the Go Active On Poll bit in WR10 must be set to "1". Once this is done, the SCC will change the next received EOP into Flag and begin transmitting on the loop. When the EOP is received, the Break/Abort and Hunt bits in RRO will be set to "1", and the Loop Sending bit in RR10 will also be set to "1". Data to be transmitted may be written after the Go Active On Poll bit has been set or after the receiver enter Hunt mode. If the data is written immediately after the Go Active On Poll bit has been set, the SCC will only insert one flag after the EOP is changed into a flag. If the data is not written until after the receiver enters the Hunt mode, flags will be transmitted until the data is written. If only one frame is to be transmitted on the loop in response to an EOP, the processor must set the Go Active on Poll bit to "0" before the last data is written to the transmitter. In this case the transmitter will close the frame with a single flag, and then revert to the one-bit delay. The Loop Sending bit in RR10 is set to "0" when the closing Flag has been sent. If more than one frame is to be transmitted, the Go Active On Poll bit should not be set to "0" until the last frame is being sent. If this bit is not set to "0" before the end of a frame, the transmitter will send Flags until either more data is written to the transmitter, or until the Go Active On Poll bit is set to "0". Note that the state of the Abort/ $\overline{\text{Flag}}$ on Underrun and Mark/ $\overline{\text{Flag}}$ Idle bits in WR10 are ignored by the SCC in SDLC Loop mode.

To go off the loop in an orderly manner requires actions similar to those taken to go the loop. First, the Go Active On Poll bit must be set to "0" and any transmission in progress completed, if the SCC is currently sending on the loop. Once the SCC is not sending on the loop, an exit from the loop is accomplished by setting the Loop Mode bit in WR10 to "0", and at the same time writing the Abort/ Flag on Underrun and Mark/$\overline{\text{Flag}}$ Idle bits with the desired values. The SCC will revert to normal SDLC operation as soon as an EOP is received, or immediately, if the receiver is already in Hunt mode because of the receipt of an EOP.

The initialization sequence for the SCC in SDLC Loop mode is similar to the sequence used in SDLC mode, except that it is somewhat longer. The processor should program WR4 first, to select SDLC mode, and then WR10 to select the CRC preset value and program the Mark/Flag Idle bit. The Loop Mode and Go Active On Poll bits in WR10 should not be set to"1" yet. The flag is written in WR7 and the various options are selected in WR3 and WR5. At this point the other registers should be initialized as necessary, then Loop Mode bit (D_1) in WR10 should be set to "1". When all of this is complete the transmitter may be enabled by setting bit D_3 of WR5 to "1". Now that the transmitter is enabled, the CRC generator may be initialized by issuing the Reset TxCRC Generator command in WR0. The receiver is enabled by setting the Go Active on Poll bit (D_4) in WR10 to "1". The SCC will go on the loop when seven consecutive "1s" are received, and will signal this by setting the On Loop bit in RR10. Note that the seven consecutive "1s" will set the Break/Abort and Hunt bits in RR0 also. Once the SCC is on the loop, the Go Active on Poll bit should be set to "0" until a message is to be transmitted on the loop. To transmit a message on the loop, the Go Active on Poll bit should be set to "1". At this point the processor may either write the first character to the transmit buffer and wait for a transmit buffer empty condition, or wait for the Break/Abort and Hunt bits to be set in RR10 and the Loop Sending bit to be set in RR10 before writing the first data to the transmitter. The Go Active On Poll bit should be set to "0" after the transmission of the frame has begun. To go off of the loop, the processor should set the Go Active On Poll bit in WR10 to "0" and then wait for the Loop Sending bit in RR10 to be set to "0". At this point the Loop Mode bit (D_1) in WR10 is set to "0" to request an orderly exit from the loop. The SCC will exit SDLC Loop mode when seven consecutive "1s" have been received ; at the same time the Break/Abort and Hunt bits in RR0 will be set to "1", and the On Loop bit in RR10 will be set to "0".

SUPPORT CIRCUITRY PROGRAMMING

The SCC incorporates additional circuitry to aid in serial communications. This circuitry includes clocking options, baud rate generator, data encoding, and internal loopback. This section discusses how to program these functions.

Clock Options

The SCC may be programmed to select one of several sources to provide the transmit and receive clocks. In addition, the SCC requires a fundamental, parallel resonant crystal oscillator in each channel, as well as the ability to echo one of several internal clock sources to the outside world. These options are controlled by the bits in WR11.

The crystal oscillator option is controlled by bit D_7 in WR11. When this is set to "0", the crystal oscillator is disabled and all pins function normally. When this bit is set to "1" the crystal oscillator is enabled and a high-gain amplifier is connected between the $\overline{\text{RTxC}}$ pin and the $\overline{\text{SYNC}}$ pin. While the crystal oscillator is enabled, anything that has $\overline{\text{RTxC}}$ selected as its clock source will automatically be connected to the output of the crystal oscillator. While the crystal oscillator is enabled, the $\overline{\text{SYNC}}$ pin is obviously unavailable for other use. In synchronous modes no sync pulse is output, and the External Sync mode cannot be selected. In asynchronous modes the state of the Sync/ Hunt bit in RR0 is no longer controlled by the $\overline{\text{SYNC}}$ pin. Instead, the Sync/ Hunt bit is forced to "0". The crystal oscillator requires some finite time to stabilize. The oscillator must be allowed to stabilize before it is used as a clock source.

The source of the receive clock is controlled by bits D_6 and D_5 of WR11. The receive clock may be programmed to come from the $\overline{\text{RTxC}}$ pin , the $\overline{\text{TRxC}}$ pin, the output of the baud rate generator, or the transmit output of the DPLL.

The source of the transmit clock is controlled by bits D_4 and D_3 of WR11. The transmit clock may be programmed to come from the $\overline{\text{RTxC}}$ pin, the $\overline{\text{TRxC}}$ pin, the output of the baud rate generator, or the transmit output of the DPLL.

Ordinarily the TRxC pin is an input, but it becomes on output if this pin has not been selected as the source for the transmitter or the receiver, and bit D_2 of WR11 is set to "1". The selection of the signal provided on the $\overline{\text{TRxC}}$ ouput pin is controlled by bits D_1 and D0 of WR11. The $\overline{\text{TRxC}}$ pin may be programmed to provide the output of the crystal oscillator, the output of the baud rate generator, the receive output of the DPLL or the actual transmit clock. If the output of the crystal oscillator is selected but the crystal oscillator has not been enabled the $\overline{\text{TRxC}}$ pin will be driven HIGH. The option of placing the transmit colck signal on the $\overline{\text{TRxC}}$ pin when it is an output allows access to the transmit output of the DPLL.

Figure 32 shows a simplified schematic diagram of the circuitry used in the clock multiplexing. It shows the inputs to the multiplexer section as well as the various signal inversions that occur in the paths to the outputs. Also shown are the edges used by the receiver, transmitter, baud rate generator and DPLL to sample or send data or otherwise change state. For example, the receiver samples data on the falling edge, but since there is an inversion in the clock path between the $\overline{\text{RTxC}}$ pin and the receiver, a rising edge of the $\overline{\text{RTxC}}$ pin samples the data for the receiver.

Selection of the clocking options may be done anywhere in the initialization sequence, but the final values must be selected before the receiver, transmitter, baud rate generator, or DPLL are enabled to prevent problems from arbitrarily narrow clock signals out of the multiplexers. The same is true of the crystal oscillator, in that the output should be allowed to stabilize before it is used as a clock source.

Baud Rate Generator

Figure 33 shows a block diagram of the baud rate generator. It consists of a 16-bit down-counter, two 8-bit time constant registers and an output divide-by-two. The baud rate generator input comes from the output of a two-input multiplexer, the zero count condition is output to the External/ Status Interrupt Section. The baud rate generator may be enabled and disabled by command and is disabled by a hardware reset.

The time constant for the baud rate generator is programmed in WR12 and WR13, with the least-significant byte in WR12. The formulas relating the baud rate to the time constant and vice versa are shown with an example. In these formulas the baud rate generator clock frequency is in Hertz, the desired baud rate in bits/ second and the time constant is dimensionless. The example in Table 7 assumes a 2.4576 MHz clock factor of 16 and shows the time constant for a number of popular baud rates.

SUPPORT CIRCUITRY PROGRAMMING (cont'd)

Time Constant Formulas :

$$\text{Time Constant} = \frac{\text{Clock Frequency}}{2 \bullet (\text{Clock Mode}) \cdot (\text{Baud Rate})} - 2$$

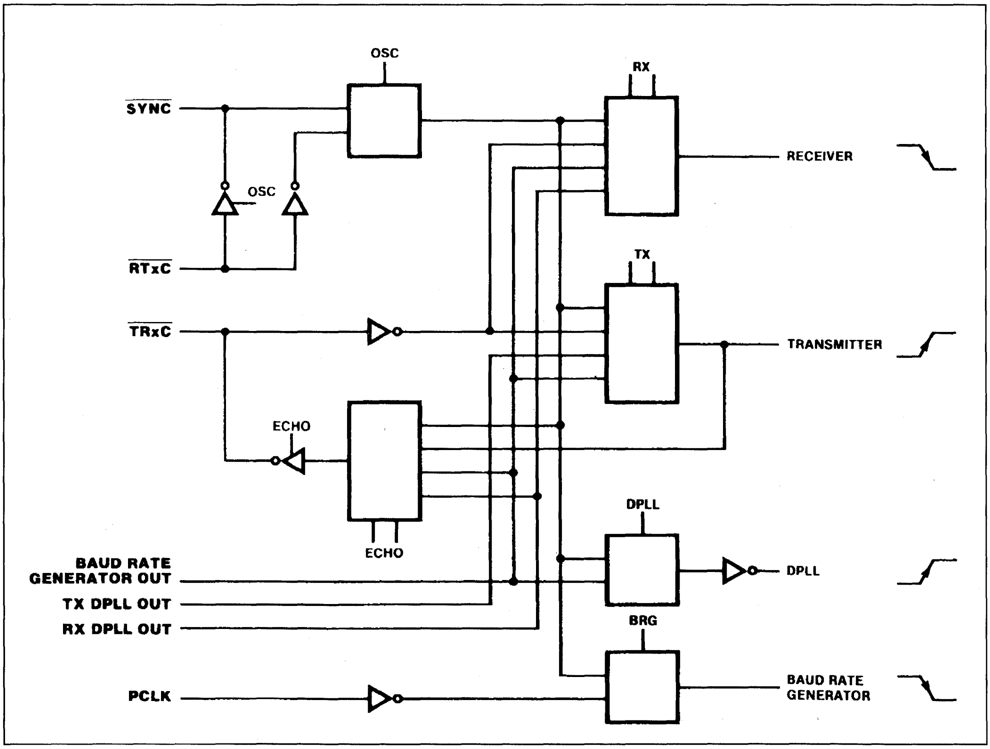
$$\text{Baud Rate} = \frac{\text{Clock Frequency}}{2 \bullet (\text{Clock Mode}) \cdot (\text{Time Const.} + 2)}$$

The clock source for the baud rate generator is selected by bit D_1 of WR14. When this bit is set to "0" the baud rate generator uses the signal on the TxC

Table 7 : Baud Rate Example.

Baud Rate	Divider	
	Decimal	**Hex**
38400	0	0000H
19200	2	0002H
9600	6	0006H
4800	14	000EH
2400	30	001EH
1200	62	003EH
600	126	007EH
300	254	00FEH
150	510	01FEH
For 2.4576 MHz Clock, X 16 Mode		

Figure 32 : Clock Multiplexer.



SUPPORT CIRCUITRY PROGRAMMING (cont'd)

pin as its clock, independent of whether the $\overline{TxC}$ pin is a simple input or part of the crystal oscillator circuit. When this bit is set to "1" the baud rate generator is clocked by PCLK. To avoid metastable problems in the counter, this bit should be changed only while the baud rate generator is disabled, since arbitrarily narrow pulses can be generated at the output of the multiplexer when it changes status. The baud rate generator is enabled while bit D_0 of WR14 is set to "1" and is disabled while this bit is

Figure 33 : Baud Rate Generator.

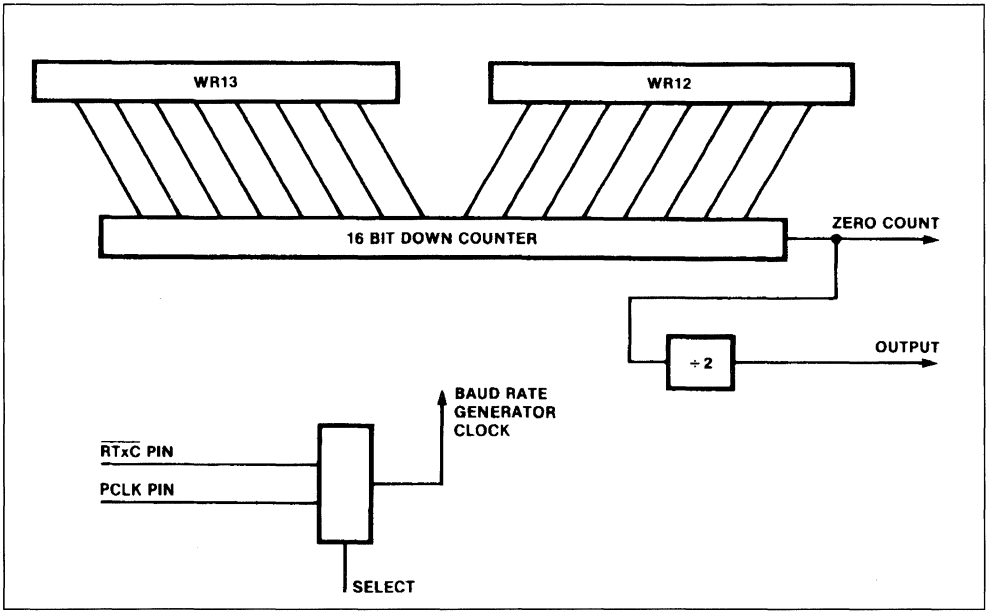
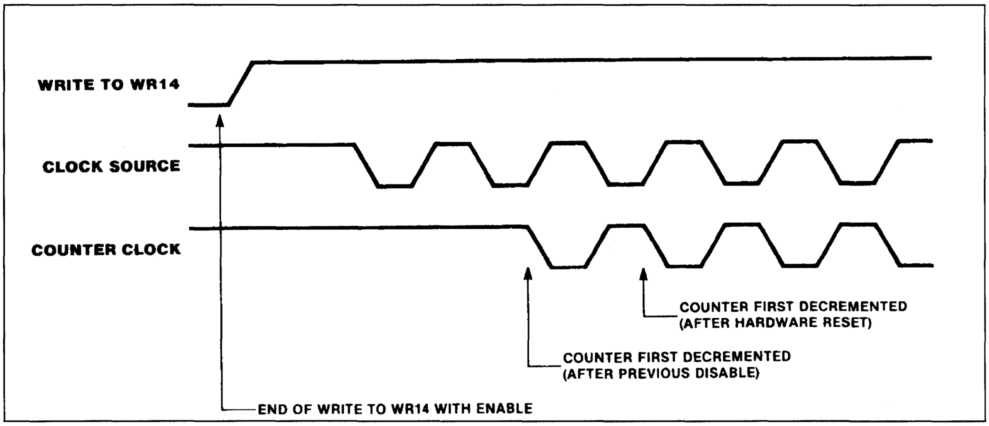


Figure 34 : Baud Rate Generator Start-Up.


WRITE TO WR14
CLOCK SOURCE
COUNTER CLOCK
COUNTER FIRST DECREMENTED (AFTER HARDWARE RESET)
COUNTER FIRST DECREMENTED (AFTER PREVIOUS DISABLE)
END OF WRITE TO WR14 WITH ENABLE

SUPPORT CIRCUITRY PROGRAMMING (cont'd)

set to "0". To prevent metastable problems when the baud rate generator is first enabled, the enable bit is synchronized to the baud rate ge-nerator clock. This introduces an additional delay then the baud rate generator is first enabled and this is shown in Figure 5-3. The baud rate generator is disabled immediately when bit D_0 of WR14 is set to "0", because the delay is only necessary on startup. The baud rate generator may be enabled and disabled on the fly, but this delay on startup must be taken into consideration.

Upon reaching a count of "0" the time constant held in WR12 and WR13 is reloaded into the downcounter so that the process of counting down may start over. In addition to reloading the time constant, the output of the baud rate generator toggles, and for the clock cycle with a zero count, the zero count signal goes active to the External/Status Section. This zero count condition from the baud rate generator does not persist, so if it is to be used by the processor, it should be latched in the External/Status latch. While the baud rate generator is disabled the state of the zero count signal is held. This signal is forced active by a hardware reset.

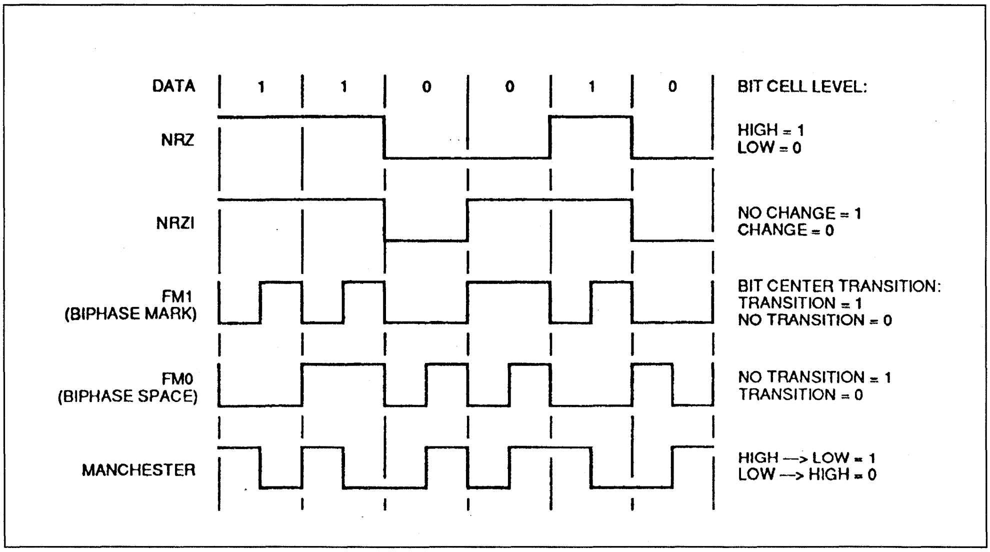
Initializing the baud rate generator is done in four steps. First, the time constant is determined and loaded into WR12 and WR13. Next, the processor must select the clock source for the baud rate generator by writing to bit D_1 of WR14. Finally, the baud rate generator is enabled by setting bit D_0 of WR14 to "1". Note that the first write to WR14 is not necessary after a hardware reset if the clock source is to be the $\overline{\text{RTxC}}$ pin. This is because a hardware reset automatically selects the $\overline{\text{RTxC}}$ pin as the baud rate generator clock source.

Data Encoding

The SCC provides four different data encoding methods, selected by bits D_6 and D_5 in WR10. An example for these four encoding methods is shown in Figure 35. Any encoding method may be used in any X1 mode in the SCC, asynchronous or synchronous. The data encoding selected is active even though the transmitter or receiver may be idling or disabled.

In NRZ encoding a "1" is represented by a HIGH level and a "0" is represented by a LOW level. In this encoding method only a minimal amount of clocking

Figure 35 : Data Encoding Methods.



SUPPORT CIRCUITRY PROGRAMMING (cont'd)

information is available in the data stream in the form of transitions on bit-cell boundaries. In an arbitrary data pattern this may not be sufficient to generate a clock for the data from the data itself. In the case of SDLC, where the number of consecutive "1s" in the data stream is limited, a minimum number of transitions to generate a clock are guaranteed.

In FM1 encoding, also known as biphase mark, a transition is present on every bit cell boundary, and an addition transition may be present in the middle of the bit cell. In FM1 a "0" is sent as no transition in the center of the bit cell and a "1" is sent as a transition in the center of the bit cell. FM1 encoded data contains sufficient information to recover a clock from the data.

In FM0 encoding, also known as biphase space, a transition is present on every bit cell boundary and an additional transition may be present in the middle on the bit cell. In FM0 a "1" is sent as no transition in the center of the bit cell and a "0" is sent as a transition in the center of the bit cell. FM0 encoded data contains sufficient information to recover a clock from the data.

The data encoding method should be selected in the initialization procedure before the transmitter and receiver are enabled but no other restrictions apply. Note, in Figure 35, that in NRZ and NRZI the receiver samples the data only on one edge. However, in FM1 and FM0 the receiver samples the data on both edges. Also, as shown in Figure 5-4, the transmitter defines bit cell boundaries by one edge in all cases and uses the other edge in FM1 and FM0 to create the mid-bit transition.

Digital Phase-locked Loop

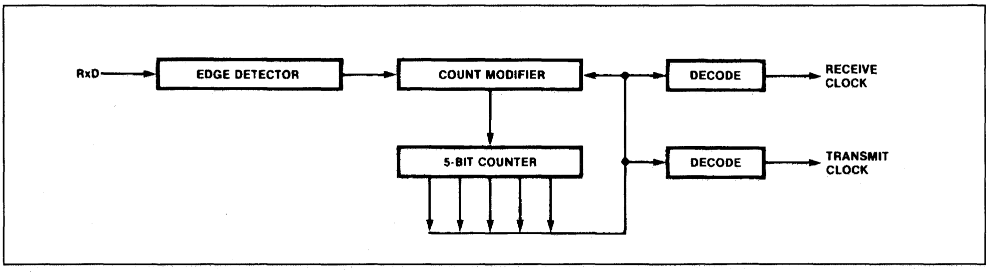
Figure 36 shows a block diagram of the digital phase-locked loop. It consists of a 5-bit counter, an edge detector, and a pair of output decoders. The clock for the DPLL comes from the output of a two-input multiplexer, and the two outputs go to the transmitter and receive clock multiplexers. The DPLL is controlled by the seven commands that are encoded in bits D_7, D_6 and D_5 of WR14.

The clock for the DPLL is selected by two of the commands in WR14. One command selects the output of the baud rate generator as the clock source, and the other command selects the RTxC pin as the clock source, independent of whether the RTxC pin is a simple input or part of the crystal oscillator circuit. To avoid metastable problems in the counter, the clock source selection should be made only while DPLL is disabled, since arbitrarily narrow pulses can be generated at the output of the multiplexer when it changes status.

The DPLL is enabled by issuing the Enter Search Mode command in WR14. This command is also used to reset the DPLL to a known state if it is suspected that synchronization has been lost. When used to enable the DPLL, the Enter Search Mode command unlocks the counter, which is held while the DPLL is disabled and enables the edge detector. If the DPLL is already enabled when this command is issued, the DPLL also enters Search Mode. While in Search mode, the counter is held at a specific count and no outputs are provided. The DPLL remains in this status until an edge is detected in the receive data stream. This first edge is assumed to occur on a bit cell boundary, and the DPLL will begin providing an output to the receiver that will properly sample the data. From this point on the DPLL will adjust its output to remain in phase with the receive data. If the first edge that the DPLL sees does not occur on a bit cell boundary, the DPLL will eventually lock on to the receive data but it will take longer to do so.

The DPLL may be programmed to operate in either of two modes, as selected by command in WR14. In the NRZI mode the DPLL clock must be 32 times the date rate. In this mode the transmit and receive clock outputs of the DPLL are indentical, and the clocks are phased so that the receiver samples the data in the middle of the bit cell. In NRZI mode the DPLL does not require a transition in every bit cell,

Figure 36 : Digital Phase-Locked Loop.



SUPPORT CIRCUITRY PROGRAMMING (cont'd)

so this mode is useful for recovering the clocking information from NRZ and NRZI data streams. In the FM mode the DPLL clock must be 16 times the data rate. In this mode the transmit clock output of the DPLL lags the receive clock outputs by 90o, to make the transmit and receive bit cell boundaries the same, because the receiver must sample FM data at one-quarters and three-quarters bit time. In FM mode the DPLL requires a transition in every bit cell, and if this transition is not present in two consecutively sampled bit cells, the DPLL will automatically enter the search mode. As in the case of the clock source selection, the mode of operation should only be changed while the DPLL is disabled to prevent unpredictable results.

NRZI Mode Operation

To operate in NRZI mode the DPLL must be supplied with a clock that is 32 times the data rate. The DPLL uses this clock, along with the receive data, to construct receive and transmit clock outputs that are phased to properly receive and transmit data. To do this, the DPLL divides each bit cell into four regions, and makes an adjustment to the count cycle of the 5-bit counter dependent upon in which region a transition on the receive data input occured. This is shown in Figure 37. Ordinarily, a bit cell boundary will occur between count 15 and count 16, and the DPLL output will cause the data to be sampled in the middle of the bit cell. The DPLL actually allows the transition marking a bit cell boundary to occur anywhere during the second half of count 15 or the first half of count 16 without making a correction to its count cycle. However, if the transition marking a bit cell boundary occurs between the middle of count 16 and count 31 the DPLL is sampling the data too early in the bit cell. In response to this the DPLL extends its count by one during the next 0 to 31 counting cycle, which effectively moves the edge of the clock that samples the receive data closer to

Figure 37 : DPLL in NRZI Mode.

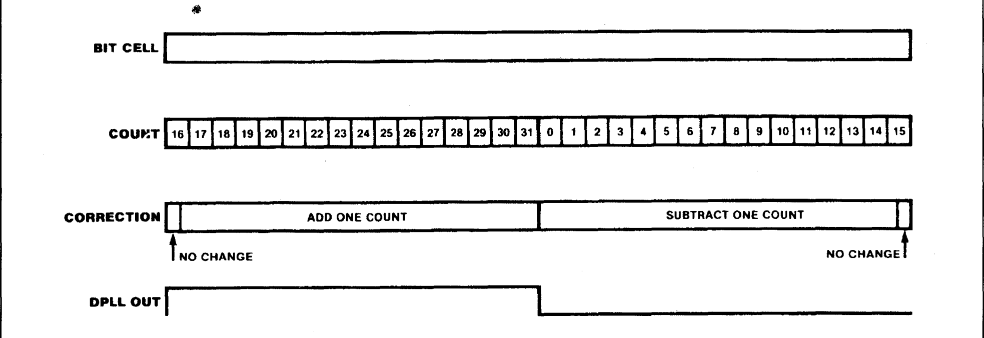
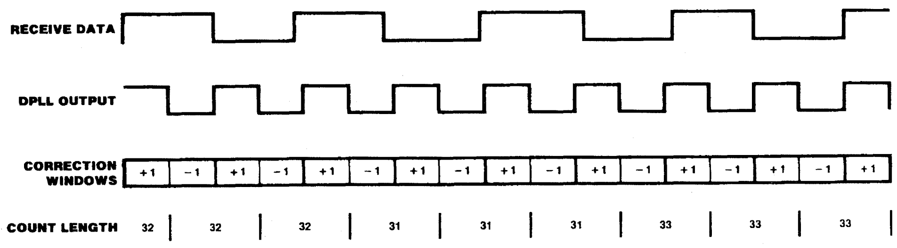


Figure 38 : DPLL Operating Example.



SUPPORT CIRCUITRY PROGRAMMING (cont'd)

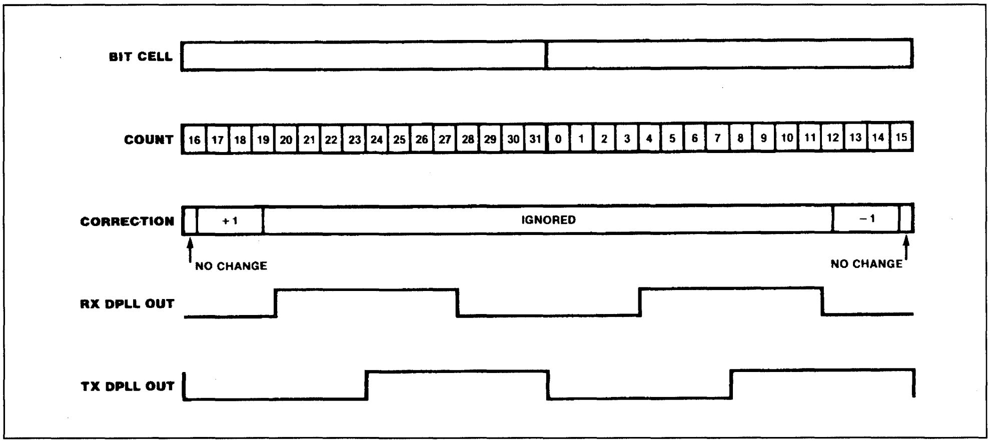
the center of the bit cell. In a similar manner, if the transition occurs between count 0 and the middle of count 15, the output of the DPLL is sampling the data too late in the bit cell. To correct this, the DPLL shortens its count by one during the next 0 to 31 counting cycle, which effectively moves the edge of the clock that samples the receive data closer to the center of the bit cell. In NRZI mode, if the DPLL does not see any transition during a counting cycle, no adjustment is made in the following counting cycle. If an adjustment to the counting cycle is necessary the DPLL modifies count five, either deleting it or doubling it. Thus only the LOW time of the DPLL output will be lengthened or shortened. While the DPLL is in search mode, the counter remains at count 16, where the DPLL outputs are both HIGH. The missing clock latches in the DPLL which may be accessed in RR10, are not used in NRZI mode. An example of the DPLL in operation is shown in Figure 38.

FM Mode Operation

To operate in FM mode the DPLL must be supplied with a clock that is 16 times the data rate. The DPLL uses this clock, along with the receive data, to construct receive and transmit clock outputs that are phased to receive and transmit data properly. In FM mode that the counter in the DPLL still counts from 0 to 31 but now each cycle corresponds to 2-bit cells. To make adjustments to remain in phase with the receive data, the DPLL divides a pair of bit cells into five regions, making the adjustment to the counter dependent upon which region the transition on the receive data input occured. This is shown in Figure 39. Ordinarily a bit cell boundary will occur between count 15 or count 16, and the DPLL receive output will cause the data to be sampled at one-fourth and three-fourths of the way through the bit cell. The DPLL actually allows the transition marking a bit-cell boundary to occur anywhere during the second half of count 15 or the first half of count 16 without making a correction to its count cycle. However, if the transition marking a bit cell boundary occurs between the middle of count 16 and the middle of count 19 the DPLL is sampling the data too early in the bit cell.In response to this the DPLL extends its count by 1 during the next 0 to 31 counting cycle, which effectively moves the receive clock edges closer to where they should be. In FM mode any transitions occurring between the middle of count 19 in one cycle and the middle of count 12 during the next cycle are ignored by the DPLL. This is necessary to guarantee that any data transitions in the bit cells will not cause an adjustment to the counting cycle.

In FM mode the transmit clock and receive clock outputs from the DPLL are not in phase. This is necessary to make the transmit and receive bit cell boundaries coincide, since the receive clock must sample the data one-fourth and three-fourths of the way through the bit cell. As in NRZI mode, if an adjustment to the counting cycle is necessary, the DPLL

Figure 39 : DPLL in FM Mode.



SUPPORT CIRCUITRY PROGRAMMING (cont'd)

modifies count 5, either deleting it or doubling it. If no adjustment is necessary, the count sequence proceeds normally. While the DPLL is in search mode, the counter remains at count 16, where the receive output is LOW and the transmit output is LOW. This fact can be used to provide a transmit clock under software control since the DPLL is in search mode while it it disabled. While the DPLL is disabled the transmit clock output of the DPLL may be toggled by alternately selecting FM and NRZI move in the DPLL. The same is true of the receive clock.

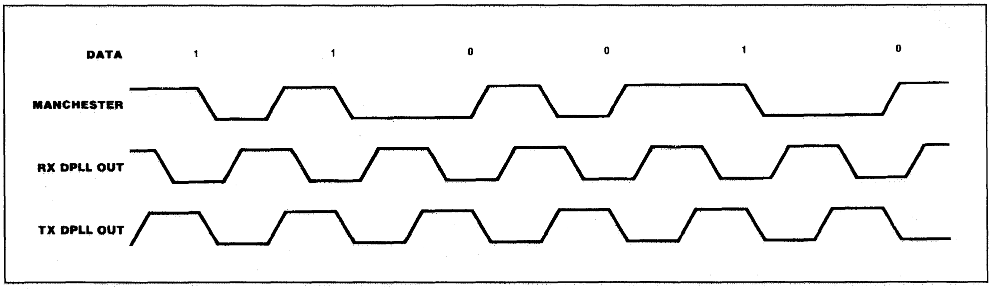
In addition to FM encoded data, the DPLL may also be used to recover the clock from Manchester encoded data, which contains a transition at the center of every bit cell. Here it is the direction of the transition that distinguishes a "1" from a "0". Another way of looking at Manchester encoding is to realize that, during the first half of the bit cell the data is sent, during the second half of the bit cell the complement of the data is sent. This is shown in Figure 40, along with the DPLL output if it thinks that the mid-bit transitions are really bit cell boundaries. As is obvious from the figure, if the receiver samples the data on the falling edge of the DPLL receive clock output, the Manschester data will be properly decoded. This occurs if the receiver is programmed to accept NRZ data.

From the above discussion together with an examination of FM0 and FM1 data encoding, it should be obvious that only clock transitions should exist on the receive data pin when the DPLL is programmed to enter search mode. If this is not case the DPLL may attempt to lock on to the data transitions. With FM0 encoding this requires continuous "1s" received when leaving search. In FM1 encoding it is continuous "0s" ; which Manchester encoded data this means alternating "1s" and "0s". With all three of these data encoding methods there will always be at least one transition in every bit cell, and in FM mode the DPLL is designed to expect this transition. In particular, if no transition occurs between the middle of count 12 and the middle of count 19, the DPLL is propably not locked onto the data properly. When the DPLL misses an edge the One Clock Missing bit is RR10 is set to "1" and latched. It will hold this value until a Reset Mission Clock command is issued in WR14 or until the DPLL is disabled or programmed to enter the Search mode. Upon missing this one edge the DPLL takes no other action and does not modify its count during the next counting cycle. However, if the DPLL does not see an edge between the middle of count 12 and the middle of count 19 in two successive 0 to 31 count cycles, a line error condition is assumed. If this occurs, the two Clocks Mission bit in RR10 is set to "1" and latched. At the same time the DPLL enters the Search mode. The DPLL makes the decision to enter Search mode during count 2, where both the receive clock and transmit clock outputs are LOW. This prevents any glitches on the clock outputs when search mode is entered. While in search mode no clock outputs are provided by the DPLL. The Two Clocks Missing bit in RR10 is latched until a Reset Missing Clock command is issued in WR14, or until the DPLL is disabled or programmed to enter the Search mode.

DPLL Initialization

Initialization of the DPLL may be done at any time during the initialization sequence, but should probably be done after the clock modes have been selected in WR11, and before the receiver and transmitter are enabled. When initializing the DPLL the clock source should be selected first, followed by the selection of the operating mode. At this point the DPLL, may be enabled by issuing the Enter Search Mode command in WR14. Note that a channel or hardware reset disables the DPLL, selects the $\overline{RTxC}$ pin as the clock source for the DPLL, and places it in the NRZI mode.

Figure 40 : Manchester Clock Recovery.



SUPPORT CIRCUITRY PROGRAMMING (cont'd)

Internal Loopback/Auto Echo

The SCC contains two other features useful for diagnostic purposes, controlled by bits in WR14. They are local loopback and auto echo.

Local loopback is selected when bit D4 of WR14 is set to "1". In this mode the output of the transmitter is internally connected to the input of the receiver. At the same time the TxD pin remains connected to the transmitter. In this mode the $\overline{DCD}$ pin is ignored as a receive enable and the $\overline{CTS}$ pin is ignored as a transmitter enable even if the Auot Enables mode has been selected. Note that the DPLL input is connected to the RxD pin, not to the input of the receiver. This precludes the use of the DPLL in local loopback.

Auto echo is selected when bit D3 of WR14 is set to "1". In this mode the TxD pin is connected directly to the RxD pin, and the receiver input is connected to the RxD pin. In this mode the $\overline{CTS}$ pin is ignored as a transmitter enable and the output of the transmitter does not connect to anything. If both the Local Loopback and Auto Echo bits are set to "1", the auto echo mode will be selected, but both the $\overline{CTS}$ pin and $\overline{DCD}$ pin will be ignored as auto enables. This, however, should not be considered a normal operating mode, however. Local Loopback is shown schematically in Figure 41 and auto echo is shown schematically in Figure 42.

Figure 41 : Local Loopback.

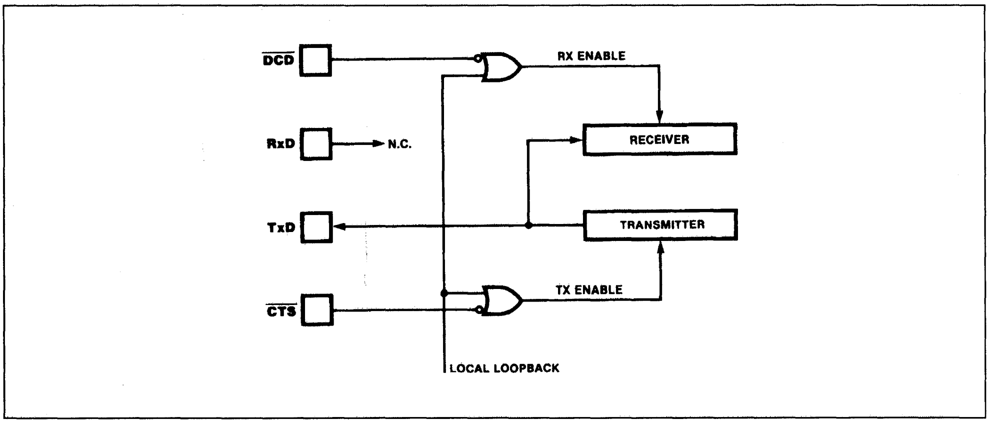
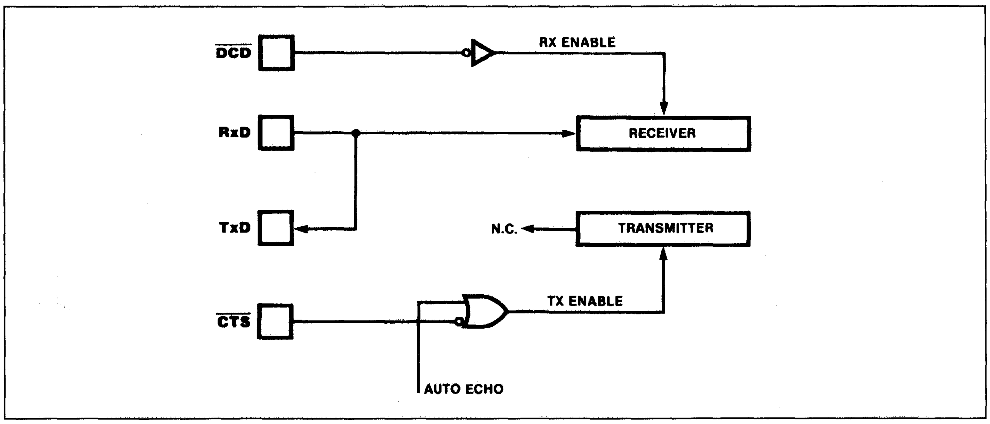


Figure 42 : Auto Echo.



REGISTERS DESCRIPTION

The following section describes the SCC registers. Each register is detailed in terms of bit configuration, the active states (See Table 8) of each bit, their definitions, their functions, and their effects upon the internal hardware and external pins.

Table 8 : SCC Registers Description.

Read Register	Description
RR0	Xmit/Receive Buffer Status and Ext Status
RR1	Receive Condition Status/Residue Codes
RR2	Interrupt Vector (modified in BChannel)
RR3	Interrupt Pending (Channel Aonly)
RR8	Receive Buffer
RR10	Loop/Clock Status
RR12	Lower Byte of Time Constant
RR13	Upper Byte of Time Constant
RR15	External Status Interrupt Enable
Write Register	**Description**
WRO	Command Register
WR1	Tx/Rx Interrupt and Data Xfer Mode Definition
WR2	Interrupt Vector
WR3	Receive Parameters and Control
WR4	Tx/Rx Miscellaneous Parameters and Modes
WR5	Transmit Parameter and Controls
WR6	Sync Character or SDLC Address Field
WR7	Sync Character or SDLC Flag
WR8	Transmit Buffer
WR9	Master Interrupt Control
WR10	Misc Transmitter/Receiver Control Bits
WR11	Clock Mode Control
WR12	Lower Byte Baud Rate Generator Time Constant
WR13	Upper Byte of Baud Rate Generator Time Constant
WR14	Miscellaneous Control Bits
WR15	External Status/Interrupt Control

WRITE REGISTERS

The SCC write register set in each channel includes ten control registers (among them is the transmit buffer), two sync character registers, and two baud rate time constant registers. The interrupt control register and the master interrupt control and reset register are shared by both channels.

Write Register 0 (command register). WR0 is the command register and the CRC reset code register. Figure 43 shows the bit configuration for the Z8530 and includes register select bits in addition to command and reset codes.

Bits D7 and D6 : CRC Reset Codes 0 And 1

Null Code (00). This command has no effect on the SCC and is used when a write to WR0 is necessary for some reason other than a CRC Reset command.

Reset Receive CRC Checker (01). This command is used to initialize the receive CRC circuitry. It is necessary in synchronous modes (except SDLC) if the Enter Hunt Mode command in Write Register 3 is not issued between received messages. Any action that disables the receiver initializes the CRC circuitry. Resetting the Receive CRC Checker command is accomplished automatically in SDLC mode.

Reset Transmit CRC Generator (10). This command initializes the CRC generator. It is usually issued in the initialization routine and after the CRC has been transmitted. A Channel Reset will not initialize the generator and this command should not be issued until after the transmitter has been enabled in the initialization routine.

Reset Transmit Underrun/EOM Latch (11). This command controls the transmission of CRC at the end of transmission (EOM). If this latch has been reset, and a transmit underrun occurs, the SCC automatically appends CRC to the mes-sage. In SDLC mode with Abort on Underrun se-lected, the SCC sends an abort, and Flag on under-run if the TX Underrun/EOM latch as been reset.

At the start of CRC transmission, the Tx Underrun/EOM latch is set. The Reset command can be issued at any time during a message. If the transmitter is disabled, this command will not reset the latch. However, if no External Status interrupt is pending, or if a Reset External Status Int command accompanies this command while the transmitter is disabled, an External/Status interrupt is generated with the Tx Underrun/ EOM bit reset in RR0.

Bits D5-D3 : Command Codes

Null Code (000). The Null command has no effect on the SCC.

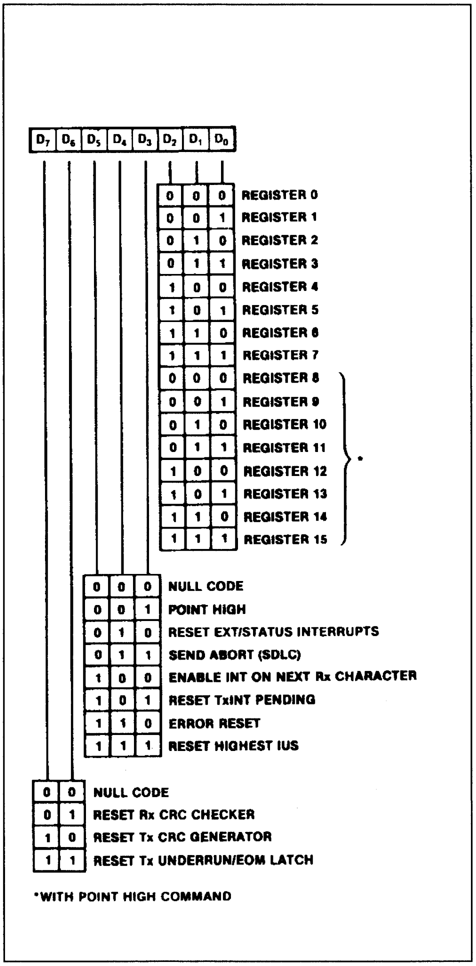
Point High (001). This command effectively adds eight to the Register Pointer (B2-B0) by allowing WR8 through WR15 to be accessed. The Point High command and the Register Pointer bits are written simultaneously.

Reset External/Status Interrupts (010). After an External/Status interrupt (a change on a modem line or a break condition, for example), the status bits in RR0 are latched. This command re-enables the bits and allows interrupts to occur again as a result of a status change. Latching the status bits captures short pulses until the CPU has time to read the change. The SCC contains simple queueing logic associated with most of the external status bits

REGISTERS DESCRIPTION (cont'd)

in RR0. If another External/Status condition changes while a previous condition is still pending (Reset External/Status Interrupts has not yet been issued) and this condition persists until after the command is issued, this second change causes another External/Status interrupt. However, if this second status change does not persist (there are two transitions), another interrupt is not generated. Exceptions to this rule are detailed in the RR0 description.

Figure 43 : Write Register 0 (Z8530).



Send Abort (011). This command is used in SDLC mode to transmit a sequence of eight to thirteen "1s". This command always empties the transmit buffer and sets Tx Underrun/EOM bit in Read Register 0.

Enable Interrupt On Next Rx Character (100). If the interrupt on the First Received Character mode is selected, this command is used to reactivate that mode after each message is received. The next character to enter the receive FIFO causes a Receive interrupt. Alternatively, the first previously stored character in the FIFO will cause a Receive interrupt.

Reset Tx Interrupt Pending (101). This command is used in cases where there are no more characters to be sent ; e.g., at the end of a message. This command prevents further transmit interrupts until after the next character has been loaded into the transmit buffer or until CRC has been completely sent. This command is necessary to prevent the transmitter from requesting an interrupt when the transmit buffer becomes empty (with Transmit Interrupt Enabled).

Error Reset (110). This command resets the error bits in RR1. If Interrupt on First Rx Character or Interrupt on Special Condition modes are selected and a special condition exists, the data with the special condition is held in the receive FIFO until this command is issued. If either of these modes is selected and this command is issued before the data has been read from the receive FIFO, the data is lost.

Reset Highest IUS (111). This command resets the highest priority Interrupt Under Service (IUS) bit, allowing lower priority conditions to request interrupts. This command allows the use of the internal daisy chain (even in systems without an extenal daisy chain) and should be the last operation in an interrupt service routine.

Bits 2 through 0 : Register Selection Code

These three bits select Registers 0 through 7. With the Point High command, Registers 8 through 15 are selected.

The following is a summary of the bit descriptions for each write register (WR1-WR15)

Write Register 1 (transmit/receive interrupt and data transfer mode definition). Write Register 1 is the control register for the various SCC interrupt and Wait/Request modes. Figure 44 shows the bit assignments for WR1.

REGISTERS DESCRIPTION (cont'd)

Bit 7 : WAIT/DMA Request Enable

This bit enables the Wait/Request function in conjunction with the Request/Wait Function Select bit (B6). If bit 7 is set to "1", the state of bit 6 determines the activity of the WAIT/REQUEST pin (Wait or Request). If bit 7 is set to "0", the selected function (bit 6) forces the WAIT/REQUEST pin in to the appropriate inactive state (High for Request, floating for Wait).

Bit 6 : WAIT/DMA Request Function

The request function is selected by setting this bit to "1". In the Request mode, the WAIT/REQUEST pin switches from High to Low when the SCC is ready to transfer data. When this bit is "0", the wait function is selected. In the Wait mode, the WAIT/REQUEST pin switches from floating to Low when the CPU attempts to transfer data before the SCC is ready.

Bit 5 : WAIT/DMA Request On Receive Transmit

This bit determines whether the WAIT/REQUEST pin operates in the Transmit mode or the Receive mode. When set to "1", this bit allows the wait/request function to follow the state of the receive buffer ; i.e., depending on the state of bit 6, the WAIT/REQUEST pin is active or inactive in relation to the empty or full state of the receive buffer. Conversely, if this bit is set to "0", the state of the WAIT/REQUEST pin is determined by bit 6 and the state of the transmit buffer. (Note that a transmit request function is available on the DTR/REQUEST pin. This allows full-duplex operation under DMA control for both channels.)

The request function may occur only when the SCC is not selected ; e.g., if the internal request becomes active while the SCC is in the middle of a read or write cycle, the external request will not become active until the cycle is complete. An active request output causes a DMA controller to initiate a read or write operation. If the request on Transmit mode is selected in either SDLC or Synchronous Mode, the Request pin is pulsed Low for one PCLK cycle at the end of CRC transmission to allow the immediate transmission of another block of data.

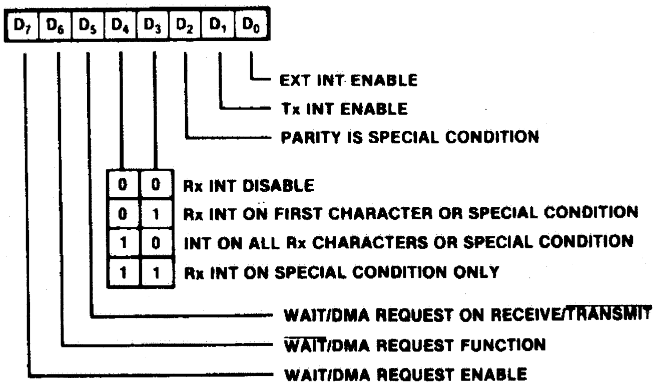
If the Wait On Receive mode, the WAIT pin is active if the CPU attempts to read SCC data that has not yet been received. In the Wait On Transmit mode, the WAIT pin is active if the CPU attempts to write data when the transmit buffer is still full. Both situations can occur frequently when block transfer instructions are used.

Bits 4 and 3 : Receive Interrupt Modes

These two bits specify the various character-available conditions that may cause interrupt requests.

Receive Interrupts Disabled (00). This mode prevents the receiver from requesting an interrupt and is normally used in a polled environment where either the status bits on RR0 or the modified vector in RR2 (Channel B) can be monitored to initiate a service routine. Although the receiver interrupts are disabled, a special condition can still provide a unique vector status in RR2.

Figure 44 : Write Register 1.



D7 D6 D5 D4 D3 D2 D1 D0
EXT INT ENABLE
Tx INT ENABLE
PARITY IS SPECIAL CONDITION
0 0 Rx INT DISABLE
0 1 Rx INT ON FIRST CHARACTER OR SPECIAL CONDITION
1 0 INT ON ALL Rx CHARACTERS OR SPECIAL CONDITION
1 1 Rx INT ON SPECIAL CONDITION ONLY
WAIT/DMA REQUEST ON RECEIVE/TRANSMIT
WAIT/DMA REQUEST FUNCTION
WAIT/DMA REQUEST ENABLE

REGISTERS DESCRIPTION (cont'd)

Receive Interrupt On First Character Or Special Condition (01). The receiver requests an interrupt in this mode on the first available character (or stored FIFO character) or on a special condition. Sync characters to be stripped from the message stream do not cause interrupts.

Special receive conditions are : receiver overrun, framing error, end of frame, or parity error (if selected). If a special receive condition occurs, the data containing the error is stored in the receive FIFO until an Error Reset command is issued by the CPU.

This mode is usually selected when a Block Transfer mode is used. In this interrupt mode, a pending special receive condition remains set until either an Error Reset command, a channel or hardware reset, or until receive interrupts are disabled.

The Receive Interrupt on First Character or Special Condition mode can be re-enabled by the Enable Rx Interrupt on Next Character command in WR0.

Interrupt On All Receive Characters Or Special Condition (10). This mode allows an interrupt for every character received (or character in the receive FIFO) and provides a unique vector when a special condition exists. The receiver Overrun bit and the Parity Error bit in RR1 are two special conditions that are latched. These two bits must be reset by the Error Reset command. Receiver overrun is always a special receive condi-tion, and parity can be programmed to be a special condition.

Data characters with special receive conditions are not held in the receive FIFO in the Interrupt On All Receive Characters or Special Conditions Mode as they are in the other receive interrupt modes.

Receive Interrupt On Special Condition (11). This mode allows the receiver to interrupt only on characters with a special receive condition. When an interrupt occurs, the data containing the error is held in the receive FIFO until an Error Reset command is issued. When using this mode in conjunction with a DMA, the DMA can be initialized and enabled before any characters have been received by the SCC. This eliminates the time critical section of code required in the Receive Interrupt on First Character or Special condition mode ; i.e., all data can be transferred via the DMA so that the CPU need not handle the first received character as a special case.

Bit 2 : Parity Is Special Condition

If this bit is set to "1", any received characters with parity not matching the sense programmed in WR4 give rise to a Special Receive Condition. If parity is disabled (WR4), this bit is ignored. A special condition modifies the status of the interrupt vector stored in WR2. During an interrupt acknowledge cycle, this vector can be placed on the data bus.

Bit 1 : Transmitter Interrupt Enable

If this bit is set to "1", the transmitter requests an interrupt whenever the transmit buffer becomes empty.

Bit 0 : External/Status Master Interrupt Enable

This bit is the master enable for External/Status interrupts including $\overline{DCD}$, $\overline{CTS}$, $\overline{SYNC}$ pins, break, abort, the beginning of CRC transmission when the Transmit/Underrun/EOM latch is set, or when the counter in the baud rate generator reaches "0". Write Register 15 contains the individual enable bits for each of these sources of External/Status interrupts. This bit is reset by a channel or hardware reset.

Figure 45 : Write Register 2.

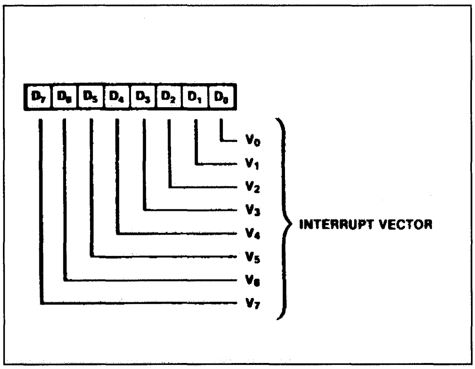
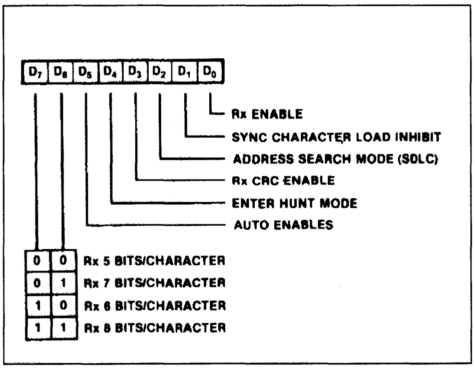


Figure 46 : Write Register 3.



REGISTERS DESCRIPTION (cont'd)

Write Register 2 (interrupt vector). WR2 is the interrupt vector register. Only one vector register exists in the SCC, but it can be accessed through either channel. The interrupt vector can be modified by status information. This is controlled by the Vector Includes Status (VIS) and the Status High/Status Low bits in WR9. The bit positions for WR2 are shown in Figure 45.

Write Register 3 (receive parameters and control). This register contains the control bits and parameters for the receiver logic as illustrated in Figure 46.

Bits 7 and 6 : Receiver Bits/Character

The state of these two bits determines the number of bits to be assembled as a character in the received serial data stream. The number of bits per character can be changed while a character is being assembled but only before the number of bits currently programmed is reached. Unused bits in the Received Data Register (RR8) are set to "1" in asynchronous modes and SDLC modes, the SCC merely transfers an 8-bit section of the serial data stream to the receive FIFO at the appropriate time. Table 9 lists the number of bits per character in the assembled character format.

Table 9 : Receive Bits/Character

B_7	B_6	
0	0	5 Bits / Character
0	1	7 Bits / Character
1	0	6 Bits / Character
1	1	8 Bits / Character

Bit 5 : Auto Enables

This bit programs the function for both the $\overline{DCD}$ and $\overline{CTS}$ pins. $\overline{CTS}$ becomes the transmitter enable and $\overline{DCD}$ becomes the receiver enable when this bit is set to "1". However, the Receiver Enable and Transmit Enable bits must be set before the $\overline{DCD}$ and $\overline{CTS}$ pins can be used in this manner. When the Auto Enables bit is set to "0", the $\overline{DCD}$ and $\overline{CTS}$ pins are merely inputs to the corresponding status bits in Read Register 0. The state of $\overline{DCD}$ is ignored in the Local Loopback mode. The state of $\overline{CTS}$ is ignored in both Auto Echo and Local Loopback modes.

Bit 4 : Enter Hunt Mode

This command forces the comparison of sync characters or flags to assembled receive characters for the purpose of synchronization. After reset, the SCC automatically enters the Hunt mode (except asynchronous). Whenever a flag or sync character is matched, the Sync/Hunt bit in Read Register 0 is reset and, if External/Status Interrupt Enable is set, an interrupt sequence is initiated. The SCC automatically enters the Hunt mode when an abort condition is received or when the receiver is disabled.

Bit 3 : Receiver CRC Enable

This bit is used to initiate CRC calculation at the beginning of the last byte transferred from the Receiver Shift register to the receive FIFO. This operation occurs independently of the number of bytes in the receive FIFO. When a particular byte is to be excluded from CRC calculation, this bit should be reset before the next byte is transferred to the receive FIFO. If this feature is used, care must be taken to ensure that eight bits per character is selected in the receiver because of an inherent delay from the Receive Shift register to the CRC checker.

This bit is internally set to "1" in SDLC mode and the SCC calculates CRC on all bits except inserted zeros between the opening and closing character flags. This bit is ignored in asynchronous mode.

Bits 2 : Adress Search Mode (SDLC)

Setting this bit in SDLC mode causes messages with addresses not matching the address programmed in WR6 to be rejected. No receiver interrupts can occur in this mode unless there is an address match. The address that the SCC attempts to match can be unique (1 in 256) or multiple (16 in 256), depending on the state of Sync Character Load Inhibit bit. The Address Search mode bit is ignored in all modes except SDLC.

Bit 1 : SYNC Character Load Inhibit

If this bit is set to "1" in any synchronous mode except SDLC, the SCC compares the byte in WR6 with the byte about to be stored in the FIFO, and it inhibits this load if the bytes are equal. The SCC does not calculate the CRC on bytes stripped from the Data stream in the manner. If the 6-bit sync option is selected while in Monosync mode, the compare is still across eight bits, so WR6 must be programmed for proper operation.

If the 6-bit sync option is selected with this bit set to "1", all sync characters except the one immediately preceding the data are stripped from the message. If the 6-bit sync option is selected while in the Bisync mode, this bit is ignored.

The address recognition logic of the receiver is modified in SDLC mode if this bit is set to "1", i.e., only the four most significant bits of WR6 must match the receiver address. This procedure allows the

REGISTERS DESCRIPTION (cont'd)

SCC to receive frames from up to 16 separate sources without programming WR6 for each source (if each station address has the four most significant bits in common). The address field in the frame is still eight bits long.

This bit is ignored in SDLC mode if Address Search mode has not been selected.

Bit 0 : Receiver Enable

When this bit is set to "1", receiver operation begins. This bit should be set only after all other receiver parameters are established and the receiver is completely initialized. This bit is reset by a channel or hardware reset command, and it disables the receiver.

Write Register 4 (transmit/receiver miscellaneous parameters and modes). WR4 contains the control bits for both the receiver and the transmitter. These bits should be set in the transmit and receiver initialization routine before issuing the contents of WR1, WR3, WR6, and WR7. Bit positions for WR4 are shown in Figure 47.

Bits 7 and 6 : Clock Rate 1 And 0

These bits specify the multiplier between the clock and data rates. In synchronous modes, the 1S mode is forced internally and these bits are ignored unless External Sync mode has been selected.

1X Mode (00). The clock rate and data rate are the same. In External Sync mode, this bit combination specifies that only the $\overline{\text{SYNC}}$ pin can be used to achieve character synchronization.

16X Mode (01). The clock rate is 16 times the data rate. In External Sync mode, this bit combin-ation specifies that only the $\overline{\text{SYNC}}$ pin can be used to achieve character synchronization.

32X Mode (10). The clock rate is 32 times the data rate. In External Sync mode, this bit combination specifies that either the $\overline{\text{SYNC}}$ pin or a match with the character stored in WR7 will signal character synchronization. The sync character can be either six or eight bits long as specified by the 6-bit/8-bit Sync bit in WR10.

64X Mode (11). The clock rate is 64 times the data rate. With this bit combination in External Sync mode, both the receiver and transmitter are placed in SDLC mode. The only variation from normal SDLC operation is that the $\overline{\text{SYNC}}$ pin can be used to start or stop the reception of a frame by forcing the receiver to act as though a flag had been received.

Bits 5 and 4 : SYNC Modes 1 And 0

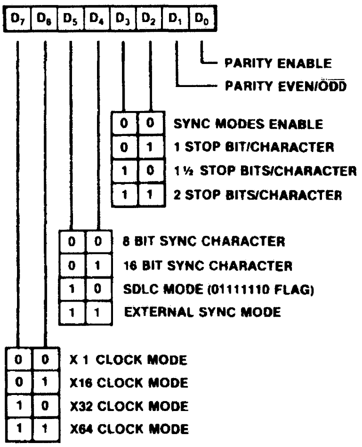
These two bits select the various options for character synchronization. They are ignored unless synchronous modes are selected in the stop bits field of this register.

Monosync (00). In this mode, the receiver achieves character synchronization by matching the character stored in WR7 with an identical character in the received data stream. The transmitter uses the character stored in WR6 as a time fill. The sync character can be either six or eight bits, depending on the state of the 6-bit/8-bit Sync bit in WR10. If the Sync Character Load Inhibit bit is set, the receiver strips the contents of WR6 from the data stream if received within character boundaries.

Bisync (01). The concatenation of WR7 with WR6 is used for receiver synchronization and as a time fill by the transmitter. The sync character can be 12 or 16 bits in the receiver, depending on the state of the 6-bit/8-bit Sync bit in WR10. The transmitted character is always 16 bits.

SDLC Mode (10). In this mode, SDLC is selected

Figure 47 : Write Register 4.



REGISTERS DESCRIPTION (cont'd)

and requires a Flag (01111110) to be written to WR7. The receiver address field should be written to WR6. The SDLC CRC polynomial must also be selected (WR5) in SDLC mode.

External Sync Mode (11). In this mode, the SCC expects external logic to signal character synchronization via the $\overline{\text{SYNC}}$ pin. If the crystal oscillator option is selected (in WR11), the internal $\overline{\text{SYNC}}$ signal is forced to "0". In this mode, bits B7-B6 of this register select special version of External Sync mode. In this mode, the transmitter is in Monosync mode using the contents of WR6 as the time fill with the sync character length specified by the 6-bit/8-bit Sync bit in WR10.

Bits 3 and 2 : Stop Bits 1 and 0

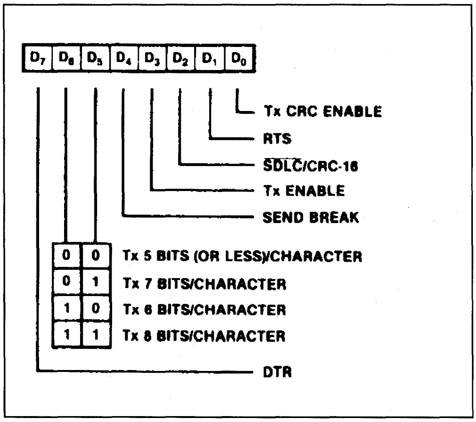
These bits determine the number of stop bits added to each asynchronous character that is transmitted. The receiver always checks for one stop bit in Asynchronous mode. A Special mode specifies that a Synchronous mode is to be selected. B2 is always set to "1" by a channel or hardware reset to ensure that the $\overline{\text{SYNC}}$ pin is in a known state after a reset.

Synchronous Modes Enable (00). This bit combination selects one of the synchronous modes specified by bits B4, B5, B6, and B7 of this register and forces the 1X Clock mode internally.

1 Stop Bit/Character (01). This bit selects Asynchronous mode with one stop bit per character.

1 1/2 Stop Bits/Character (10). These bits select Asynchronous mode with 1-1/2 stop bits per character. This mode can not be used with the 1X clock mode.

Figure 48 : Write Register 5.



2 Stop Bits/Character (11). These bits select Asynchronous mode with two stop bits per transmitted character and check for one received stop bit.

Bit 1 : Parity Even/$\overline{\text{Odd}}$

This bit determines whether parity is checked as even or odd. A "1" programmed here selects even parity, and a "0" selects odd parity. This bit is ignored if the Parity Enable bit is not set.

Bit 0 : Parity Enable

When this bit is set, an additional bit position beyond those specified in the bits/character control is added to the transmitted data and is expected in the receive data. The Received Parity bit is transferred to the CPU as part of the data unless eight bits per character is selected in the receiver.

Write Register 5 (transmit parameter and controls). WR5 contains control bits that affect the operation of the transmitter. B2 affects both the transmitter and the receiver. Bit positions for WR5 are shown in Figure 48.

Bit 7 : Data Terminal Ready

This is the control bit for the $\overline{\text{DTR}}/\overline{\text{REQ}}$ pin while the pin is in the DTR mode (selected in WR14). When set, $\overline{\text{DTR}}$ is Low ; when reset, $\overline{\text{DTR}}$ is High. This bit is ignored when $\overline{\text{DTR}}/\overline{\text{REQ}}$ is programmed to act as a $\overline{\text{REQUEST}}$ pin. This bit is reset by a channel or hardware reset.

Bits 6 and 5 : TXBits/Character 1 and 0

These bits control the number of bits in each byte transferred to the transmit buffer. Bits sent must be right justified with lease significant bits first.

The Five Or Less mode allows transmission of one to five bits per character ; however, the CPU should form at the data character as shown below in Table 10. In the Six or Seven Bits/Character modes, unused data bits are ignored.

Table 10 : Tx Bits/Character 1 and 0.

Tx BITS / CHAR 1	Tx BITS / CHAR 0	
0	0	5 or less bits / character
0	1	7 bits / character
1	0	6 bits / character
1	1	8 bits / character

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	1	1	0	0	0	0	Sends one data bit
1	1	1	0	0	0	0	0	Sends two data bits
1	1	0	0	0	0	0	0	Sends three data bits
1	0	0	0	0	0	0	0	Sends four data bits
0	0	0	0	0	0	0	0	Sends five data bits

REGISTERS DESCRIPTION (cont'd)

Bit 4 : Send Break

When set, this bit forces the TxD output to send continuous "0s" beginning with the following transmit clock, regardless of any data being transmitted at the time. This bit functions whether or not the transmitter is enabled. When reset, TxD continues to send the contents of the Transmit Shift register, which might be syncs, data or all "1s". If this bit is set while in the X21 mode (Monosync and Loop mode selected) and character synchronization is achieved in the receiver, this bit is automatically reset and the transmitter begins sending syncs or data. This bit can also be reset by a channel or hardware reset.

Bit 3 : Transmit Enable

Data is not transmitted until this bit is set, and the TxD output sends continuous "1s" unless Auto Echo mode or SDLC Loop mode is selected. If this bit is reset after transmission started, the transmission of data or sync characters is completed. If the transmitter is disabled during the transmission of a CRC character, sync or flag characters are sent instead of CRC. This bit is reset by a channel or hardware reset.

Bit 2 : $\overline{\text{SDLC}}$/CRC-16

This bit selects the CRC polynomial used by both the transmitter and receiver. When set, the CRC-16 polynomial is used ; when reset, the SDLC polynomial is used. The SDLC/CRC polynomial must be selected when SDLC mode is selected. The CRC generator and checker can be preset to all "0s" or all "1s", depending on the state of the Preset 1/ Preset 0 bit in WR10.

Bit 1 : Request To Send

This is the control bit for the RTS pin. When the RTS bit is set, the RST pin goes Low ; when reset, RTS goes High. In the Asynchronous mode with the Auto Enables bit set, RST goes High only after all bits of the character have been sent and the transmit buffer is empty. In synchronous modes of the Asynchronous mode with auto enables off, the pin directly follows the state of this bit. This bit is reset by a channel or hardware reset.

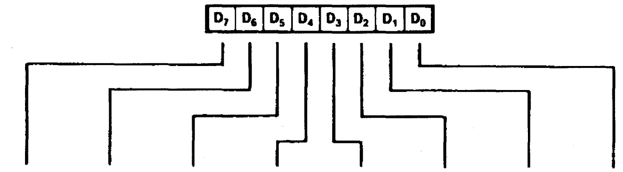
Bit 0 : Transmit CRC Enable

This bit determines whether or not CRC is calculated on a transmit character. If this bit is set at the time the character is loaded from the transmit buffer to the Transmit Shift register, CRC is calculated on that character. CRC is not automatically sent unless this bit is set when the transmit underrun exists.

Write Register 6 (sync characters or SDLC address field). WR6 is programmed to contain the transmit sync character in the Monosync mode, the first byte of a 16-bit sync character in the External Sync mode. WR6 is not used in asynchronous modes. In the SDLC modes, it is programmed to contain the secondary address field used to compare against the address field of the SDLC Frame. In SDLC mode, the SCC does not automatically transmit the station address at the beginning of the response frame. Bit positions for WR6 are shown in Figure 49.

Write Register 7 (sync character or SDLC flag). WR7 is programmed to contain the receive sync character in the Monosync mode, a second byte (the last eight bits) of a 16-bit sync character in the Bisync mode, or a Flag character (01111110) in the SDLC modes. WR7 may hold the receive sync character or a flag if one of the special versions of the External Sync mode is selected. WR7 is not used in Asynchronous mode. Bit positions for WR7 are shown in Figure 50.

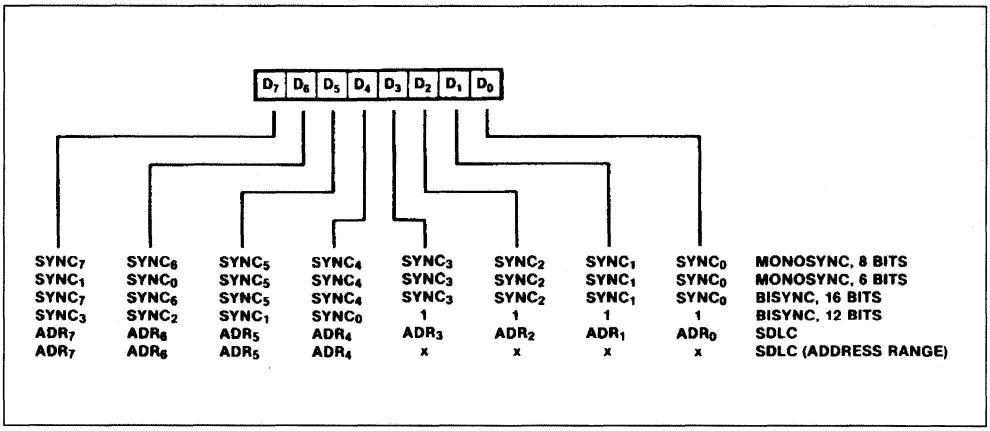
Figure 49 : Write Register 6.



SYNC$_7$	SYNC$_6$	SYNC$_5$	SYNC$_4$	SYNC$_3$	SYNC$_2$	SYNC$_1$	SYNC$_0$	MONOSYNC, 8 BITS
SYNC$_5$	SYNC$_4$	SYNC$_3$	SYNC$_2$	SYNC$_1$	SYNC$_0$	x	x	MONOSYNC, 6 BITS
SYNC$_{15}$	SYNC$_{14}$	SYNC$_{13}$	SYNC$_{12}$	SYNC$_{11}$	SYNC$_{10}$	SYNC$_9$	SYNC$_8$	BISYNC, 16 BITS
SYNC$_{11}$	SYNC$_{10}$	SYNC$_9$	SYNC$_8$	SYNC$_7$	SYNC$_6$	SYNC$_5$	SYNC$_4$	BISYNC, 12 BITS
0	1	1	1	1	1	1	0	SDLC

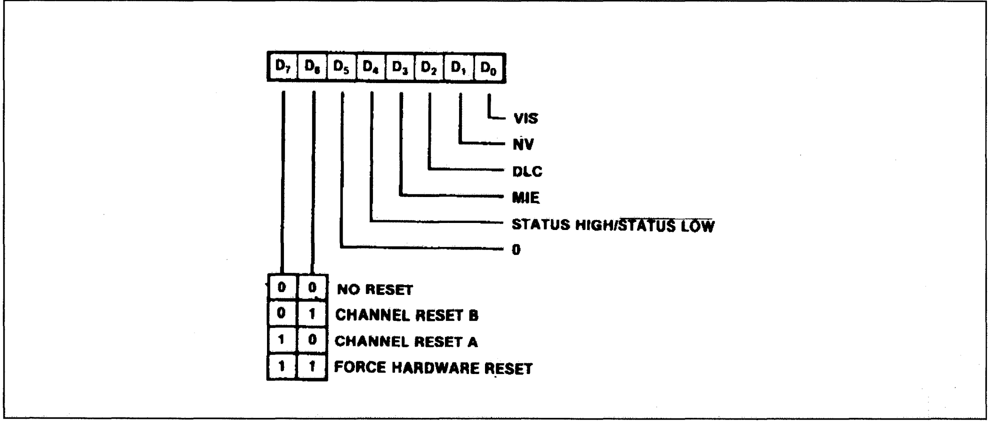
REGISTERS DESCRIPTION (cont'd)

Figure 50 : Write Register 7.



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SYNC$_7$	SYNC$_6$	SYNC$_5$	SYNC$_4$	SYNC$_3$	SYNC$_2$	SYNC$_1$	SYNC$_0$	MONOSYNC, 8 BITS
SYNC$_1$	SYNC$_0$	SYNC$_5$	SYNC$_4$	SYNC$_3$	SYNC$_2$	SYNC$_1$	SYNC$_0$	MONOSYNC, 6 BITS
SYNC$_7$	SYNC$_6$	SYNC$_5$	SYNC$_4$	SYNC$_3$	SYNC$_2$	SYNC$_1$	SYNC$_0$	BISYNC, 16 BITS
SYNC$_3$	SYNC$_2$	SYNC$_1$	SYNC$_0$	1	1	1	1	BISYNC, 12 BITS
ADR$_7$	ADR$_6$	ADR$_5$	ADR$_4$	ADR$_3$	ADR$_2$	ADR$_1$	ADR$_0$	SDLC
ADR$_7$	ADR$_6$	ADR$_5$	ADR$_4$	x	x	x	x	SDLC (ADDRESS RANGE)

Figure 51 : Write Register 9 .



Write Register 8 (transmit buffer). WR8 is the transmit bufffer register.

Write Register 9 (master interrupt control). WR9 is the Master Interrupt Control register and contains the Reset command bits. Only one WR9 exists in the SCC and can be accessed from either channel. The interrupt control bits can be programmed at the same time as the Reset command because these bits are only reset by a hardware reset. Bit positions for WR9 are shown in Figure 51.

Bits 7 and 6 : Reset Command Bits

Together, these bits select one of the reset commands for the SCC. Setting either of these bits to "1" disables both the receiver and the transmitter in the corresponding channel, forces TxD for that channel marking, forces the modem control signals High in that channel, resets all IPs and IUSs and disables all interrupts in that channel. Four extra PCLK cycles must be allowed beyond the usual cycle time after any of the active reset commands is issued before any additional commands or controls are written to the channel affected. Four extra PCLK cycles must be allowed beyond the usual cycle time before any additional command or controls are written to the SCC.

REGISTERS DESCRIPTION (cont'd)

No Reset (00). This command has no effect. It is used when a write to WR9 is necessary for some reason other than an SCC Reset command.

Channel Reset B (01). Issuing this command causes a channel reset to be performed on Channel B.

Channel Reset A (10). Issuing this command causes a channel reset to be performed on Channel A.

Force Hardware Reset (11). The effects of this command are identical to those of a hardware reset, except that the Shift Right/Shift Left bit is not changed and the MIE, Status High/Status Low and DLC bits take the programmed values that accompany this command.

Bit 5 : Not Used

Must be "0".

Bit 4 : Status High/$\overline{\text{Status Low}}$

This bit controls which vector bits the SCC will modify to indicate status. When set to"1", the SCC modifies bits V6, V5, and V4 according to Table 11. When set to "0", the SCC modifies bits V1, V2, and V3 according to Table 11. This bit controls status in both the vector returned during an interrupt acknowledge cycle and the status in RR2B. This bit is reset a hardware reset.

Table 11 : Interrupt Vector Modification.

V3	V2	V1	Status High / Status Low = 0
V4	V5	V6	Status High / Status Low = 1
0	0	0	Ch B Transmit Buffer Empty
0	0	1	Ch B External / Status Change
0	1	0	Ch B Receive Character Avail.
0	1	1	Ch B Special Receive Condition
1	0	0	Ch A Transmit Buffer Empty
1	0	1	Ch A External / Status Change
1	1	0	Ch A Receive Character Avail.
1	1	1	Ch A Special Receive Condition

Bit 3 : Master Interrupt Enable

This bit is set to 1 to globally enable interrupts, and cleared to zero to disable interrupts. Clearing this bit to zero forces the IEO pin to follow the state of the IEI pin unless there is an IUS bit set in the SCC.

No IUS bit can be set after the MIE bit is cleared to zero. This bit is reset by a hardware reset.

Bit 2 : Disable Lower Chain

The Disable Lower Chain bit can be used by the CPU to control the interrupt daisy chain. Setting this bit to "1" forces the IEO pin Low, preventing lower-priority devices on the daisy chain from requesting interrupts. This bit is reset by a hardware reset.

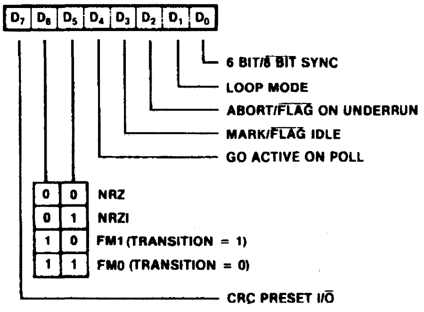
Bit 1 : No Vector

The No Vector bit controls whether or not the SCC will respond to an interrupt acknowledge cycle by placing a vector on the data bus if the SCC is the highest-priority device requesting an interrupt. If this bit is set, no vector is returned ; i.e., AD0-AD7 remain three-stated during an interrupt acknowledge cycle, even if the SCC is the highest-priority device requesting an interrupt.

Bit 0 : Vector Includes Status

The Vector Includes Status Bit controls whether or not then SCC will include status information in the vector it places on the bus in response to an interrupt acknowledge cycle. If this bit is set, the vector returned is variable, with the variable field depending on the highest-priority IP that is set. Table 11 shows the encoding of the status information. This bit is ignored if the No Vector (NV) bit is set.

Figure 52 : Write Register 10.



D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
6 BIT/$\overline{8}$ BIT SYNC
LOOP MODE
ABORT/$\overline{\text{FLAG}}$ ON UNDERRUN
MARK/$\overline{\text{FLAG}}$ IDLE
GO ACTIVE ON POLL
0 0 NRZ
0 1 NRZI
1 0 FM1 (TRANSITION = 1)
1 1 FM0 (TRANSITION = 0)
CRC PRESET I/$\overline{\text{O}}$

Write Register 10 (miscellaneous transmitter/receiver control bits). WR10 contains miscellaneous control bits for both the receiver and the transmitter. Bit positions for WR10 are shown in Figure 52.

Bit 7 : CRC Preset I/$\overline{\text{O}}$

This bit specifies the initialized condition of the receive CRC checker and the transmit CRC generator. If this bit is set to "1", the CRC generator and checker are preset to "1". If this bit is set to "0", the CRC generator and checker are preset to "0".Either option can be selected with either CRC polynomial.

REGISTERS DESCRIPTION (cont'd)

In SDLC mode, the transmitted CRC is inverted before transmission and the received CRC is checked against the bit pattern "0001110100001111". This bit is reset by a channel or hardware reset. This bit is ignored in Asynchronous mode.

Bit 6 and 5 : Data Encoding 1 and 2

These bits control the coding method used for both the transmitter and the receiver, as illustrated in Table 12. All of the clocking options are available for all coding methods. The DPLL in the SCC is useful for recovering clocking information in NRZI and FM modes. A hardware reset forces NRZ mode. Timing for the various modes in shown in Figure 53.

Table 12 : Data Encoding

Data	Data Encoding	Encoding
0	0	NRZ
0	1	NRZI
1	0	FM1 (transition = 1)
1	1	FM0 (transition = 1)

Bit 4 : Go Active On Poll

When Loop mode is first selected during SDLC operation, the SCC connects RxD to TxD with only gate delays in the path. The SCC does not go on-loop and insert the 1-bit delay between RxD and TxD until this bit has been set and an EOP received. When the SCC is on-loop, the transmitter cannot go active unless this bit is set at the time an EOP is received. The SCC examines this bit whenever the transmitter is active in SDLC Loop mode and is sending a flag. If this bit is set at the time the flag is leaving the Transmit Shift register, another flag or data byte (if the transmit buffer is full) is transmitted. If the Go Active on Poll bit is not set at this time, the transmitter finishes sending the flag and reverts to the 1-Bit Delay mode. Thus, to transmit only one response frame, this bit should be reset after the first data byte is sent to the SCC but before CRC has been transmitted. If the bit is not reset before CRC is transmitted, extra flags are sent, slowing down response time on the loop. If this bit is reset before the first data is written, the SCC completes the transmission of the present flag and reverts to the 1-Bit Delay mode. After gaining control of the loop, the SCC is not able to transmit again until a flag and another EOP have been received. Though not strictly necessary, it is good practice to set this bit only upon receipt of a poll frame to ensure that the SCC does not go on loop without the CPU noticing it.

In synchronous modes other than SDLC with the Loop Mode bit set, this bit must be set before the transmitter can go active in response to a received sync character.

This bit is always ignored in Asynchronous mode and Synchronous modes unless the Loop Mode bit is set. This bit is reset by a channel or hardware reset.

Bit 3 : Mark/ $\overline{\text{Flag}}$ Idle

This bit affects only SDLC operation and is used to control the idle line condition. If this bit is set to "0", the transmitter sends flags an idle line. If this bit is set to "1", the transmitter sends continuous "1s" after the closing flag of a frame. The idle line condition is selected byte by byte ; i.e., either a flag or eight "1s" are transmitted. The primary station in an SDLC loop should be programmed for Mark Idle to create the EOP sequence . Mark Idle must be deselected at the beginning of a frame before the first data is written to the SCC, so that an opening flag can be transmitted. This bit is ignored in Loop mode, but the programmed value takes effect upon exiting the Loop mode. This bit is reset by a channel or hardware reset.

Bit 2 : Abort/$\overline{\text{Flag}}$ On Underrun

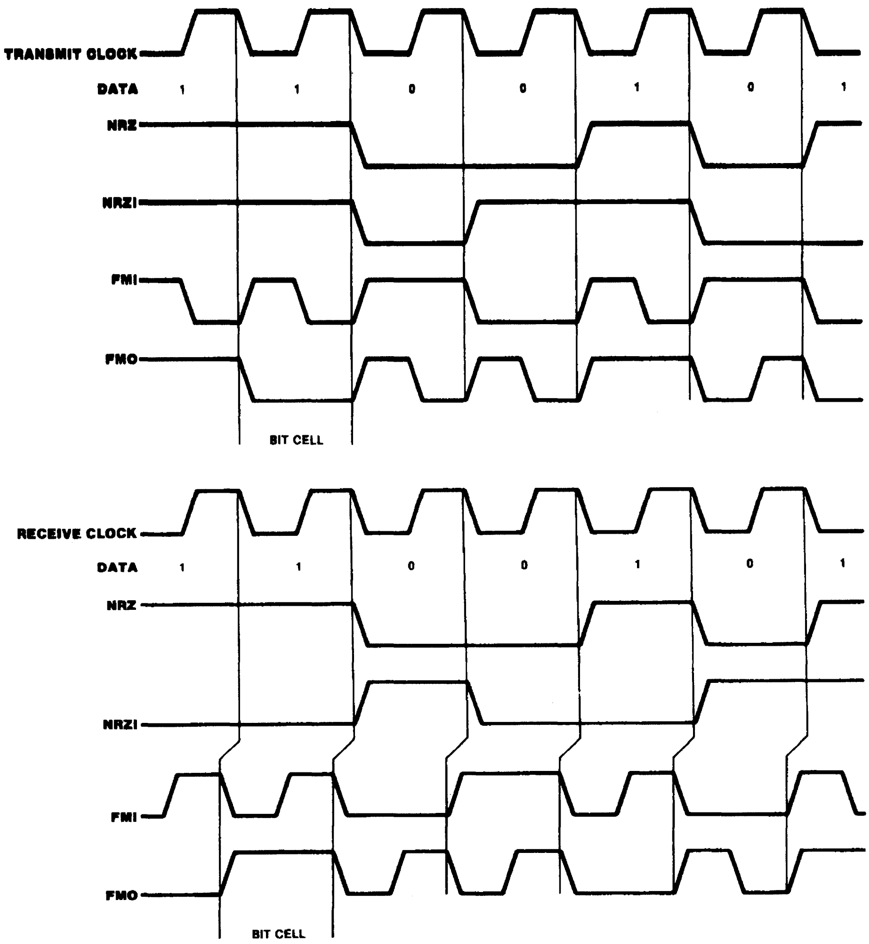
This bit affects only SDLC operation and is used to control how the SCC responds to a transmit underrun condition. If this bit is set to "1" and a transmit underrun occurs, the SCC sends an abort and a flag instead of CRC. If this bit is reset, the SCC sends CRC on a transmit underrun. At the beginning of this 16-bit transmission, the Transmit Underrun/EOM bit is set, causing an External/Status interrupt. The CPU uses this status, along with the byte count from memory or the DMA, to determine whether the frame must be retransmitted. A transmit buffer Empty interrupt occurs at the end of this 16-bit transmission to start the next frame. If both this bit and the Mark/Flag Idle bit are set to "1", all "1s" are transmitted after the transmit underrun. This bit should be set after the first byte of data is sent to the SCC and reset immediately after the last byte of data so that the frame will be terminated properly with CRC and a flag. This bit is ignored in Loop mode, but the programmed value is active upon exiting Loop mode. This bit is reset by a channel or hardware reset.

Bit 1 : Loop Mode

In SDLC mode, the initial set condition of this bit forces the SCC to connect TxD to TxD and to begin searching the incoming data stream so that it can go on loop. All bits pertinent to SDLC mode operation in other registers must be set before this mode

REGISTERS DESCRIPTION (cont'd)

Figure 53 : NRZ(NRZI)FM1(FM0) Timing.



REGISTERS DESCRIPTION (cont'd)

is selected. The transmitter and receiver should not be enabled until after this mode has been selected. As soon as the Go Active Onn Poll bit is set and an EOP is received the SCC goes on loop. If this bit is reset after the SCC is on loop, the SCC waits for the next EOP to go off loop.

In synchronous modes, the SCC uses this bit, along with the Go Active On Poll bit, to synchronize the transmitter to the receiver. The receiver should not be enabled until after this mode is selected. The TxD pin is held marking when this mode is selected unless a break condition is programmed. The receiver waits for a sync character to be received and then enables the transmitter on a character boundary. The break condition, if programmed, is removed. This mode works properly with sync characters of 6,8, or 16 bits. This bit is ignored in Asynchronous mode and is reset by a channel or hardware reset.

Bit 0 : 6 Bit/$\overline{8}$ $\overline{\text{Bit}}$ $\overline{\text{SYNC}}$

This bit is used to select a special case of synchronous modes. If this bit is set to "1" in Monosync mode, the receiver and transmitter sync characters are six bits long instead of the usual eight. If this bit is set to "1" in Bisync mode, the received sync will be 12 bits and the transmitter sync character will remain 16 bits long. This bit is ignored in SDLC and Asynchronous modes but still has effect in the special external sync modes. This bit is reset by a channel or hardware reset.

Write Register 11 (clock mode control). WR11 is the Clock Mode Control register. The bits in this register control the sources of the both the receive and transmit clocks, the type of signal on the $\overline{\text{SYNC}}$ and $\overline{\text{RTxC}}$ pins, and the direction of the $\overline{\text{TRxC}}$ pin. Bit positions for WR11 are shown in Figure 54.

Bit 7 : RTxC–XTAL/$\overline{\text{NO XTAL}}$

This bit controls the type of input signal the SCC expects to see on the $\overline{\text{RTxC}}$ pin. If this bit is set to "0", the SCC expects a TTL-compatible signal as an input to this pin. If this bit is set to "1", the SCC connects a high-gain amplifier between the $\overline{\text{RTxC}}$ and $\overline{\text{SYNC}}$ pins in expectation of a quartz crystal being placed across the pins.

The output of this oscillator is available for use as a clocking source. In this mode of operation, the $\overline{\text{SYNC}}$ pin is unavailable for other use. The $\overline{\text{SYNC}}$-signal is forced to "0" internally. A hardware reset forces $\overline{\text{NO XTAL}}$. (At least 20 ms should be allowed after this bit is set to allow the oscillator to stabilize.)

Bits 6 and 5 : Receiver Clock 1 And 0

These bits determine the source of the receive clock as shown in Table 13. They do not interfere with any of the modes of operation in the SCC but simply control a multiplexer just before the internal receive clock input. A hardware reset forces the receive clock to come from the $\overline{\text{TRxC}}$ pin.

Table 13 : Receive Clock Source.

Receive Clock 1	Receive Clock 0	
0	0	Receive Clock = RTxC Pin
0	1	Receive Clock = TRxC Pin
1	0	Receive Clock = BR Output
1	1	Receive Clock = DPLL Output

Bits 4 and 3 : Transmit Clock 1 and 0

These bits determine the source of the transmit clock as shown in Table 14. They do not interfere with any of the modes of operation of the SCC but simply control a multiplexer just before the internal transmit clock input. The DPLL output that may be used to feed the transmitter in FM modes lags by 90 the output of the DPLL used by the receiver. This makes the received and transmitted bit cells occur simultaneously, neglecting delays. A hardware reset selects the $\overline{\text{TRxC}}$ pin as the source of the transmit clocks.

Table 14 : Transmit Clock Source.

Transmit Clock 1	Transmit Clock 0	
0	0	Transmit Clock = RTxC Pin
0	1	Transmit Clock = TRxC Pin
1	0	Transmit Clock = BR Output
1	1	Transmit Clock = DPLL Output

Bit 2 : TRxC O/I

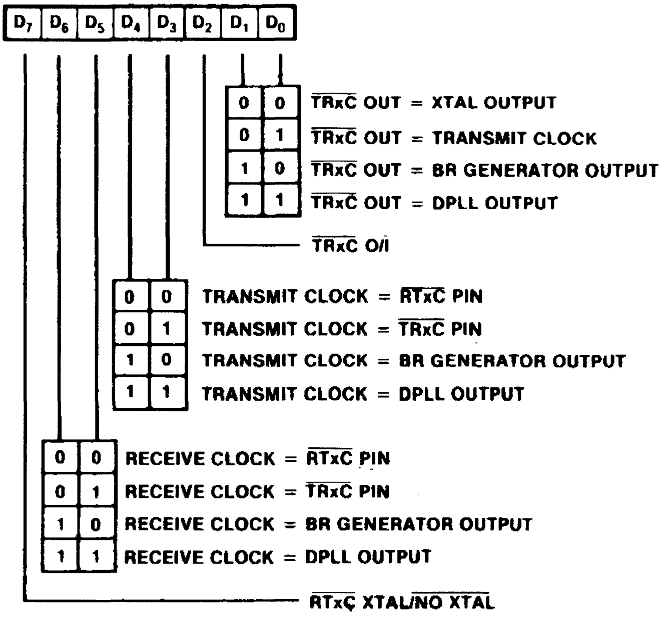
This bit determines the direction of the $\overline{\text{TRxC}}$ pin. If this bit is set to "1", the TRxC pin is an output and carries the signal selected by D1 and D0 of this register. However, if either the receive or the transmit clock is programmed to come from the $\overline{\text{TRxC}}$ pin, $\overline{\text{TRxC}}$ will be an input, regardless of the state of this bit. The $\overline{\text{TRxC}}$ pin is also an input if this bit is set to "0". A hardware reset forces this bit to "0".

Bits 1 and 0 : $\overline{\text{TRxC}}$ Output Source 1 And 0

These bits determine the signal to be echoed out of the SCC via the $\overline{\text{TRxC}}$ pin. No signal is produced if $\overline{\text{TRxC}}$ has been programmed as the source of either

REGISTERS DESCRIPTION (cont'd)

Figure 54 : Write Register 11.


D7 D6 D5 D4 D3 D2 D1 D0
0 0 TRxC OUT = XTAL OUTPUT
0 1 TRxC OUT = TRANSMIT CLOCK
1 0 TRxC OUT = BR GENERATOR OUTPUT
1 1 TRxC OUT = DPLL OUTPUT
TRxC O/I
0 0 TRANSMIT CLOCK = RTxC PIN
0 1 TRANSMIT CLOCK = TRxC PIN
1 0 TRANSMIT CLOCK = BR GENERATOR OUTPUT
1 1 TRANSMIT CLOCK = DPLL OUTPUT
0 0 RECEIVE CLOCK = RTxC PIN
0 1 RECEIVE CLOCK = TRxC PIN
1 0 RECEIVE CLOCK = BR GENERATOR OUTPUT
1 1 RECEIVE CLOCK = DPLL OUTPUT
RTxC XTAL/NO XTAL

the receive or the transmit clock. If TRxC O/I (bit 2) is set to "0", these bits are ignored.

If the XTAL oscillator output is programmed to be echoed, and the Xtal oscillator has not been enabled, the TRxC pin goes High. The DPLL signal that is echoed is the DPLL signal used by the receiver. Hardware reset selects the XTAL oscillator as the output source.

Table 15 : Transmit External Control Selection.

Output Signal	Output Signal	
0	0	TRxC = XTAL Oscillator Output
0	1	TRxC = Transmit Clock
1	0	TRxC = B R Output
1	1	TRxC = DPLL Output (Receive)

Write Register 12 (lower byte of baud rate generator time constant). WR12 contains the lower byte of the time constant for the baud rate generator. The time constant can be changed at any time, but the new value does not take effect until the next time the time constant is loaded into the down counter. No attempt is made to synchronize the loading of the time constant into WR12 and WR13 with the clock driving the down counter. For this reason, it is advisable to disable the baud rate generator while the new time constant is loaded into WR12 and WR13. Ordinarily, this is done anyway to prevent a load of the down counter between the writing of the upper and lower bytes of the time constant.

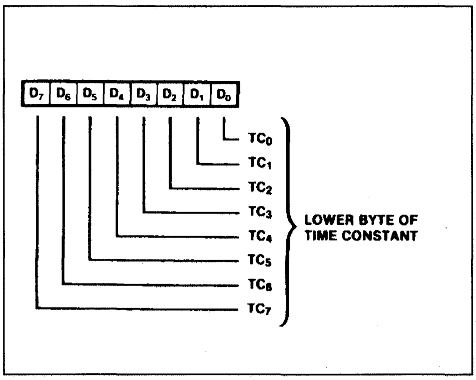
The formula for determining the appropriate time constant for a given baud is shown below with the desired rate in bits per second and the BR clock pe-

REGISTERS DESCRIPTION (cont'd)

riod in seconds. This formula is derived because the counter decrements from N down to "0"- plus-one-cycle for reloading the time constant and is then fed to a toggle flip-flop to make the output a square wave. Bit positions for WR12 are shown in Figure 55.

Time constant = [1/2 desired rate • BR clock period] - 2

Figure 55 : Write Register 12.



D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
TC_0
TC_1
TC_2
TC_3
TC_4
TC_5
TC_6
TC_7
LOWER BYTE OF TIME CONSTANT

Write Register 13 (upper byte of baud rate generator time constant). WR13 contains the upper byte of the time constant for the baud rate generator. Bit positions for WR13 are shown in Figure 56.

Write Register 14 (miscellaneous control bits). WR14 contains some miscellaneous control bits. Bit positions for WR14 are shown in Figure 57.

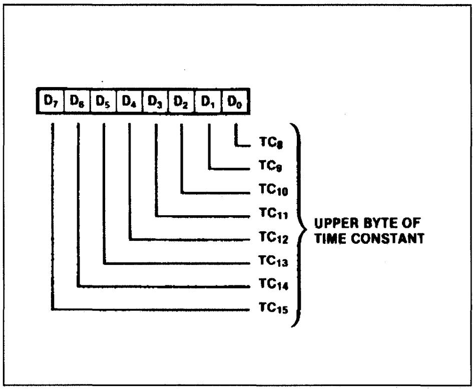
Bits 7 and 5 : Digital Phase-Locked Loop Command Bits

These three bits encode the eight commands for the Digital Phase-Locked Loop. A channel or hardware reset disables the DPLL, resets the mission clock latches, sets the source to the RTxC pin and selects NRZI mode. The Enter Search Mode command enables the DPLL after a reset.

Null Command (000). This command has no effect on the DPLL.

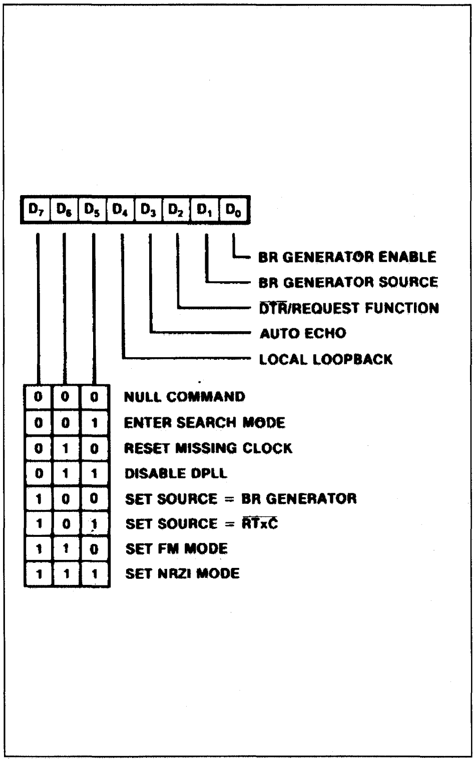
Enter Search Mode (001). Issuing this command causes the DPLL to enter the Search mode, where the DPLL searches for a locking edge in the incoming data stream. The action taken by the DPLL upon receipt of this command depends on the operating mode of the DPLL.

Figure 56 : Write Register 13.



D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
TC_8
TC_9
TC_{10}
TC_{11}
TC_{12}
TC_{13}
TC_{14}
TC_{15}
UPPER BYTE OF TIME CONSTANT

Figure 57 : Write Register 14.



D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
BR GENERATOR ENABLE
BR GENERATOR SOURCE
$\overline{DTR}$/REQUEST FUNCTION
AUTO ECHO
LOCAL LOOPBACK
0 0 0 NULL COMMAND
0 0 1 ENTER SEARCH MODE
0 1 0 RESET MISSING CLOCK
0 1 1 DISABLE DPLL
1 0 0 SET SOURCE = BR GENERATOR
1 0 1 SET SOURCE = $\overline{RTxC}$
1 1 0 SET FM MODE
1 1 1 SET NRZI MODE

REGISTERS DESCRIPTION (cont'd)

In NRZI mode, the output of the DPLL is High while the DPLL is waiting for an edge in the incoming data stream. After the Search mode is entered, the first edge the DPLL sees is assumed to be a valid data edge, and the DPLL begins the clock recovery operation from that point. The DPLL clock rate must be 32 times the data rate in NRZI mode. Upon leaving the Search mode, the first sampling edge of the DPLL occurs 16 of these 32X clocks after the first data edge and the second sampling edge occurs 48 of these 32X clocks after the first data edge. Beyond this point, the DPLL begins normal operation, adjusting the output to remain in sync with the incoming data.

In FM mode, the output of the DPLL is Low while the DPLL is waiting for an edge in the incoming data stream. The first edge the DPLL detects is assumed to be a valid clock edge. For this to be the case, the line must contain only clock edges ; i.e., with FM1 encoding, the line must be continuous "0s". With FM0 encoding the line must be continuous "1s", whereas Manchester encoding requires alternating "1s" and "0s" on the line. The DPLL clock rate must be 16 times the data rate in FM mode. The DPLL output causes the receiver to sample the data stream in the nominal center of the two halves of the bit cell to decide whether the data was a "1" or a "0". After this command is issued, as in NRZI mode, the DPLL starts sampling immediately after the first edge is detected. (In FM mode, the DPLL examines the clock edge of every other bit cell to decide what correction must be made to remain in sync). If the DPLL does not see an edge during the expected window, the one clock mission bit in RR10 is set. If the DPLL does not see an edge after two successive attempts, the two clocks missing bit in RR10 is set and the DPLL automatically enters the Search mode. This command resets both clock missing latches.

Reset Clock Missing (010). Issuing this command disables the DPLL, resets the clock missing latches in RR10, and forces a continuous Search mode state.

Disable DPLL (001). Issuing this command disables the DPLL, resets the clock missing latches in RR10, and forces a continuous Search mode state.

Set Source = BR Gen (100). Issuing this command forces the clock for the DPLL to come from the output of the baud rate generator.

Set Source = $\overline{\text{RTxC (101)}}$. Issuing the command forces the clock for the DPll to come from the RTxC pin or the crystal oscillator, de-pending on the state of the XTAL/no XTAL bit in WR11. This mode is selected by a channel or hardware reset.

Set FM Mode (110). This command forces the DPLL to operate in the FM mode and is used to recover the clock from FM or Manchester-encoded data. (Manchester is decoded by placing the receiver in NRZ mode while the DPLL is in FM mode.)

Set NRZI Mode (111). Issuing this command forces the DPLL to operate in the NRZI mode. This mode is also selected by a hardware or channel reset.

Bit 4 : Local Loopback

Setting this bit to "1" selects the local Loopback mode of operation. In this mode, the internal transmitted data is routed back to the receiver, as well as to the TxD pin. The $\overline{\text{CTS}}$ ans $\overline{\text{DCD}}$ inputs are ignored as enables in Local Loopback mode, even if auto enables is selected. (If so programmed, transitions on these inputs still cause interrupts.) This mode works with any Transmit/Receive mode ex-cept Loop mode. For meaning-ful results, the frequency of the transmit and receive clocks must be the same. This bit is reset by a channel or hardware reset.

Bit 3 : Auto Enable

Setting this bit to "1" selects the Auto Enable mode of operation. In this mode, the TxD pin is connected to RxD, as in Local Loopback mode, but the receiver still listens to the RxD input. Transmitted data is never seen inside or outside the SCC in this mode, and $\overline{\text{CTS}}$ is ignored as a transmit enable. This bit is reset by a channel or hardware reset.

Bit 2 : DTR/Request Function

This bit selects the function of the $\overline{\text{DTR}}/\overline{\text{REQ}}$ pin follows the state of the DTR bit in WR5. If this bit is set to "1", the $\overline{\text{DTR}}/\overline{\text{REQ}}$ pin follows the state of the DTR bit in WR5. If this bit is set to "1", the $\overline{\text{DTR}}/\overline{\text{REQ}}$ pin goes Low whenever the transmit buffer becomes empty and in any of the synchronous mode when CRC has been sent at the end of a message. The request function on the $\overline{\text{DTR}}/\overline{\text{REQ}}$ pin differs somewhat from the transmit request function available on the $\overline{\text{W/REQ}}$ pin in that $\overline{\text{REQUEST}}$ does not go inactive until the internal operation satisfying the request is complete, which occurs four to five PCLK cycles after the rising edge of $\overline{\text{DS}}$, $\overline{\text{READ}}$ or $\overline{\text{WRITE}}$. If the DMA used is edge-triggered, this difference is unimportant. This bit is reset by a channel or hardware reset.

Bit 1 : Baud Rate Generator Source

This bit selects the source of the clock for the baud rate generator. If this bit is set to "0", the baud rate

REGISTERS DESCRIPTION (cont'd)

generator clock comes from either the $\overline{\mathrm{RTxC}}$ pin or the XTAL oscillator (depending on the state of the XTAL/no XTAL bit). If this bit is set to "1", the clock for the baud rate generator is the SCC's PCLK input. Hardware reset sets this bit to "0", selecting the $\overline{\mathrm{RTxC}}$ pin as the clock source for the baud rate generator.

Bit 0 : Baud Rate Generator Enable

This bit controls the operation of the baud rate generator. The counter in the baud rate generator is enabled for counting when this bit is set to "1", and counting is inhibited when this bit is set to "0". When this bit is set to "1", change in the state of this bit is not reflected by the output of the baud rate generator for two counts of the counter. This allows the command to be synchronized. However, when set to "0", disabling is immediate. This bit is reset by a hardware reset.

Write Register 15 (external/status interrupt control). WR15 is the External/Status Source Control register. If the External/Status interrupts are enabled as a group via WR1, bits in this register control which External/Status conditions can cause an interrupt. Only the External/Status conditions that occur after the controlling bit are sent to "1" will cause an interrupt. This is true even if an External/Status condition is pending at the time the bit is set. Bit positions for WR15 are shown in Figure 58.

Bit 7 : Break/Abort IE

If this bit is set to "1", a change in the Break/Abort status of the receiver causes an External/Status interrupt. This bit is set by a channel or hardware reset.

Bit 6 : Tx Underrun/EOM

If this bit is set to "1", a change of state by the Tx Underrun/EOM latch in the transmitter causes an External/Status interrupt. This bit is set to "1" a channel or hardware reset.

Bit 5 : CTS IE

If this bit is set to "1", a change of state on the $\overline{\mathrm{CTS}}$ pin causes an External/Status interrupt. This bit is set by a channel or hardware reset.

Bit 4 : SYNC/Hunt IE

If this bit is set to "1", a change of state on the $\overline{\mathrm{SYNC}}$ pin causes an External/Status interrupt in Asynchronous mode, and a change of state in the Hunt bit in the receiver causes and External/Status interrupt in synchronous modes. This bit is set by a channel or hardware reset.

Bit 3 : DCD IE

If this bit is set to "1", a change of state on the $\overline{\mathrm{DCD}}$ pin causes an External/Status interrupt. This bit is set by a channel or hardware reset.

Bit 2 : Not Used

Must be "0".

Bit 1 : Zero Count IE

If this bit is set to "1", an External/Status interrupt is generated whenever the counter in the baud rate generator reaches "0". This bit is set to "0" by a channel or hardware reset.

Bit 0 : Not Used

Must be "0".

READ REGISTERS

The Z8530 SCC contains seven read registers in each channel. In addition there are two registers which are shared by both channels. The status of these registers is continually changing and depends on the mode of communication, received and transmitted data, and the manner in which this data is transferred to and from the CPU. The following description details the bit assignments for each register.

Read Register 0 (transmit/ receiver buffer status and external status). Read Register 0 contains the status of the receive and transmit buffers. RR0 also contains the status bits for the six sources of External/Status interrupts. The bit configuration is illustrated in Figure 59.

Bit 7 : Break/Abort

In the Asynchronous mode, this bit is set when a Break sequence (null character plus framing error) is detected in the receive data stream. This bit is reset when the sequence is terminated, leaving a single null character in the receive FIFO. This character should be read and discarded. In SDLC mode, this bit is set by the detection of an Abort sequence (seven or more "1s"), then reset automatically at the termination of the Abort sequence. In either case, if the Break/Abort IE bit is set, an External/Status interrupt is initiated. Unlike the remainder of the External/Status bits, both transitions are guaranteed to

REGISTERS DESCRIPTION (cont'd)

cause an External/Status interrupt, even if another External/Status interrupt is pending at the time these transitions occur. This procedure is necessary because Abort or Break conditions may not persist.

Bit 6 : TX Underrun/EOM

This bit is set by a channel or hardware reset and when the transmitter is disabled or a Send Abort command is issued. This bit can only be reset by the reset Tx Underrrun/EOM Latch command in WR0. When the Transmit Underrun occurs, this bit is set and causes an External/Status interrupt (if the Tx Underrun/EOM IE bit is set).

Only the 0-to-1 transition of this bit causes an interrupt. This bit is always "1" in Asynchronous mode, unless a reset Tx Underrun/EOM Latch command has been erroneously issued. In this case, the Send Abort command can be used to set the bit to one and at the same time cause an External/Status interrupt.

Bit 5 : Clear to Send

If the CTS IE bit in WR15 is set, this bit indicates the state of the $\overline{CTS}$ pin the last time any of the enabled External/Status bits changed. Any transition on the $\overline{CTS}$ pin while no interrupt is pending latches the state of the $\overline{CTS}$ pin and generates an External/Status interrupt. Any odd number of transitions on the $\overline{CTS}$ pin while another External/Status interrupt is pending also causes an External/Status interrupts condition. If the CTS IE bit is reset, it merely reports the current unlatched state of the $\overline{CTS}$ pin.

Bit 4 : SYNC/Hunt

The operation of this bit is similar to that of the $\overline{CTS}$ bit, except that the condition monitored by the bit varies depending on the mode in which the SCC is operating.

When the XTAL oscillator option is selected in asynchronous modes, this bit is forced to "0" (no External/Status interrupt is generated). Selecting the XTAL oscillator in synchronous or SDLC modes had no effect on the operation of this bit.

The XTAL oscillator should not be selected in External Sync mode.

In Asynchronous mode, the operation of this bit is identical to that of the CTS status bit, except that this bit reports the state of the SYNC pin.

In External sync mode the $\overline{SYNC}$ pin is used by external logic to signal character synchronization. When the Enter Hunt Mode command is issued in External Sync mode, the $\overline{SYNC}$ pin must be held High by the external sync logic until character synchronization is achieved. A High on the $\overline{SYNC}$ pin holds the Sync/Hunt bit in the reset condition.

When external synchronization is achieved, $\overline{SYNC}$ must be driven Low on the second rising edge of the Receive Clock after the last rising edge of the Receive Clock on which the last bit of the receive character was received. Only $\overline{SYNC}$ is forced Low, it is good practice to keep it Low until the CPU informs the external sync logic that synchronization has been lost or that a new message is about to start. Both transitions on the $\overline{SYNC}$ pin cause External/Status interrupts if the Sync/Hunt IE bit is set to "1".

The Enter Hunt Mode command should be issued

Figure 58 : Write Register 15.

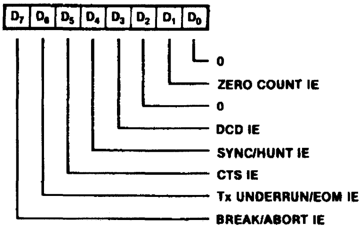
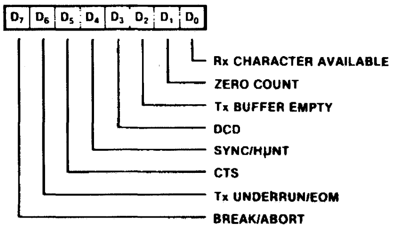


Figure 59 : Read Register 0.



REGISTERS DESCRIPTION (cont'd)

whenever character synchronization is lost. At the same time, the CPU should inform the external logic that character synchronization has been lost and that the SCC is waiting for $\overline{SYNC}$ to become active.

In the Monosync and Bisync Receive modes, the Sync/Hunt status bit is initially set to "1" by the Enter Hunt Mode command. The Sync/Hunt bit is reset when the SCC established character synchronization. Both transitions cause External/Status interrupts if the Sync/Hunt IE bit is set. When the CPU detects the end of message or the loss of character synchronization, the Enter Hunt Mode command should be issued to set the Sync/Hunt bit and cause an External/Status interrupt. In this mode, the $\overline{SYNC}$ pin is an output, which goes Low every time a sync pattern is detected in the data stream.

In the SDLC modes, the Sync/Hunt bit is initially set by the Enter Hunt Mode command or when the receiver is disabled. It is reset when the opening flag of the first frame is detected by the SCC. An External/Status interrupt is also generated if the Sync/Hunt IE bit is set. Unlike the Monosync and Bisync modes, once the Sync/Hunt bit is reset in SDLC mode, it does not need to be set when the end of the frame is detected. The SCC automatically maintains synchronization. The only way the Sync/Hunt bit can be set again is by the Enter Hunt Mode command or by disabling the receiver.

Bit 3 : Data Carrier Detect

If the DCD bit in WR15 is set, this bit indicates the state of the $\overline{DCD}$ pin the last time the Enabled External/Status bits changed. Any transition on the $\overline{DCD}$ pin while no interrupt is pending latches the state of the $\overline{DCD}$ pin, and generates an External/Status interrupt. Any odd number of transitions on the $\overline{DCD}$ pin while another External/Status interrupt is pending also causes an External/Status interrupt condition. If the DCD IE is reset, this bit merely reports the current, unlatched state of the $\overline{DCD}$ pin.

Bit 2 : TX Buffer Empty

This bit is set to "1" when the transmit buffer is empty. It is reset while CRC is sent in a synchronous or SDLC mode and while the transmit buffer is full. The bit is reset when a character is loaded into the transmit buffer. This bit is always in the set condition after a hardware or channel reset.

Bit 1 : Zero Count

If the Zero Count Interrupt Enable bit is set in WR15, this bit is set to one while the counter in the baud rate generator is at the count of zero. If there is no other External / Status interrupt condition pending at the time this bit is set, an External/Status interrupt is generated. However, if there is another External/Status interrupt pending at this time, no interrupt is initiated until interrupt service is complete. If the Zero Count condition does not persit beyond the end of the interrupt service routine, no interrupt will be generated. This bit is not latched High, even thought the other External/Status latches close as a result of the Low-to-High transition on ZC. The interrupt service routine should check the other External / Status conditions for changes. If none changed, ZC was the source. In polled applications, check the IP bit in RR3A for a status change and then proceed as in the interrupt service routine.

Bit 0 : RX Character Available

This bit is set to "1" when at least one character is available in the receive FIFO and is reset when the receive FIFO is complety empty. A channel or hardware reset empties the receive FIFO.

Read Register 1. RR1 contains the Special Receive Condition status bits and the residue codes for the I-field in SDLC mode. Figure 60 shows the bit positions for RR1.

Bit 7 : End of Frame (SDLC)

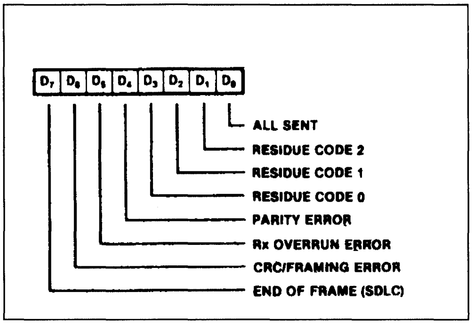
This bit is used only in SDLC mode and indicates that a valid closing flag has been received and that the CRC Error bit and residue codes are valid. This bit can be reset by issuing the Error Reset command. It is also updated by the first character of the following frame. This bit is reset in any mode other than SDLC.

Bit 6 : CRC/Framing Error

If a framing error occurs (in Asynchronous mode), this bit is set (and not latched) for the receive character in which the framing error occurred. Detection of a framing error adds an additional one-half bit to the character time so that the framing error is not interpreted as a new Start bit. In Synchronous and SDLC modes, this bit indicates the result of comparing the CRC checker to the apppropriate check value. This bit is reset by issuing an Error Reset command, but the bit is never latched. Therefore, it is always updated when the next character is received. When used for CRC error status in Synchronous or SDLC modes, this bit is usually set since most bit combination, except for a correctly completed message, result in a non-zero CRC.

REGISTERS DESCRIPTION (cont'd)

Figure 60 : Read Register 1.



Bit 5 : Receiver Overrun Error

This bit indicates that the receive FIFO has overflowed. Only the character that has been written over is flagged with this error, and when the character is read, the Error condition is latched until reset by the Error Reset command. The overrun character and all subsequent characters received until the Error Reset command is issued causes a Special Receive Condition vector to be returned.

Bit 4 : Parity Error

When parity is enabled, this bit is set for the characters whose parity does not match the programmed sense (even/odd). This bit is latched so that once an error occurs, it remains set until the Error Reset command is issued. If the parity in Special Condition bit is set, a parity error causes a Special Receive Condition vector to be returned on the character containing the error and on all subsequent characters until the Error Reset command is issued.

Bit 3,2, and 1 : Residue Codes 2,1, And 0

In those cases in SDLC mode where the received I-Field is not an integral multiple of the character length, these three bits indicate the length of the I-Field and are meaningful only for the transfer in which the end of frame bit is set. This field is set to "011" by a channel or hardware reset and is forced to this state in Asynchronous mode. These three bits can leave this state only if SDLC is selected and a character is received. The codes signify the following (Reference Table 16 when a receive character length is eight bits per character).

I-Field bits are right-justified in all cases. If a receive character length other than eight bits is used for the I-Field, a table similar to Table 16 can be constructed for each different character length. Table 17 shows the residue codes for no residue (The I-Field boundary lies on a character boundary).

Table 16 : I-Field Bit Selection (8 Bits only).

Residue	Residue	Residue	I-Field Bits in Last Byte	I-Field Bits in Previous Byte
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

Bit 0 : All Sent

In Asynchronous mode, this bit is set when all characters have completely cleared the transmitter pins. Most modems contain additional delays in the data path, which requires the modem control signals to remain active until after the data has cleared both the transmitter and the modem. This bit is always set in synchronous and SDLC modes.

Table 17 : Residue Bits/Character.

Bits / Char	Residue	Residue	Residue
8	0	1	1
7	0	0	0
6	0	1	0
5	0	0	1

Read Register 2. RR2 contains the interrupt vector written into WR2. When the register is accessed in Channel A, the vector returned is the vector actually stored in WR2. When this register is accessed in Channel B, the vector returned includes status information in bits 1, 2, and 3, or in bits 6, 5, and 4, depending on the state of the Status High/Status Low bit in WR9 and independent of the state of the VIS bit in WR9. The vector is modified according to Table 11 shown in the explanation of the VIS bit in WR9. If no interrupts are pending, the status is V3, V2, V1 = 011, or V6, V5, V4 = 110. Figure 61 shows the bit positions for RR2.

Read Register 3. RR3 is the Interrupts Pending register. The status of each of the Interrupt Pending bits in the SCC is reported in this register. This register exists only in Channel A. If this register is accessed in Channel B, all "0s" are returned. The two unused bits are always returned as "0". Figure 62 shows the bit positions for RR3.

REGISTERS DESCRIPTION (cont'd)

Read Register 8. RR8 is the Receive Data register.

Read Register 10. RR10 contains some miscellaneous status bits. Unused bits are always "0". Bit positions for RR10 are shown in Figure 63.

Bit 7 : One Clock Missing

While operating in the FM mode, the DPLL sets this bit to "1" when it does not see a clock edge on the incoming lines in the window where it expects one. This bit is latched until reset by a Reset Missing Clock or Enter Search Mode command in WR14. In the NRZI mode of operation and while the DPLL is disabled, this bit is always "0".

Bit 6 : Two Clocks Missing

While operating in the FM mode, the DPLL sets this bit to "1" when it does not see a clock edge in two successive tries. At the same time the DPLL enters the Search mode. This bit is latched until reset by a Reset Missing Clock or Enter Search Mode command in WR10. In the NRZI mode of operation and while the DPLL is disabled, this bit is always "0".

Bit 4 : Loop Sending

This bit is set to "1" in SDLC Loop mode while the transmitter is in control of the Loop, that is, while the SCC is actively transmitting on the loop. This bit is reset at all other times.

This bit can be polled in SDLC mode to determin when the closing flag has been sent.

Bit 1 : On Loop

This bit is set to "1" while the SCC is actually onloop in SDLC Loop mode. This bit is set to "1" in the X.21 mode (Loop mode selected while in monosync) when the transmitter goes active. This bit is "0" at all other times. This bit can also be pulled in SDLC mode to determine when the closing flag hass been sent.

Read Register 12. RR12 returns the value stored in WR12, the lower byte of the time constant for the baud rate generator. Figure 64 shows the bit positions for RR12

Read Register 13. RR13 returns the value stored in WR13, the upper byte of the time constant for the baud rate generator. Figure 65 shows the bit positions for RR13.

Read Register 15. RR15 reflects the value stored in WR15, the External/Status IE bits. The two unused bits are always returned as "0s". Figure 66 shows the bits positions for RR15.

Figure 61 : Write Register 2.

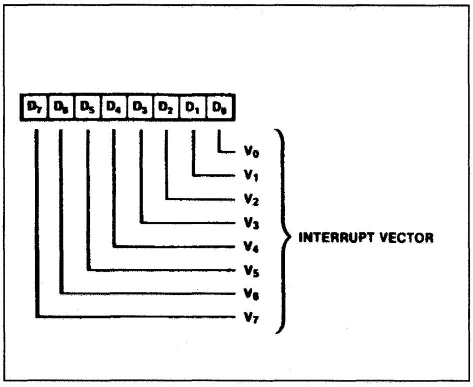
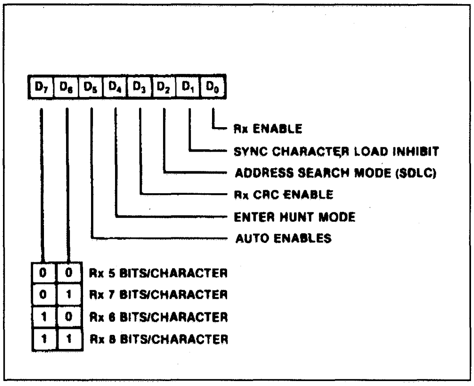


Figure 62 : Write Register 3.



REGISTERS DESCRIPTION (cont'd)

Figure 63 : Write Register 10.

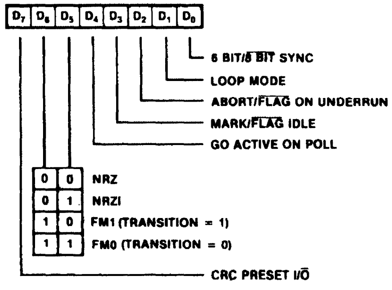


Figure 65 : Write Register 13.

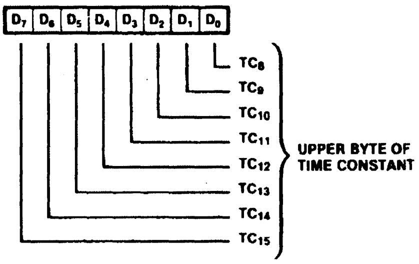


Figure 64 : Write Register 12.

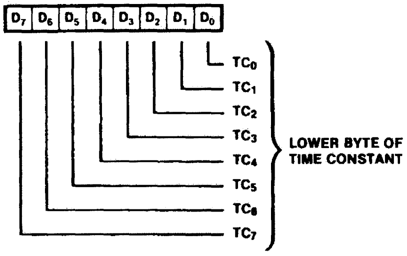
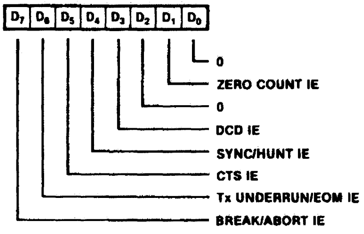


Figure 66 : Write Register 15.



SCC INITIALIZATION WORKSHEET

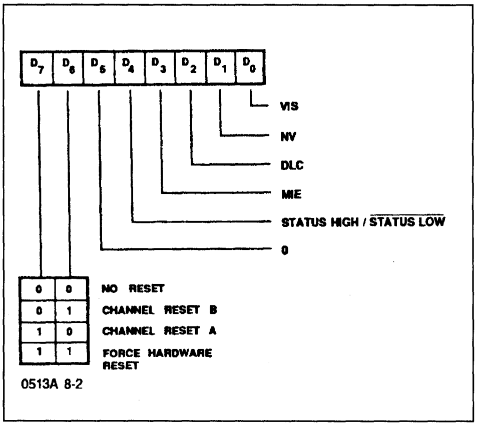
This section describes the software initialization procedure for the Serial Communications Controller (SCC).

Figure 68 provides a worksheet that can be used as an aid when initializing the SCC. Since all SCC operation modes are initialized in a similar manner, the worksheet can be used to tailor the SCC device to the user's individual need. Specific examples are given in the following sections.

Register Overview

Each of the SCC's two channels has its own separate Write registers that are programmed to initialize different operating modes. There are two types of bits in the Write registers : Command bits and Mode bits. An example of a register that contains both types of bits is Write Register 9 (WR9), and is shown in figure 67.

Figure 67 : Write Register 2.



WR9 is the Master Interrupt Control register and contains the Reset command bits. Command bits are denoted by having boxes drawn around them in register diagrams. Bit D5 in this register is not used in this register and must be 0 at all times.

The Command bits, D7 and D6, select one of the reset commands for the SCC. Setting either of these bits to 1 disables both the receiver and the transmitter in the corresponding channel, forces TxD for the channel marking, forces the modem control signals High in that channel, resets all IPs and IUSs, and disables all interrupts in that channel. Functions controlled by the Command bits can only be enabled or disabled, they cannot be toggled.

Bits D4-D0 are Mode bits that can be enabled or disabled either by being set to "1" or reset to "0". Each Mode bit affects only one function. For example, Bit D1 is the No Vector mode bit, it controls whether or not the SCC will respond to an interrupt acknowledge cycle by placing a vector on the data bus. If this bit is set, no vector is returned. In command bits entry, each new command requires a separate rewrite of the entire register. Care must be taken when issuing a command, so that the Mode bits are not changed accidentally.

Initialization Procedure

The SCC initialization procedure is divided into three parts. The first part consists of programming the operation modes (e.g. bits-per-character, parity) and loading the constants (e.g., interrupt vector, time constants). The second part enables the hardware functions (e.g., transmitter, receiver, baud-rate generator). It is important that the operating modes are programmed before the hardware functions are enabled. The third part, if required, consists of enabling the different interrupts.

Table 18 shows the order (from top to bottom) in which the SCC registers are to be programmed. Those registers that need not be programmed are listed as optional in the comments column. The bits in the registers that are marked with an "X" are to be programmed by the user. The bits marked with an "S" are to be set to their previous programmed value. For example, in part 2, Write Register 3, bits D1-D7 are shown with an "S" because they have been programmed in part 1 and must remain set to the same value.

Initialization Table Generation

Figure 68 is a worksheet for the initialization of the SCC. All the bits that must be programmed as either a "0" or a "1" are already filled in ; the remaining bits are left blank and are to be programmed by the user according to the desired mode of operation. The binary value can then be converted to a hexadecimal number and placed in the table, following the Write register notation in the column labeled "HEX". A Program Initialization Table is produced when this worksheet is completed.

SCC INITIALIZATION WORKSHEET (cont'd)

Figure 68 : SCC Initialization Worksheet.

Label of SCC Table: ____________________ SCC Base Address ____________________

Description: ____

	REGISTER	HEX		BINARY 7	6	5	4	3	2	1	0	COMMENTS
MODES	WR9	C	0	1	1	0	0	0	0	0	0	SOFTWARE RESET
	WR0	0		0	0	0	0	0	0			
	WR4											
	WR1			0			0	0		0	0	
	WR2											
	WR3										0	
	WR5							0				
	WR6											
	WR7											
	WR9			0	0	0		0				
	WR10											
	WR11											
	WR12											
	WR13											
	WR14										0	
	WR14										0	
ENABLES	WR14			0	0	0					1	
	WR3										1	
	WR5							1				
	WR0	8	0	1	0	0	0	0	0	1	0	RESET TxCRC
	WR1											
INTERRUPT	WR15											
	WR0	1	0	0	0	0	1	0	0	0	0	RESET Ext/STATUS
	WR0	1	0	0	0	0	1	0	0	0	0	RESET Ext/STATUS
	WR1											
	WR9			0	0	0						

SCC INITIALIZATION WORKSHEET (cont'd)

Table 18 : SCC Initialization Order.

Part 1. Modes and Constants		
WR9	1100000	Hardware Reset
WR0	000000XX	Select Shift Mode (8030 only)
WR4	XXXXXXXX	Tx/Rx Con, Async or Sync Mode
WR1	0XX00X00	Select W/REQ (opt)
WR2	XXXXXXXX	Program Interrupt Vector (opt)
WR3	XXXXXXX0	Select Rx Control
WR5	XXXX0XXX	Selects Tx Control
WR6	XXXXXXXX	Program Sync Character (opt)
WR7	XXXXXXXX	Program Sync Character (opt)
WR9	000X0XXX	Select Interrupt Control
WR10	XXXXXXXX	Miscellaneous Control (opt)
WR11	XXXXXXXX	Clock Control
WR12	XXXXXXXX	Time Constant Lower Byte (opt)
WR13	XXXXXXXX	Time Constant Upper Byte (opt)
WR14	XXXXXXX0	Miscellaneous Control
WR14	XXXSSSSS	Commands (opt)

Part 2. Enables		
WR14	000SSSS1	Baud Rate Enable
WR3	SSSSSSS1	Rx Enable
WR5	SSSS1SSS	Tx Enable
WR0	10000000	Reset Tx CRG (opt)
WR1	XSS00S00	DMA Enable (opt)
Part 3. Interrupt Status		
WR15	XXXXXXXX	Enable External/status
WR0	00010000	Reset External Status
WR0	00010000	Reset External Status Twice
WR1	SSSXXSXX	Enable Rx, Tx and Ext/status
WR9	000SXSSS	Enable Master Interrupt Enable
1 = Set to one 0 = Reset to zero		X = User defined S = Same as previously prog.

Reset Conditions

Prior the initialization, the SCC should be reset by either hardware or software. A hardware reset can be accomplished by simultaneously grounding. A software reset can be executed by writing a 0H to Write Register 9.

POLLED ASYNCHRONOUS MODE

This section describes the use of the SCC in polled Asynchronous Mode. The device can be set with 5 to 8 bits per character, 1, 1.5, or 2 stop bits, and a wide range of baud rates. In this particular example, 8 bits per character, 2 stop bits and 9600 baud rate are used. An external 2.4576MHz, crystal oscillator is used for baud-rate generation. The SCC can be programmed for local loopback for on-board diagnostics. The user can make use of this feature to test-program the part without additional hardware to simiulate an actual transmit and receive environment.

SCC Interface

Figure 69 shows the SCC to CPU interface required for this application. The 8-bit data bus and control lines all come from the user's CPU. The 8530 control lines are $\overline{RD}$, $\overline{WR}$, A/$\overline{B}$, D/$\overline{C}$ and $\overline{CE}$. PCLK comes from the system clock, or an external crystal, up to the maximum rate of the SCC (ex. 6MHz for the Z8530A). The IEI and the $\overline{INTACK}$ pins should be pulled up. The baud-rate generator clock is connected to the $\overline{RTxC}$ pin.

SCC Initialization

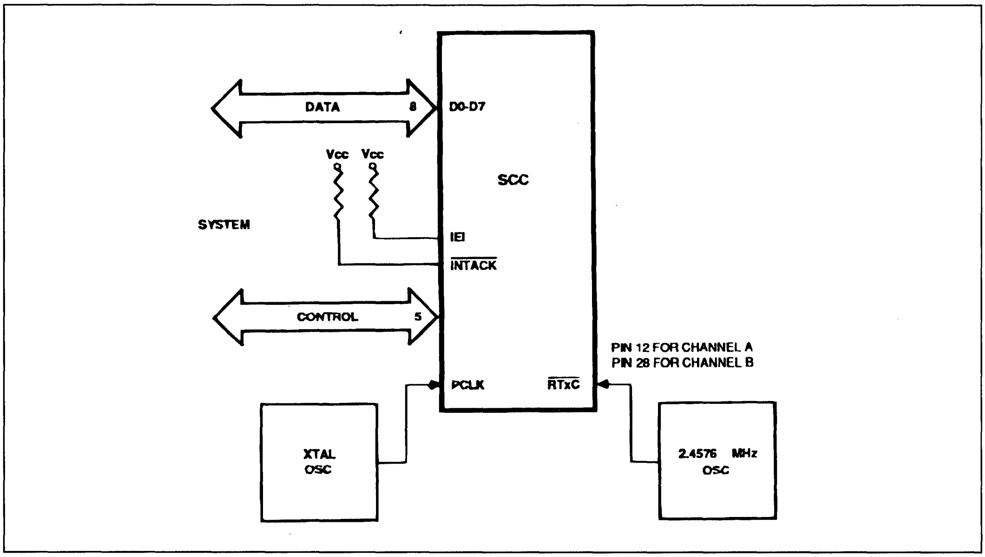
Initialization of the SCC for polled asynchronous communication is divided into two parts ; part one programs the operating modes of the SCC and part two enables them. Care must be taken when writing the software to meet the SCC's Cycle and Reset Recovery times. The Cycle Recovery time, 6 PCLK cycles, applies to the period between any Read or Write cycles affecting the SCC. The Reset Recovery time is the period after a hardware reset caused either by hardware or software ; this recovery time extends the Cycle Recovery time to 11 PCLK cycles. For more details about these recovery times, see the section Interfacing the SCC.

Table 19 : Polled Asynchronous Initialization Procedure.

Register	Value	Comments
WR9	C0H	Force Hardware Reset
WR4	4CH	x16 Clock, 2 Stop Bits, no Parity
WR3	C0H	Rx8 Bits, Rx Disabled
WR5	60H	Tx8 Bits, DTR, RTS, Tx off
WR9	00H	Int. Disabled
WR10	00H	NRZ
WR11	56H	Tx & Rx = BRG out, TRxC = BRG out
WR12	06H	Time Constant = 6
WR13	00H	Time Constant High = 0
WR14	10H	BRG in = RTxC, BRG off, Loopback
Enables		
WR14	11H	BRG Enable
WR3	C1H	Rx Enable
WR5	68H	Tx Enable

POLLED ASYNCHRONOUS MODE (cont'd)

Figure 69.



SCC Operating Mode Programming

WR9 resets the SCC to a known state by writing a C0 hex. The command, Force Hardware Reset, is identical to a hardware reset.

WR4 selects the asynchronous, x 16 mode, with 2 stop bits and no parity. The x 16 mode means that clock rate is 16 times the data rate.

WR3 selects 8 bits per character and does not enable the receiver. The 8 bits per character allows 8 bits to be assembled from the data stream. The receiver is not enabled at this time because the SCC has not been initialized.

WR5 selects 8 bits per character and does not enable the transmitter. The 8 bits per character allows 8 bits to be sent, as data, with the least significant bit first. The transmitter is not enabled at this time because the SCC has not been initialized.

WR9 selects that there are no interrupts enabled. This inhibits the SCC from requesting an interrupt from the CPU.

WR10 selects NRZ encoding. This NRZ coding is used on the transmitter as well as the receiver.

WR11 selects the $\overline{\text{RTxC}}$ pin to TTL clock ; the baud-rate generator is the transmit and receive clocks source, and the TRxC pin as a baud-rate generator output.

WR12 & WR13 and select the baud-rate generator's time constant. The WR13 time constant is determined by the equation :

$$\text{Time Constant} = \frac{\text{Clock Frequency}}{2 \times \text{Baud Rate} \times \text{clock mode}} - 2$$

In this example, the clock frequency is 2.4576MHz, the baud rate is 9600, and the clock mode is 16, the time constant is, therefore, 6 ; expressed as a 16-bit, hexadecimal number, it is 0006H. The time constant LOW (WR12) is, therefore, 06H and the time constant HIGH (WR13) is 00H. The baud rate for this example can be varied, as long as the data rate is less than 1/4 of the PCLK rate.
Table 20 shows the time constants for other common baud rates.

Table 20 : Time Constants for Common Baud Rates.

Baud Rate	Divider	
	Decimal	Hex
38400	0	0000H
19200	2	0002H
9600	6	0006H
4800	14	000EH
2400	30	001EH
1200	62	003EH
600	126	007EH
300	254	00FEH
150	510	01FEH
For 2.4576 MHz Clock, X 16 Mode		

POLLED ASYNCHRONOUS MODE (cont'd)

WR14 selects the baud-rate generator as the $\overline{RTxC}$ pin, baud-rate generator disabled, and internal loop-back. The baud-rate generator uses the $\overline{RTxC}$ pin as the clock source and is not enabled at this time because the SCC initialization is not complete.

SCC Operating Mode Enables

WR14 enables the baud-rate generator. Bit 0 (LSB) is changed to a 1 to enable the baud-rate generator; all other bits must maintain the value selected during initialization.

WR3 enables the receiver. Bit 0 (LSB) is changed to a 1 to enable the receiver, all other bits must maintain the value selected during initialization.

WR5 enables the transmitter. Bit 3 is changed to a 1 to enable the transmitter, all other bits must maintain the value selected during initialization.

Transmit and Receive Routines

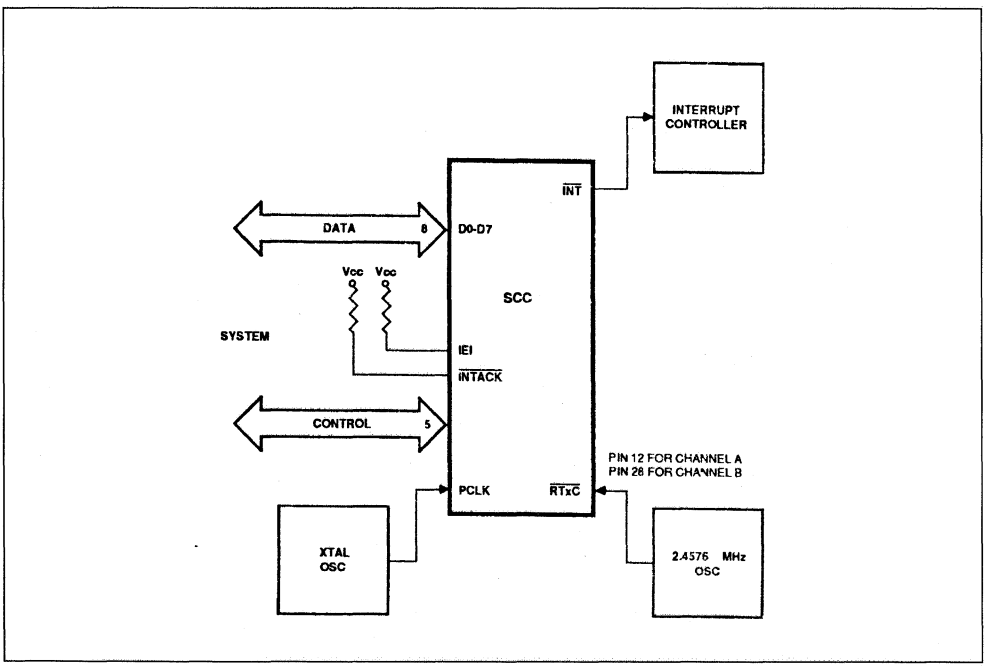
After initialization, and after all enables have been selected, the SCC is ready for communication.

The transmitter buffer and the receive FIFO are empty. The example shown below is coded to transmit and receive characters.

Figure 70 : Transmit and Receive Routine.

```
; Transmit a character
TXCHAR : INPUT    RRO      ;Read RRO
         Test     Bit 2    ;Test transmit
                           buffer empty
         JZ       TXCHAR   ;Loop if not
                           empty
         OUTPUT   CHAR     ;Output chara
                           cter to data port
         RET               Return
; Receive a character
RXCHAR : INPUT    RRO      ;Read RRO
         TEST     BIT 0    ;Test Receive
                           buffer
         JZ       RXCHAR   ;Loop if not
                            full
         INPUT    CHAR     ;Input charac
                           ter from data
                            port
         RET               ; Return
```

Figure 71.



INTERRUPT WITHOUT INTACK ASYNCHRONOUS MODE

This section describes the use of the SCC for interrupt-driven Asynchronous Mode. As with the example in the previous chapter, the SCC is set with 8 bits per character, 2 stop bits, at 9600 baud rate. An external 2.4576MHz, crystal oscillator is used for baud-rate generation. Interrupt acknowledge is not generated because of the extra hardware required to produce this signal. In this chapter, the SCC is also programmed for local loopback so that no external loop between the transmit and the receive data lines is needed for on-board diagnostics. This feature allows the user to test-program the part without additional hardware to simiulate an actual transmit and receive environment.

SCC Interface

Figure 71 shows the SCC to CPU interface required for this application. The 8-bit data bus and control lines all come from the user's CPU. For the 8530, the control lines are $\overline{RD}$, $\overline{WR}$, A/$\overline{B}$, D/$\overline{C}$ and $\overline{CE}$. The $\overline{INT}$ signal goes to an interrupt controller which must produce the interrupt vector to the CPU. The PCLK comes from the system clock, or an external crystal oscillator, up to the maximum rate of the SCC (ex. 6MHz for the Z8530A). The IEI and the $\overline{INTACK}$ pins should be pulled up. The baud-rate generator clock is connected to the $\overline{RTxC}$ pin.

SCC Initialization

The initialization of the SCC for interrupt-driven asynchronous communication is divided into three parts. Part one programs the operating modes of the SCC, part two and three enable them. Care must be taken when writing the code to meet the SCC's Cycle and Reset Recovery times. The Cycle Recovery time applies to the period between any Read or Write cycles to the SCC, and is 6PCLK cycles. The Reset Recovery time applies to a hardware reset caused either by hardware or software ; this recovery time extends the Cycle Recovery time to 11PCLK cycles. More details about these recovery times can be found in the section Interfacing the SCC.

Table 21.

Register	Value	Comments
WR9	C0H	Force Hardware Reset
WR4	4CH	x16 Clock, 2 Stop Bits, no Parity
WR2	00H	Interrupt Vector 00WR3
	C0H	Rx8 Bits, Rx Disabled
WR5	60H	Tx8 Bits, DTR, RTS, Tx off
WR9	00H	Int. Disabled WR10
	00H	NRZ
WR11	56H	Tx & Rx = BRG out, TRxC = BRG out
WR12	06H	Time Constant = 6
WR13	00H	Time Constant High = 0
WR14	10H	BRG in = RTxC, BRG off, Loopback
Enables		
WR14	11H	BRG Enable
WR3	C1H	Rx Enable
WR5	68H	Tx Enable
Enable Interrupts		
WR1	12H	Rx Int on All Char and Tx Int Enables
WR9	08H	MIE

SCC Operating Modes Programming

WR9 resets the SCC to a known state by writing a C0 hex. This command, Force Hardware Reset, is identical to a hardware reset.

WR4 selects asynchronous mode, x16 mode, 2 stop bits and no parity. The x16 mode means that the clock rate is 16 times the data rate.

WR2 is the interrupt vector of the SCC. Even though a vector is not placed on the bus in this mode the vector including status in read from RR2. By writing 00H to this register the status read will be the only bits set in RR2.

WR3 selects 8 bits per character and does not enable the receiver. The 8 bits per character allows 8 bits to be assembled from the data stream. The receiver is not enabled at this time because the SCC is not completely initialized.

WR5 selects 8 bits per character and does not

INTERRUPT WITHOUT INTACK ASYNCHRONOUS MODE (cont'd)

enable the transmitter. The 8 bits per character allows 8 bits to be sent as data with the least significant bit first. The transmitter is not enabled at this time because the SCC is not completely initialized.

WR9 selects that there are no interrupts enabled. This will inhibit the SCC from requesting an interrupt from the CPU.

WR10 selects NRZ encoding. This selects NRZ coding is to be used on the transmitter and the receiver.

WR11 selects the RTxC pin to TTL clock, the transmit and receive clocks source as the baud-rate generator and the TRxC pin as a baud-rate generator output.

WR12 & WR13 select the baud-rate generators time constant. The time constant is determined by the equation :

$$\text{Time Constant} = \frac{\text{Clock Frequency}}{2 \times \text{Baud Rate} \times \text{clock mode}} - 2$$

In this example, the clock frequency is 2.4576MHz, the baud rate is 9600, and the clock mode is 16 ; the time constant is 6. Converting this time constant to a 16-bit hexadecimal number, it becomes 0006H. The time constant LOW (WR12) is 06H and the time constant HIGH (WR13) is 00H. The baud rate for this example can be varied for as long as the data rate is less than 1/4 of the PCLK rate. Table 22 gives the time constants for other common baud rates.

Table 22 : Time Constants for Common Baud Rates.

Baud Rate	Divider	
	Decimal	Hex
38400	0	0000H
19200	2	0002H
9600	6	0006H
4800	14	000EH
2400	30	001EH
1200	62	003EH
600	126	007EH
300	254	00FEH
150	510	01FEH
For 2.4576 MHz Clock, X 16 Mode		

WR14 selects the baud rate source as the $\overline{\text{RTxC}}$ pin, baud rate generator disabled, and internal loopback. The baud-rate generator will use the $\overline{\text{RTxC}}$ pin as the clock source for the baud-rate generator. The baud-rate generator is not enabled at this time because the SCC initialization is not complete.

SCC Operating Mode Enables

WR14 enables the baud-rate generator. Bit 0 (LSB) is changed to a 1 to enable the baud-rate generator ; all other bits must maintain the value selected during initialization.

WR3 enables the receiver. Bit 0 (LSB) is changed to a 1 to enable the receiver ; all other bits must maintain the value selected during initialization.

WR5 enables the transmitter. Bit 3 is changed to a 1 to enable the transmitter ; all other bits must maintain the value selected during initialization.

SCC Operating Mode Interrupts

WR1 enables the Tx and the Rx interrupts. The Rx interrupt is programmed to generate an interrupt an all received characters or special conditions. This provides an interrupt on every character received by the SCC. The external/status interrupts are not enabled in this application.

WR9 sets the master interrupt enable (MIE) bit 3. Seting this bit enables the interrupts pending to generate and interupt on the INT pin.

Interrupt Routine

When the SCC has been initialized and enabled, it is ready for communication. The transmitter buffer and the receive FIFO are both empty. An interrupt will not be generated until the software writes the first character to the transmit buffer. Once the first character is in the SCC shift register, the first transmit interrupt will occur. The SCC will then keep setting transmit and receive interrupts to the interrupt controller until the end of the message. At the end of the message, a Reset Transmitter Interrupt Pending (WR0) is issued to clear the transmit interrupt. After the last character is read into the SCC, the interrupts will cease until another message is written into the transmitter.

Once an interrupt is received and the interrupt controller vectors to the interrupt routine, RR2 is read from channel B. The value read from RR2 is the vector, including status. This vector shows the status of the highest priority interrupt pending (IP) at the time it is read. Once the highest priority interrupt condition is cleared, RR2 will show the status of the next highest interrupt pending, if one is present. This allows multiple interrupts to be serviced without the overhead of the interrupt acknowledge cycle of the interrupt controller.

The following example shows how the interrupt routine should be coded.

HARDWARE INFORMATION

Absolute Maximum Ratings

Parameter	Test Conditions	Unit
Operating Temperature Z8530 B1/D1/C1 B6/D6/C6	0 to + 70 – 40 to + 85	°C °C
Storage Temperature	– 65 to + 150	°C
Voltages on any Pin with Respect to GND	– 0.3 to + 7	V
Total Power Dissipation	700	mW
Oscillator Frequency Z8530 Z8530A Z8530B	0 to 4 0 to 6 0 to 8	MHz

Note : 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 conditions above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin.

Standard conditions are as follows :
- $+4.75\ V \le V_{CC} \le +5.25\ V$
- GND = 0 V
- $0\ °C \le T_A \le +70\ °C$

All ac parameters assume a load capacitance of 50 pf max.

Figure 73 : Standard Test Load.

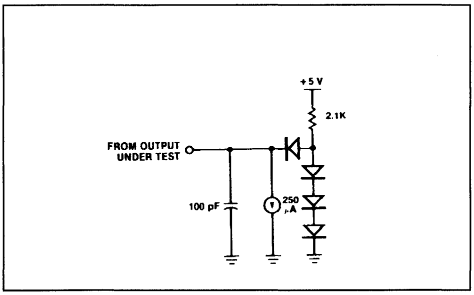
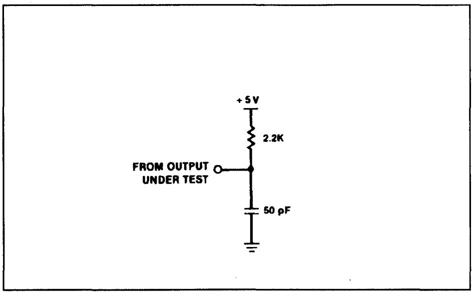


Figure 74 : Open-Drain Test Load.



DC Characteristics

Symbol	Parameter	Test Conditions	Min.	Max.	Unit
V_{IH}	Input High Voltage		2.0	V_{CC}+0.3	V
V_{IL}	Input Low Voltage		– 0.3	0.8	V
V_{OH}	Output High Voltage	I_{OH} = – 250 µA	2.4		V
V_{OL}	Output Low Voltage	I_{OL} = + 2.0 mA		0.4	V
I_{IL}	Input Leakage	$0.4 \le V_{IN} \le +2.4$ V		± 10.0	µA
I_{OL}	Output Leakage	$0.4 \le V_{OUT} \le +2.4$ V		± 10.0	µA
I_{CC}	V_{CC} Supply Current		250		mA

V_{CC} = 5 V ± 5 % unless otherwise specified, over specified temperature range.

HARDWARE INFORMATION (cont'd)

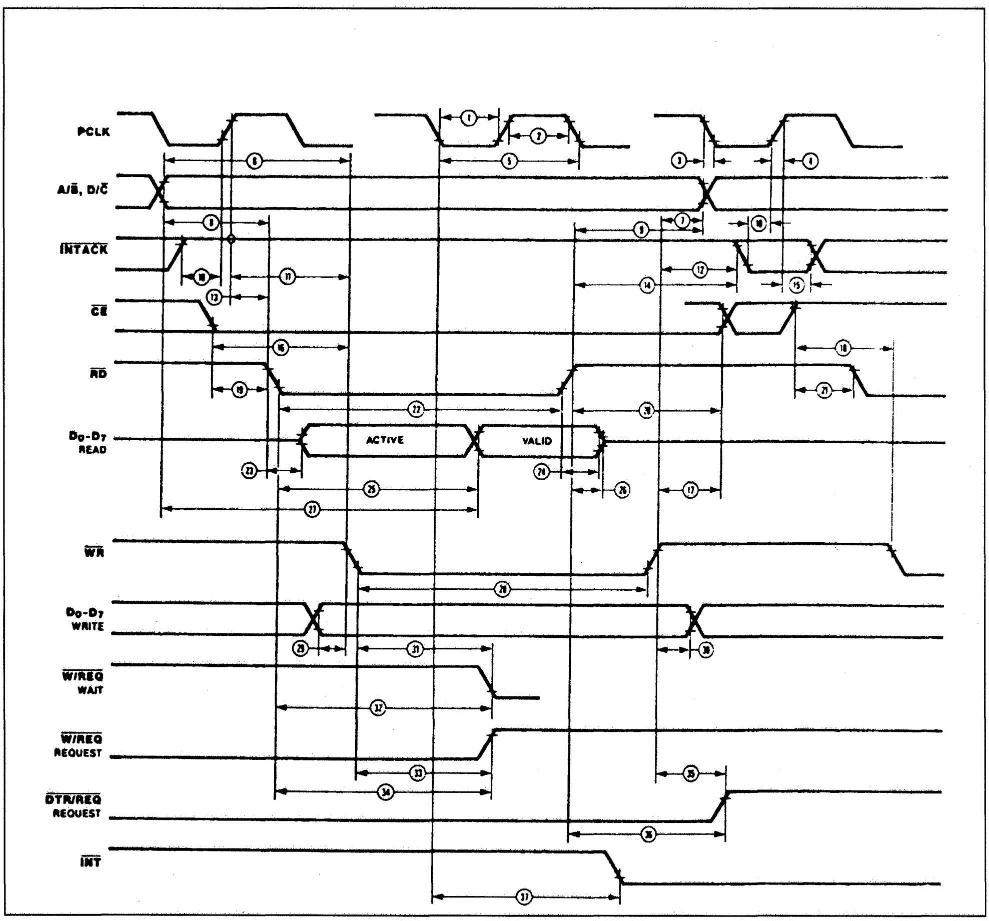
Capacitance

Symbol	Parameter	Test Conditions	Min.	Max.	Unit
C_{IN}	Input Capacitance			10	pf
C_{OUT}	Output Capacitance			15	pf
$C_{I/O}$	Bidirectional Capacitance			20	pf

f = 1 MHz, over specified temperature range.
Unmeasured pins returned to ground.

AC Timing Characteristics

Figure 75 : Read and Write Timing.



SGS-THOMSON MICROELECTRONICS

HARDWARE INFORMATION (cont'd)

Table 23 : AC Timing Characteristics.

N°	Symbol	Parameter	Z8530		Z8530A		Z8530B		Notes*
			Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
1	TwPC1	PCLK Low Width	105	2000	70	1000	50	1000	
2	TwPCh	PCLK High Width	105	2000	70	1000	50	1000	
3	TfPC	PCLK Fall Time		20		10		10	
4	TrPC	PCLK Rise Time		20		15		10	
5	TcPC	PCLK Cycle Time	250	4000	165	2000	125	2000	
6	TsA(WR)	Address to $\overline{WR}$ ↓ Setup Time	80		80		70		
7	ThA(WR)	Address to $\overline{WR}$ ↑ Hold Time	0		0		0		
8	TsA(RD)	Address to $\overline{RD}$ ↓ Setup Time	80		80		70		
9	ThA(RD)	Address to $\overline{RD}$ ↑ Hold Time	0		0		0		
10	TsIA(PC)	$\overline{INTACK}$ to PCLK ↑ Setup Time	10		10		10		
11	TsIAi(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↓ Setup Time	200		160		145		1
12	ThIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↑ Hold Time	0		0		0		
13	TsIAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ Setup Time	200		160		145		1
14	ThIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↑ Hold Time	0		0		0		
15	ThIA(PC)	$\overline{INTACK}$ to PCLK ↑ Hold Time	100		100		85		
16	TsCE1(WR)	$\overline{CE}$ Low to $\overline{WR}$ ↓ Setup Time	0		0		0		
17	ThCE(WR)	$\overline{CE}$ to $\overline{WR}$ ↑ Hold Time	0		0		0		
18	TsCEh(WR)	$\overline{CE}$ High to $\overline{WR}$ ↓ Setup Time	100		70		60		
19	TsCE1(RD)	$\overline{CE}$ Low to $\overline{RD}$ ↓ Setup Time	0		0		0		1
20	ThCE(RD)	$\overline{CE}$ to $\overline{RD}$ ↑ Hold Time	0		0		0		1
21	TsCEh(RD)	$\overline{CE}$ High to $\overline{RD}$ ↓ Setup Time	100		70		60		1
22	TwRD1	$\overline{RD}$ Low Width	240		200		150		1
23	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		0		0		
24	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		0		0		
25	TdRDf(DR)	$\overline{RD}$ ↓ to Read Data Valid Delay		250		180		140	
26	TdRD(DRz)	$\overline{RD}$ ↑ to Read Data Float Delay		70		45		40	2

Notes : 1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is defined as the time required for a ± 0.5 V change in the output with a maximum dc load and minimum ac load.
* Timings are preliminary and subject to change.

HARDWARE INFORMATION (cont'd)

Figure 76 : Interrupt Acknowledge Timing.

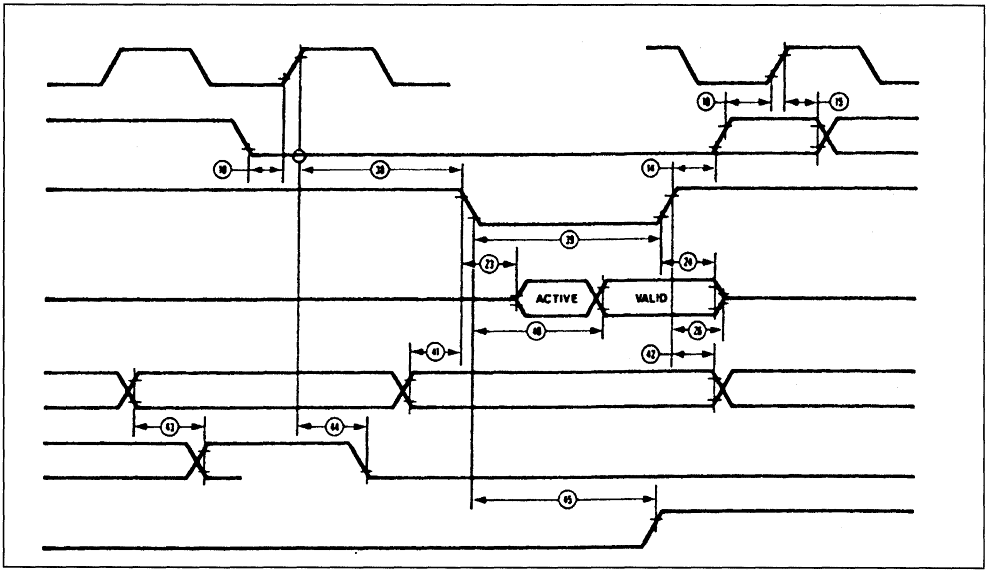


Figure 77 : Reset Timing.

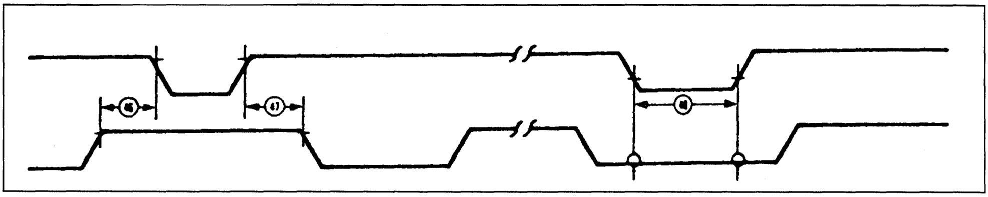
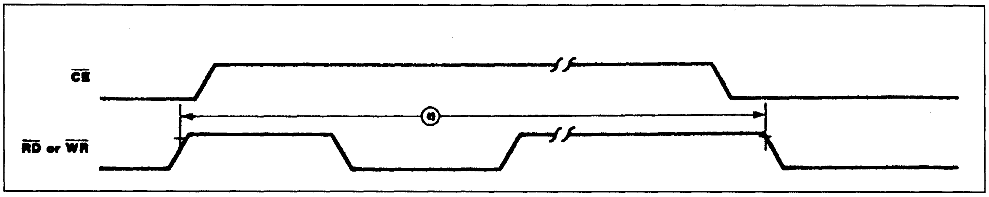


Figure 78 : Cycle Timing.



HARDWARE INFORMATION (cont'd)

Table 24 : AC Timing Characteristics.

N°	Symbol	Parameter	Z8530		Z8530A		Z8530B		Notes*
			Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
27	TdA(DR)	Address Required Valid to Read Data Valid Delay		300		280		220	
28	TwWR1	$\overline{WR}$ Low Width	240		200		150		
29	TsDW(WR)	Write Data to $\overline{WR}\downarrow$ Setup Time	10		10		10		
30	ThDW(WR)	Write Data $\overline{WR}\uparrow$ Hold Time	0		0		0		
31	TdWR(W)	$\overline{WR}\downarrow$ to Wait Valid Delay		240		200		170	4
32	TdRD(W)	$\overline{RD}\downarrow$ to Wait Valid Delay		240		200		170	4
33	TdWRf(REQ)	$\overline{WR}\downarrow$ to $\overline{W}$ / $\overline{REQ}$ Not Valid Delay		240		200		170	
34	TdRDf(REQ)	$\overline{RD}\downarrow$ to $\overline{W}$ / $\overline{REQ}$ Not Valid Delay		240		200		170	
35	TdWRr(REQ)	$\overline{WR}\uparrow$ to $\overline{DTR}$ / $\overline{REQ}$ Not Valid Delay		4TcPC		4TcPC		4TcPC	
36	TdRDr(REQ)	$\overline{RD}\uparrow$ to $\overline{DTR}$ / $\overline{REQ}$ Not Valid Delay		4TcPC		4TcPC		4TcPC	
37	TdPC(INT)	PCLK $\downarrow$ to $\overline{INT}$ Valid Delay		500		500		500	4
38	TdIAi(RD)	$\overline{INTACK}$ to $\overline{RD}\downarrow$ (Acknowledge) Delay	250		200		150		5
39	TwRDA	$\overline{RD}$ (Acknowledge) Width	250		200		150		
40	TdRDA(DR)	$\overline{RD}\downarrow$ (Acknowledge) to Read Data Valid Delay		250		180		140	
41	TsIEI(RDA)	IEI to $\overline{RD}\downarrow$ (Acknowledge) Setup Time	120		100		95		
42	ThIEI(RDA)	IEI to $\overline{RD}\uparrow$ (Acknowledge) Hold Time	0		0		0		
43	TdIEI(IEO)	IEI to IEO Delay Time		120		100		95	
44	TdPC(IEO)	PCLK $\uparrow$ to IEO Delay		250		250		200	
45	TdRDA(INT)	$\overline{RD}\downarrow$ to $\overline{INT}$ Inactive Delay		500		500		450	4
46	TdRD(WRQ)	$\overline{RD}\uparrow$ $\overline{WR}\downarrow$ Delay for No Reset	30		15		15		
47	TdWRQ(RD)	$\overline{WR}\uparrow$ to $\overline{RD}\downarrow$ Delay for No Reset	30		30		20		
48	TwRES	$\overline{WR}$ and $\overline{RD}$ Coincident Low for Reset	250		200		150		
49	Trc	Valid Access Recovery Time	4TcPC		4TcPC		4TcPC		3

Notes : 3. Parameter applies only between transactions involving the SCC.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any SCC in the daisy chain, TdIAi(RD) must be greater than the num of TdPC(IEO) for the highest priority device in the daisy chain, TsIEI(RDA) for the SCC, and TdIEIf(IEO) for each device separating them in the daisy chain.
* Timings are preliminary and subject to change.

HARDWARE INFORMATION (cont'd)

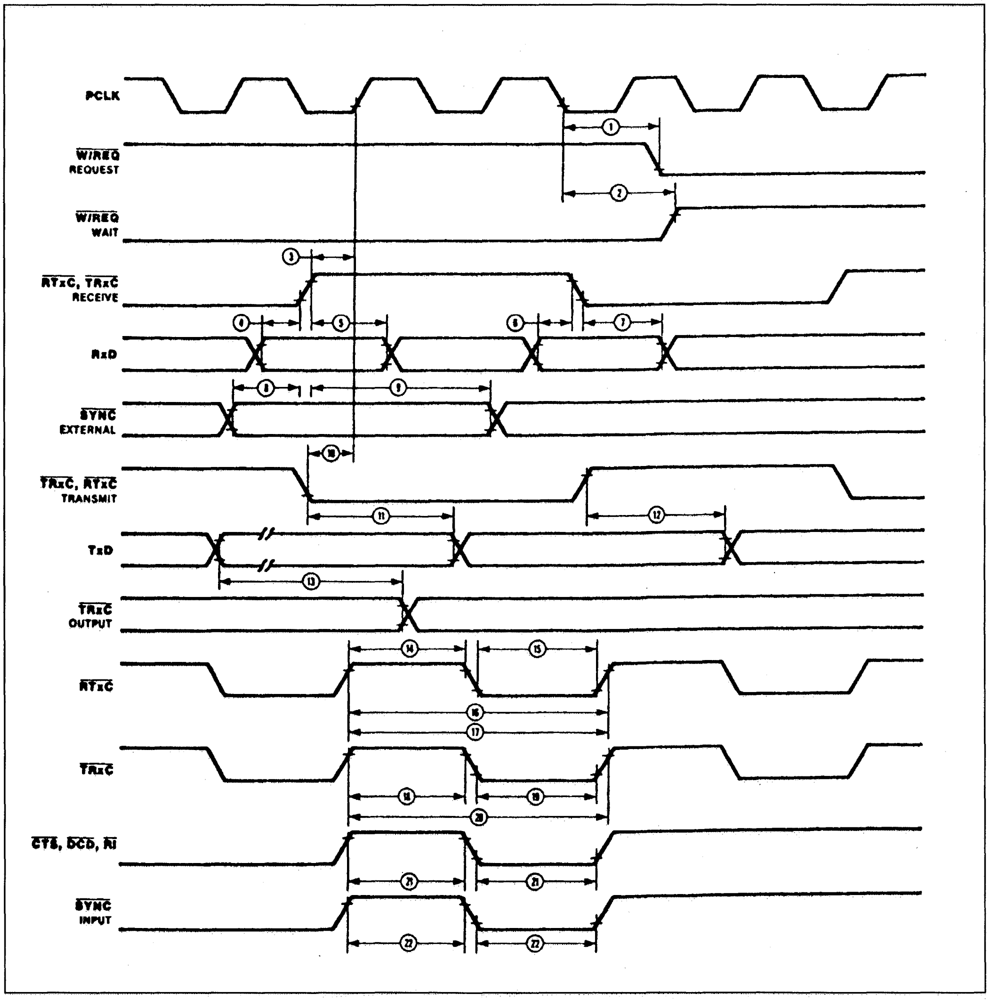


Figure 79 : General Timing.

HARDWARE INFORMATION (cont'd)

Table 25 : General Timing Characteristics (cont'd).

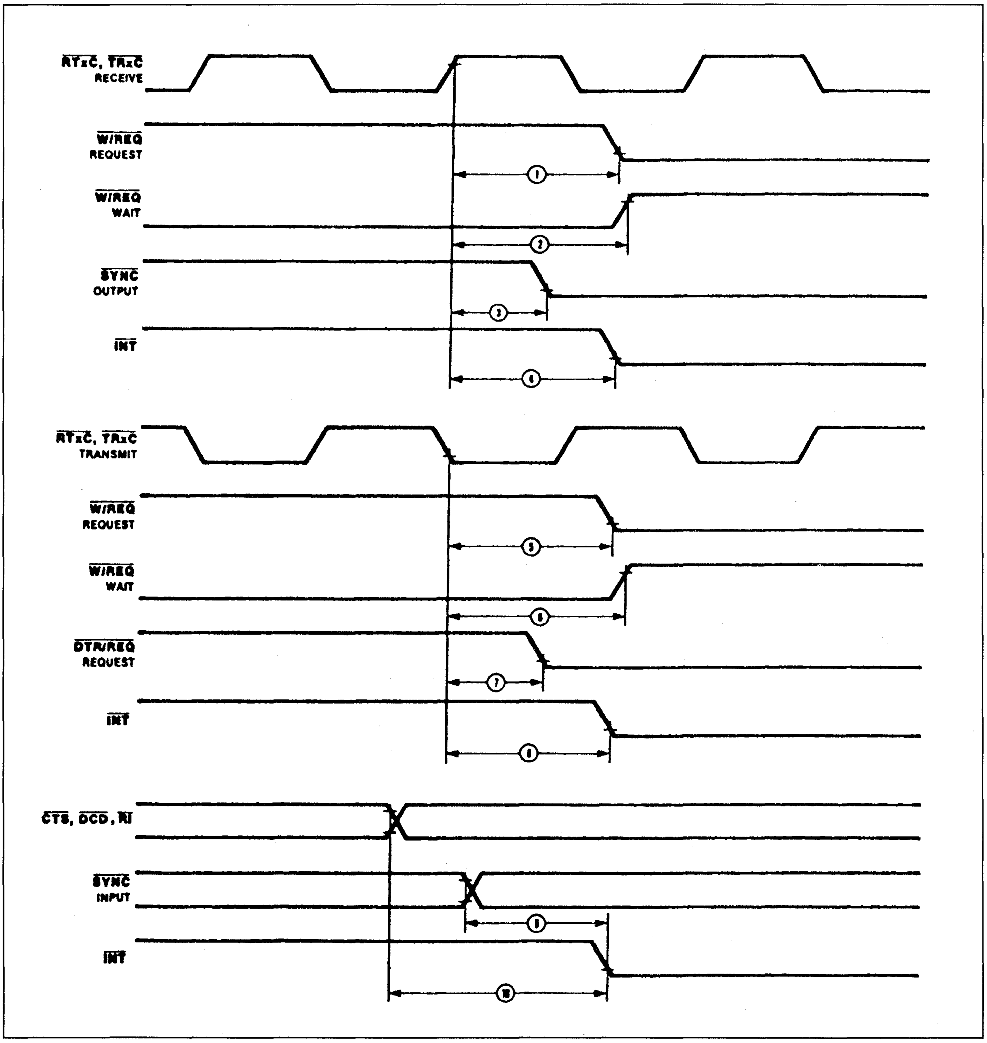
N°	Symbol	Parameter	Z8530		Z8530A		Z8530B		Notes*
			Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
1	TdPC(REQ)	PCLK $\downarrow$ to $\overline{W}$ / $\overline{REQ}$ Valid Delay		250		250		250	
2	TdPC(W)	PCLK $\downarrow$ to Wait Inactive Delay		350		350		350	
3	TsRXC(PC)	$\overline{RxC}$ $\uparrow$ to PCLK $\uparrow$ Setup Time (PCLK ÷ 4 case only)	80	TwPCL	70	TwPCL	60	TwPCL	1.4
4	TsRXD(RXCr)	RxD to $\overline{RxC}$ $\uparrow$ Setup Time (X1 Mode)	0		0		0		1
5	ThRXD(RXCr)	RxD to $\overline{RxC}$ $\uparrow$ Hold Time (X1 Mode)	150		150		150		1
6	TsRXD(RXCf)	RxD to $\overline{RxC}$ $\downarrow$ Setup Time (X1 Mode)	0		0		0		1.5
7	ThRXD(RXCf)	RxD to $\overline{RxC}$ $\downarrow$ Hold Time (X1 Mode)	150		150		150		1.5
8	TsSY(RXC)	$\overline{SYNC}$ to $\overline{RxC}$ $\uparrow$ Setup Time	– 200		– 200		– 200		1
9	ThSY(RXC)	$\overline{SYNC}$ to $\overline{RxC}$ $\uparrow$ Hold Time	3TcPC + 400		3TcPC + 320		3TcPC + 250		1
10	TsTXC(PC)	$\overline{TxC}$ $\downarrow$ to PCLK $\uparrow$ Setup Time	0		0		0		2.4
11	TdTXCf(TXD)	$\overline{TxC}$ $\downarrow$ to TxD Delay (X1 Mode)		300		230		200	2
12	TdTXCr(TXD)	$\overline{TxC}$ $\uparrow$ to TxD Delay (X1 Mode)		300		230		200	2.5
13	TdTXD(TRX)	TxD to $\overline{TRxC}$ Delay (Send Clock Echo)		200		200		200	
14	TwRTXh	$\overline{RTxC}$ High Width	180		180		150		6
15	TwRTX1	$\overline{RTxC}$ Low Width	180		180		150		6
16	TcRTX	$\overline{RTxC}$ Cycle Time	1000		640		500		6.7
17	TcRTXX	Crystal Oscillator Period	250	1000	165	1000	125	1000	3
18	TwTRXh	$\overline{TRxC}$ High Width	180		180		150		6
19	TwTRX1	$\overline{TRxC}$ Low Width	180		180		150		6
20	TcTRX	$\overline{TRxC}$ Cycle Time	1000		640		500		6.7
21	TwEXT	$\overline{DCD}$ or $\overline{CTS}$ Pulse Width	200		200		200		
22	TwSY	$\overline{SYNC}$ Pulse Width	200		200		200		

Notes :
1. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
2. TxC is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
3. Both RTxC and SYNC have 30 pF capacitors to ground connected to them.
4. Parameter applies only if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between $\overline{RxC}$ and PCLK or $\overline{TxC}$ and PCLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Parameter applies only for transmitter and receiver ; DPLL and baud rate generator timing requirements are identical to chip PCLK requirements.
7. The maximum receive on transmit data is 1/4PLCK.

* Timings are preliminary and subject to change.

HARDWARE INFORMATION (cont'd)

Figure 80 : System Timing.



HARDWARE INFORMATION (cont'd)

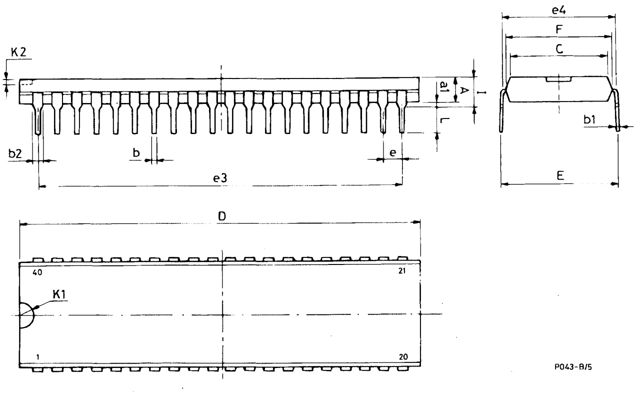
Table 26 : System Timing Characteristics (cont'd).

N°	Symbol	Parameter	Z8530		Z8530A		Z8530B		Notes* †
			Min.	Max.	Min.	Max.	Min.	Max.	
1	TdRXC(REQ)	$\overline{\text{RxC}}$ ↑ to $\overline{\text{W}}$ / $\overline{\text{REQ}}$ Valid Delay	8	12	8	12	8	12	2
2	TdRXC(W)	$\overline{\text{RxC}}$ ↑ to Wait Inactive Delay	8	12	8	12	8	14	1.2
3	TdRXC(SY)	$\overline{\text{RxC}}$ ↑ to $\overline{\text{SYNC}}$ Valid Delay	4	7	4	7	4	7	2
4	TdRXC(INT)	$\overline{\text{RxC}}$ ↑ to $\overline{\text{INT}}$ Valid Delay	10	16	10	16	10	16	1.2
5	TdTXC(REQ)	$\overline{\text{TxC}}$ ↓ to $\overline{\text{W}}$ / $\overline{\text{REQ}}$ Valid Delay	5	8	5	8	5	8	3
6	TdTXC(W)	$\overline{\text{TxC}}$ ↓ to Wait Inactive Delay	5	8	5	8	5	11	1.3
7	TdTXC(DRQ)	$\overline{\text{TxC}}$ ↓ to $\overline{\text{DTR}}$ / $\overline{\text{REQ}}$ Valid Delay	4	7	4	7	4	7	3
8	TdTXC(INT)	$\overline{\text{TxC}}$ ↓ to $\overline{\text{INT}}$ Valid Delay	6	10	6	10	6	10	1.3
9	TdSY(INT)	$\overline{\text{SYNC}}$ Transition to $\overline{\text{INT}}$ Valid Delay	2	6	2	6	2	6	1
10	TdEXT(INT)	$\overline{\text{DCD}}$ or $\overline{\text{CTS}}$ Transition to $\overline{\text{INT}}$ Valid Delay	2	6	2	6	2	6	1

Notes : 1. Open-drain output, measured with open-drain test load.
2. $\overline{\text{RxC}}$ is $\overline{\text{RTxC}}$ or $\overline{\text{TRxC}}$, whichever is supplying the receive clock.
3. $\overline{\text{TxC}}$ is $\overline{\text{TRxC}}$ or $\overline{\text{RTxC}}$, whichever is supplying the transmit clock.
* Timings are preliminary and subject to change.
† Units equal to TcPC.

PACKAGES MECHANICAL DATA

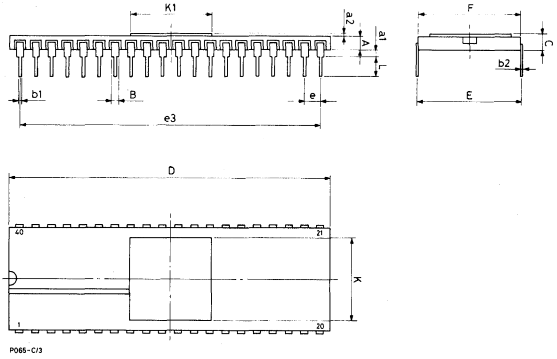
Figure 81: Z8530 40-Pin Dual in Line Plastic



Dim.	mm			Inches		
	Min	Typ	Max	Min	Typ	Max
A						
a1		0.63			.024	
b		0.45			.017	
b1	0.23		0.31	.009		.012
b2		1.27			.050	
C						
D			52.58			2.07
E	15.20		16.68	.598		.656
e		2.54			.100	
e3		48.26			1.90	
e4						
F			14.10			.555
I		4.45			.175	
L		3.30			.129	
K1						
K2						

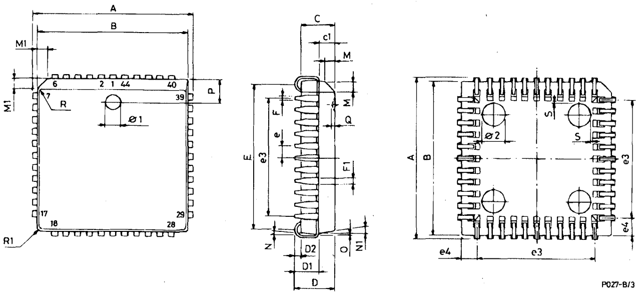
PACKAGE MECHANICAL DATA (cont'd)

Figure 82: Z8530 40-Pin Dual in Line Ceramic Multilayer.



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A						
a1		1.27			.050	
a2						
B		1.27			.050	
b1		0.45			.017	
b2		0.25			.010	
C		2.33			.091	
D			51.30			2.01
E		15.24			.600	
e		2.54			.100	
e3		48.26			1.90	
F			15.36			.604
L		3.30			.129	
K		12.80			.503	
K1		12.80			.503	

Figure 83 : Z8530 44–Lead Plastic Leaded Chip Carrier.



Dim.	mm			Inches		
	Min	Typ	Max	Min	Typ	Max
A	17.40		17.65	0.685		0.694
B	16.51		16.65	0.650		0.655
C	3.65		3.70	0.143		0.145
c1						
D	4.20		4.57	0.165		0.179
d1	2.59		2.74	0.101		0.107
d2		0.68			0.026	
E	14.99		16.00	0.590		0.629
e		1.27			0.050	
e3		12.70			0.500	
e4			1.98			0.077
F		0.46			0.018	
F1		0.71			0.027	
G			0.101			0.004
M		1.16			0.045	
M1		1.14			0.044	
Ø1		1.57				

ORDERING INFORMATION

Sales Type	Frequency	Supply Voltage	Temp. Range	Package
Z8530B1V	4 MHz	5 V ± 5 %	0 to + 70 °C	PDIP-40
Z8530B6V	4 MHz	5 V ± 5 %	– 40 to + 85 °C	PDIP-40
Z8530D1N	4 MHz	5 V ± 5 %	0 to + 70 °C	CDIP-40
Z8530D6N	4 MHz	5 V ± 5 %	– 40 to + 85 °C	CDIP-40
Z8530D2N	4 MHz	5 V ± 5 %	– 55 to + 125 °C	CDIP-40
Z8530C1V	4 MHz	5 V ± 5 %	0 to + 70 °C	PLCC44
Z8530C6V	4 MHz	5 V ± 5 %	– 40 to + 85 °C	PLCC44
Z8530AB1V	6 MHz	5 V ± 5 %	0 to + 70 °C	PDIP-40
Z8530AB6V	6 MHz	5 V ± 5 %	– 40 to + 85 °C	PDIP-40
Z8530AD1N	6 MHz	5 V ± 5 %	0 to + 70 °C	CDIP-40
Z8530AD6N	6 MHz	5 V ± 5 %	– 40 to + 85 °C	CDIP-40
Z8530AD2N	6 MHz	5 V ± 5 %	– 55 to + 125 °C	CDIP-40
Z8530AC1V	6 MHz	5 V ± 5 %	0 to + 70 °C	PLCC44
Z8530AC6V	6 MHz	5 V ± 5 %	– 40 to + 85 °C	PLCC44
Z8530BB1V	8 MHz	5 V ± 5 %	0 to + 70 °C	PDIP-40
Z8530BB6V	8 MHz	5 V ± 5 %	– 40 to + 85 °C	PDIP-40
Z8530BD1N	8 MHz	5 V ± 5 %	0 to + 70 °C	CDIP-40
Z8530BD6N	8 MHz	5 V ± 5 %	– 40 to + 85 °C	CDIP-40
Z8530BC1V	8 MHz	5 V ± 5 %	0 to + 70 °C	PLCC44
Z8530BC6V	8 MHz	5 V ± 5 %	– 40 to + 85 °C	PLCC44

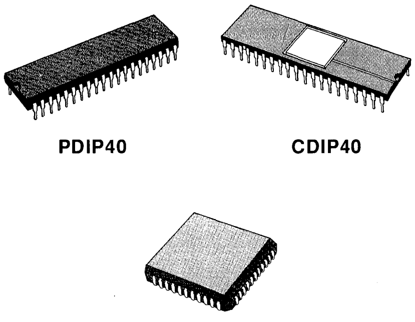
Note : PDIP = Plastic DIP ; CDIP = Ceramic Multilayer DIP ; PLCC = Plastic Leaded Chip Carrier.

SGS-THOMSON
MICROELECTRONICS

Z8531

ASYNCHRONOUS SERIAL COMMUNICATIONS CONTROLLER

- TWO INDEPENDENT, 0 TO 1M BIT/SECOND, FULL-DUPLEX CHANNELS, EACH WITH A SEPARATE CRYSTAL OSCILLATOR AND BAUD RATE GENERATOR
- PROGRAMMABLE FOR NRZ, NRZI, OR FM DATA ENCODING
- LOCAL LOOPBACK AND AUTO ECHO MODES
- ASYNCHRONOUS COMMUNICATIONS WITH FIVE TO EIGHT BITS PER CHARACTER AND ONE, ONE AND ONE-HALF, OR TWO STOP BITS PER CHARACTER ; PROGRAMMABLE CLOCK FACTOR ; BREAK DETECTION AND GENERATION ; PARITY, OVERRUN, AND FRAMING ERROR DETECTION


PDIP40 CDIP40

PLCC44

(Ordering Information at the end of the datasheet)

DESCRIPTION

The Z8531 ASCC Asynchronous Serial Communications Controller is a dual-channel, multi-protocol data communications peripheral designed for use with conventional non-multiplexed buses. The ASCC functions as a serial-to-parallel, parallel-to-serial converter/controller. The device contains a variety of new, sophisticated internal functions including on-chip baud rate generators and crystal oscillators that dramatically reduce the need for external logic

The ASCC also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

The Z-BUS daisy-chain interrupt hierarchy is also supported–as is standart for Zilog peripheral components.

Figure 1 : Pin Functions.

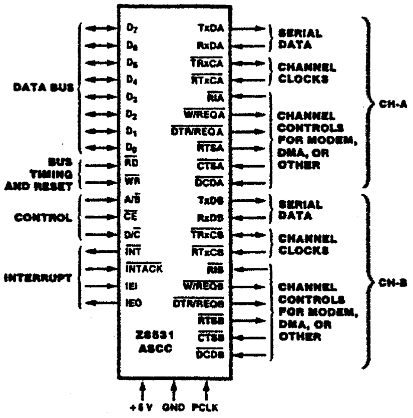


Figure 2a : DIP Pin Connections.

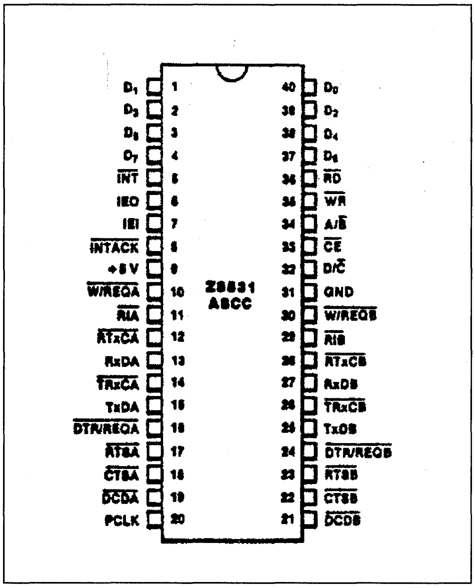
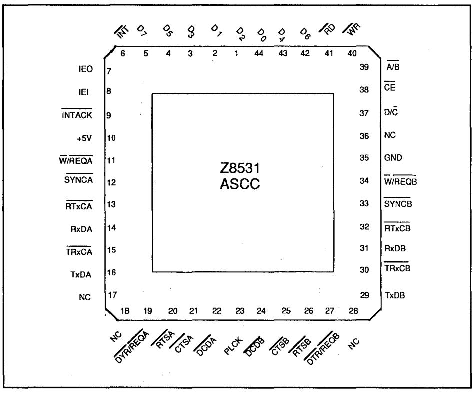


Figure 2b : Chip Carrier Pin Connections.



PIN DESCRIPTION

The following section describes the pin functions of the ASCC. Figures 1 and 2 detail the respective pin functions and pin assignments.

A/B. *Channel A/Channel B Select* (input). This signal selects the channel in which the read or write operation occurs.

CE. *Chip Enable* (input, active Low). This signal selects the ASCC for a read or write operation.

CTSA, CTSB. *Clear To Send* (inputs, active Low). If these pins are programmed as Auto Enables, a Low on the inputs enables the respective transmitters.

If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The ASCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.

D/C. *Data/Control Select* (input). This signal defines the type of information transferred to or from the ASCC. A High means data is transferred ; a Low indicates a command.

DCDA, **DCDB**. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if they are programmed for Auto Enables ; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accomodate slow rise-time signals. The ASCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.

D_0-D_7. *Data Bus* (bidirectional, 3-state). These lines carry data and commands to and from the ASCC.

DTR/REQA, **DTR/REQB**. *Data Terminal Ready/Request* (outputs, active Low). These outputs follow the state programmed into the DTR bit.They can also be used as general-purpose outputs or as Request lines for a DMA controller.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an ASCC interrupt or the ASCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

INT. *Interrupt Request* (output, open-drain, active Low). This signal is activated when the ASCC requests an interrupt.

INTACK. *Interrupt Acknowledge* (input, active Low). This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the ASCC interrupt daisy chain settles. When RD becomes active, the ASCC places an interrupt vector on the data bus (if IEI is High).

INTACK is latched by the rising edge of PCLK.

SGS-THOMSON MICROELECTRONICS

PIN DESCRIPTION (continued)

PCLK. *Clock* (input). This is the master ASCC clock used to synchronize internal signals ; PCLK is a TTL level signal.

$\overline{\text{RD}}$. *Read* (input, active Low). This signal indicates a read operation and when the ASCC is selected, enables the ASCC's bus drivers. During the Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the bus if the ASCC is the highest priority device requesting an interrupt.

RxDA, RxDB. *Receive Data* (inputs, active High). These input signals receive serial data at standard TTL levels.

$\overline{\text{RIA}}$, $\overline{\text{RIB}}$. *Ring Indicator* (inputs, active Low).These pins can act either as inputs, or part of the crystal oscillator circuit. In normal mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. In this mode, transitions on these lines affect the state of the Ring Indicator status bits in Read Register 0 (Figure 8) but have no other function.

$\overline{\text{RTxCA}}$, $\overline{\text{RTxCB}}$. *Receive/Transmit Clocks* (inputs, active Low). These pins can be programmed in several different modes of operation. In each channel, $\overline{\text{RTxC}}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the Digital Phase-Locked Loop. These pins can also be programmed for use with the respective $\overline{\text{RI}}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.

$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$. *Request To Send* (outputs, active Low). When the Request To Send (RTS) bit in Write Register 5 (Figure 9) is set, the $\overline{\text{RTS}}$ signal goes Low. When the RTS bit is reset in the Asynchronous mode and Auto Enable is on, the signal goes High after the transmitter is empty. With Auto Enable off, the $\overline{\text{RTS}}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

TxDA, **TxDB**. *Transmit Data* (outputs, active High). These output signals transmit serial data at standard TTL levels.

$\overline{\text{TRxCA}}$, **$\overline{\text{TRxCB}}$**. *Transmit/Receive Clocks* (inputs or outputs, active Low). These pins can be programmed in several different modes of operation. $\overline{\text{TRxC}}$ may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase-Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.

$\overline{\text{WR}}$. *Write* (input, active Low). When the ASCC is selected, this signal indicates a write operation. The coincidence of $\overline{\text{RD}}$ and $\overline{\text{WR}}$ is interpreted as a reset.

$\overline{\text{W/REQA}}$, **$\overline{\text{W/REQB}}$**. *Wait/Request* (outputs, open-drain when programmed for a Wait function, driven High or Low when programmed for a Request function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the ASCC data rate. The reset state is Wait.

FUNCTIONAL DESCRIPTION

The functional capabilities of the ASCC can be described from two different points of view : as a data communications device, it transmits and receives data in a wide variety of data communications protocols ; as a microprocessor peripheral, the ASCC offers valuable features such as vectored interrupts, polling, and simple handshake capability.

Data Communications Capabilities. The ASCC provides two independent full-duplex channels programmable for use in any common Asynchronous data communication protocol. Figure 3 and the following description briefly detail this protocol.

Asynchronous Modes. Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 1). If the Low does not persist (as in the case of a tran-sient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids the interpretation of a framing error as a new start bit : a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

The ASCC does not require symmetric transmit and receive clock signals–a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1/16, 1/32, or

FUNCTIONAL DESCRIPTION (continued)

1/64 of the clock rate supplied to the receive and transmit clock inputs.

Baud Rate Generator. Each channel in the ASCC contains a programmable baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in the High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching 0, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the Digital Phase-Locked Loop (see next section).

If the receive clock or transmit clock is not programmed to come from the $\overline{\text{TRxC}}$ pin, the output of the baud rate generator may be echoed out via the $\overline{\text{TRxC}}$ pin.

The following formula relates the time constant to the baud rate (the baud rate is in bits/second and the BR clock period is in seconds) :

$$\text{time constant} = \frac{\text{PCLK}}{2\ (\text{clock factor})\ (\text{baud})} - 2$$

Digital Phase-Locked Loop. The ASCC contains a Digital Phase-Locked-Loop (DPLL) to recover clock information from a data stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a clock for the data. This clock may then be used as the ASCC receive clock, the transmit clock, or both.

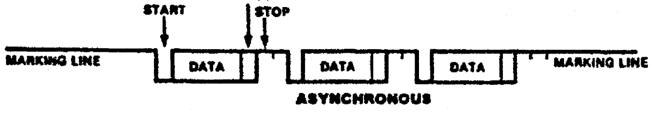
For NRZI encoding, the DPLL counts the 32x clock to create nominal bit times. As the 32x clock is counted, the DPLL is searching the incoming data stream for edges (either 1 to 0 or 0 to 1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 0 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the data stream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the 15 to 16 counting transition.

The 32x clock for the DPLL can be programmed to come from either the $\overline{\text{RTxC}}$ input of the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the ASCC via the $\overline{\text{TRxC}}$ pin (if this pin is not being used as an input).

Data Encoding. The ASCC may be programmed to encode and decode the serial data in four different ways (Figure 4). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encodint, a 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi-phase mark), a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FMO (bi-phase space), a transi-tion occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the ASCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0 to 1, the bit is a 0. If the transition is 1 to 0, the bit is a 1.

Figure 3 : ASCC Protocol.



FUNCTIONAL DESCRIPTION (continued)

Auto Echo and Local Loopback. The ASCC is capable of automatically echoing everything it receives. In Auto Echo mode, RxD is connected to TxD internally. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the data stream is not decoded before retransmission. In Auto Echo mode, the CTS input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and WAIT/REQUEST on transmit.

The ASCC is also capable of local loopback. In this mode TxD is connected to RxD internally, just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). The CTS and DCD inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works with NRZ, NRZI or FM coding of the data stream.

I/O Interface Capabilities. ASCC offers the choice of Polling, Interrupt (vectored or non vectored), and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. All interrupts are disabled. Three status registers in the ASCC are automatically updated whenever any function is performed. The idea behing polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be read ; depending on its contents, the CPU either writes data, reads data, or continues.

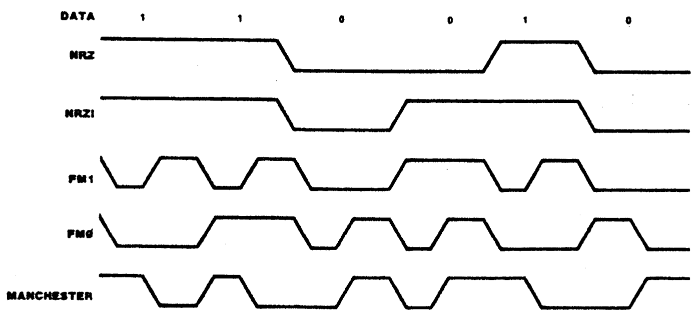
Two bits in the register indicate the need for data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

Interrupts. When an ASCC responds to an Interrupt Acknowledge signal (INTACK) from the CPU an interrupt vector may be placed on the data bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 8 and 9).

To speed interrupt response time, the ASCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included ; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the ASCC (Transmit, Receive, and External/ Status interrupts in both channels) has three bits associated with the interrupt source : Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bit is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write only.

Figure 4 : Data Encoding Methods.



FUNCTIONAL DESCRIPTION (continued)

The other two bits are related to the interrupt priority chain (Figure 5). As a microprocessor peripheral, the ASCC may request an interrupt only when no higher priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{\text{INT}}$. The CPU then res-ponds with INTACK, and the interrupting device places the vector on the data bus.

In the ASCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{\text{INT}}$ output is pulled Low, requesting an interrupt. In the ASCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the ASCC and external to the ASCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the ASCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts :

Transmit, Receive, and External/Status. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit, and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one ot three ways :

- Interrupt on First Receive Character or Special Receive Condition.
- Interrupt on All Receive Characters or Special Receive Condition.
- Interrupt on Special Receive Condition Only.

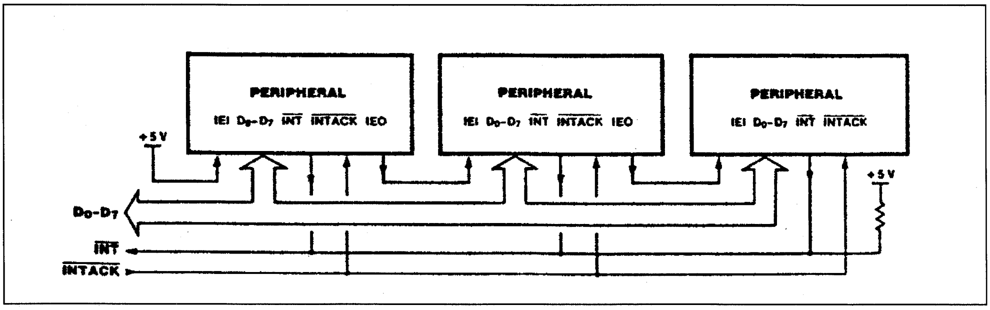
Interrupt on First Character or Special Condition and Interrupt on Special Condition Only are typically used with the Block Transfer mode. A Special Receive Condition is a receiver overrun, and, optionally, a parity error. The Special Receive Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector during the Interrupt Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive Conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{\text{CTS}}$, $\overline{\text{DCD}}$ and $\overline{\text{RI}}$ pins ; however, an External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count in the baud rate generator, or by the detection of a Break.

CPU/ DMA Block Transfer. The ASCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{\text{WAIT}}$/ $\overline{\text{REQUEST}}$ output in conjunction with the Wait/Request bits in WR1. The $\overline{\text{WAIT}}$/ $\overline{\text{REQUEST}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{REQUEST}}$ line in the DMA Block Transfer mode.

To a DMA controller, the ASCC $\overline{\text{REQUEST}}$ output indicates that the ASCC is ready to tranfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ line indicates that the ASCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{\text{DTR}}$/ $\overline{\text{REQUEST}}$ line allows full-duplex operation under DMA control.

Figure 5 : Interrupt Schedule.



ARCHITECTURE

The ASCC internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to a nonmultiplexed bus. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface to modems of other external devices (Figure 6).

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten control (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a write only Master Interrupt Control register and three read registers : one containing the vector with status information (Channel B only), one containing the vector without status (Channel A only), and one containing the Interrupt Pending bits (Channel A only).

The registers for each channel are designated as follows :

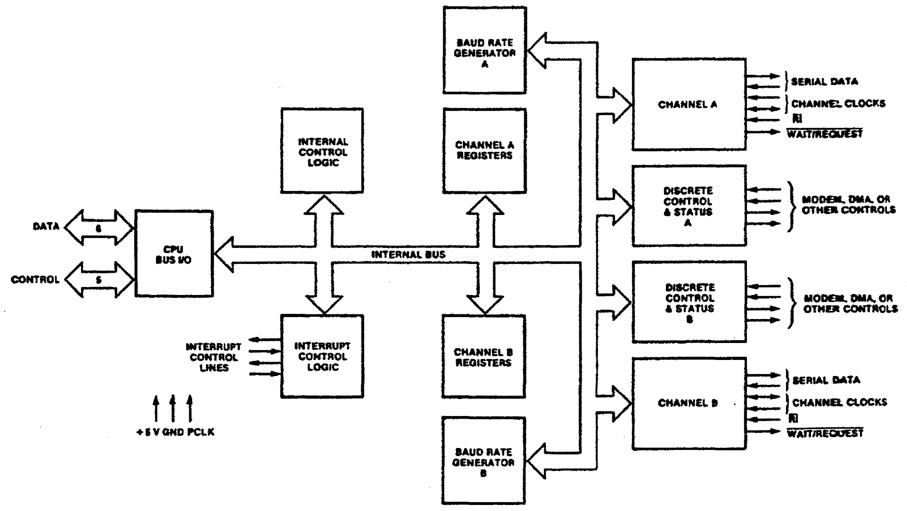
WR0-WR15 - Write Registers 0-5, 8-15.

RR0-RR3, RR10, RR12, RR13, RR15 - Read Registers 0 through 3, 10, 12, 13, 15.

Table 1 lists the functions assigned to each read or write register. The ASCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

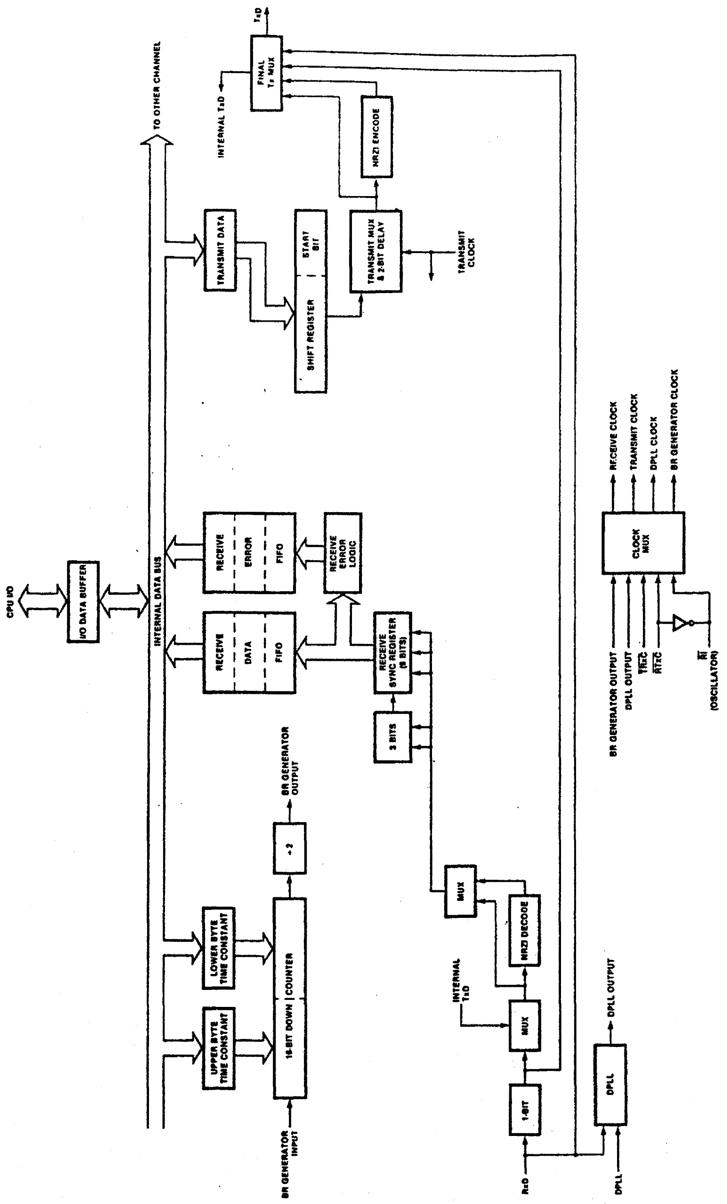
Data path. The transmit and receive data path illustrated in Figure 7 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high speed data. Incoming data is routed through one of several paths depending on the selected mode (the character length also determines the data path).

Figure 6 : Block Diagram of ASCC Architecture.



ARCHITECTURE (continued)

Figure 7 : Data Path.



SGS-THOMSON MICROELECTRONICS

ARCHITECTURE (continued)

The transmitter has an 8-bit Transmit Data buffer register loaded from the internal data bus and an 11-bit Transmit Shift register that can be loaded from the Transmit Data register.

Table 1. Read and Write Register Functions.

Read Register Functions	
RR0	Transmit/Receive buffer status and External status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Chanel B only) Unmodified interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive buffer
RR10	Miscellaneous status
RR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant
RR15	External/Status interrupt information

Write Register Functions	
WR0	CRC initialize, initialization commands for the various modes, Registers Pointers
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (accessed through either channel)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR8	Transmit buffer
WR9	Master it interrupt control and reset (accessed through either channel)
WR10	Miscellaneous transmitter/receiver control bits
WR11	Clock mode control
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits
WR15	External/Status interrupt control

PROGRAMMING

The ASCC contains 11 write registers in each channel that are programmed by the system separately to configure the functional personality of the channels.

In the ASCC, register addressing is direct for the data registers only, which are selected by a High on the D/$\overline{C}$ pin. In all other cases (with the exception of WR0 and RR0), programming the write registers requires two write operations and reading the read registers requires both a write and a read operation. The first write is to WR0 and contains three bits that point to the selected register. The second write is the actual control word for the selected register, and if the second operation is read, the selected read register is accessed. All of the registers in the ASCC, including the data registers, may be accessed in this fashion. The pointer bits are automatically cleared after the read or write operation so that WR0 (or RR0) is addressed again.

The system program first issues a series of commands to initialize the basic mode of operation. For example, the character length, clock rate, number of stop bits, even or odd parity might be set first. Then the interrupt mode would be set, and finally, receiver or transmitter enable.

Read Registers. The ASCC contains eight read registers (actually nine, counting the receive buffer (RR8) in each channel). Four of these may be read ot obtain status information (RRO, RR1, RR10, and RR15). Two registers baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 8 shows the formats for each read register.

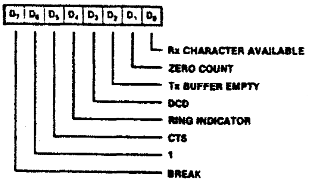
The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring; e.g., when the interrupt vector indicated a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

Write Registers. The ASCC contains 11 write registers (12 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 9 shows the format of each write register.

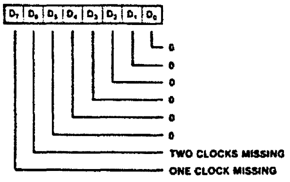
PROGRAMMING (continued)

Figure 8 : Read Register Bit Functions.

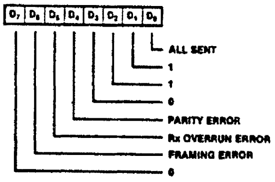
Read Register 0



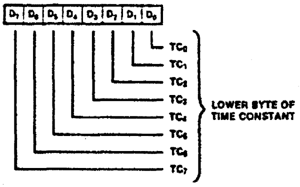
Read Register 10



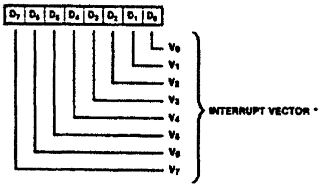
Read Register 1



Read Register 12

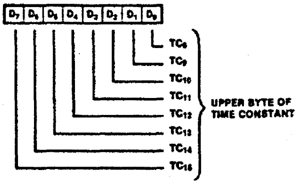


Read Register 2

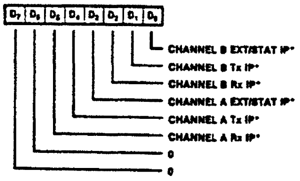


*MODIFIED IN B CHANNEL

Read Register 13

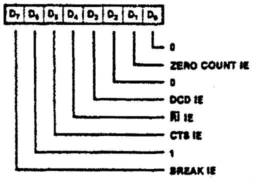


Read Register 3



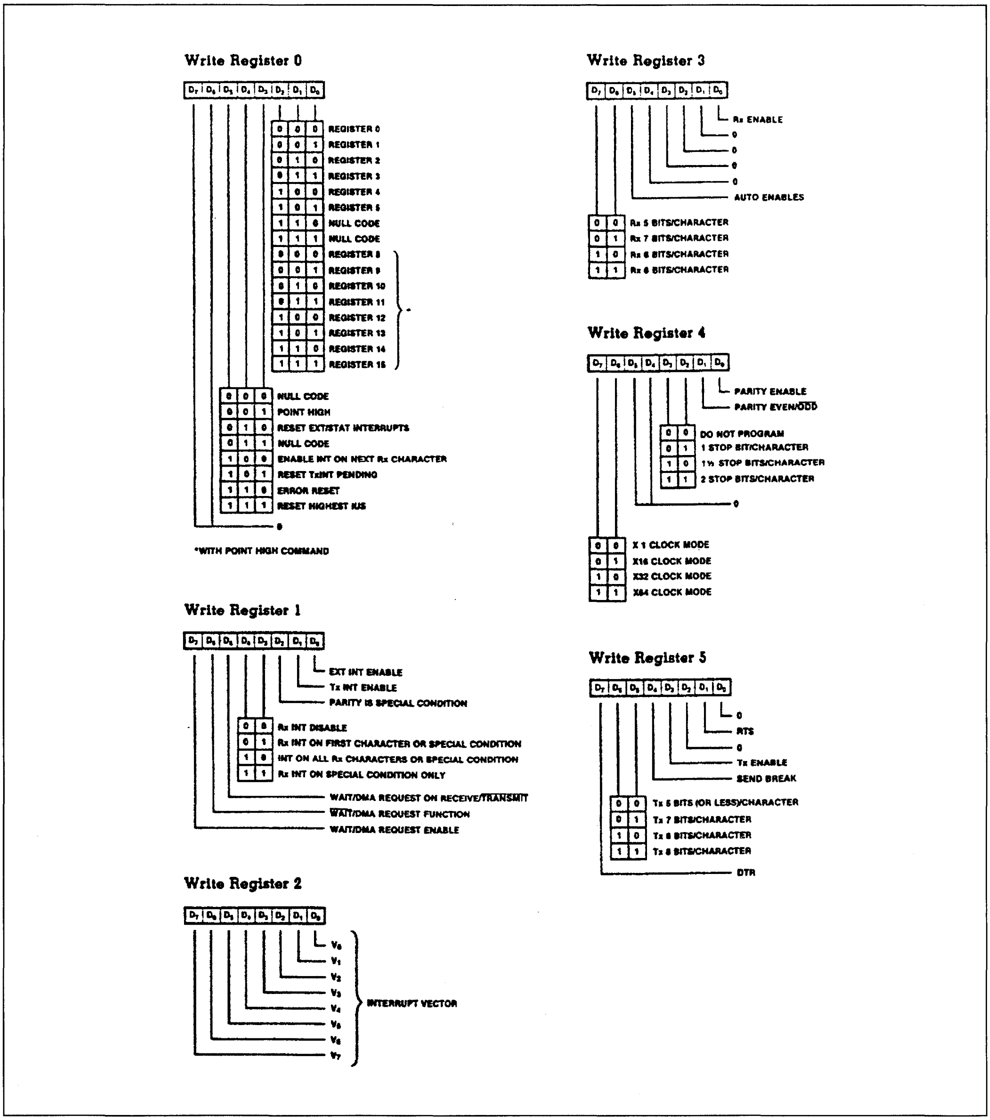
*ALWAYS 0 IN B CHANNEL

Read Register 15



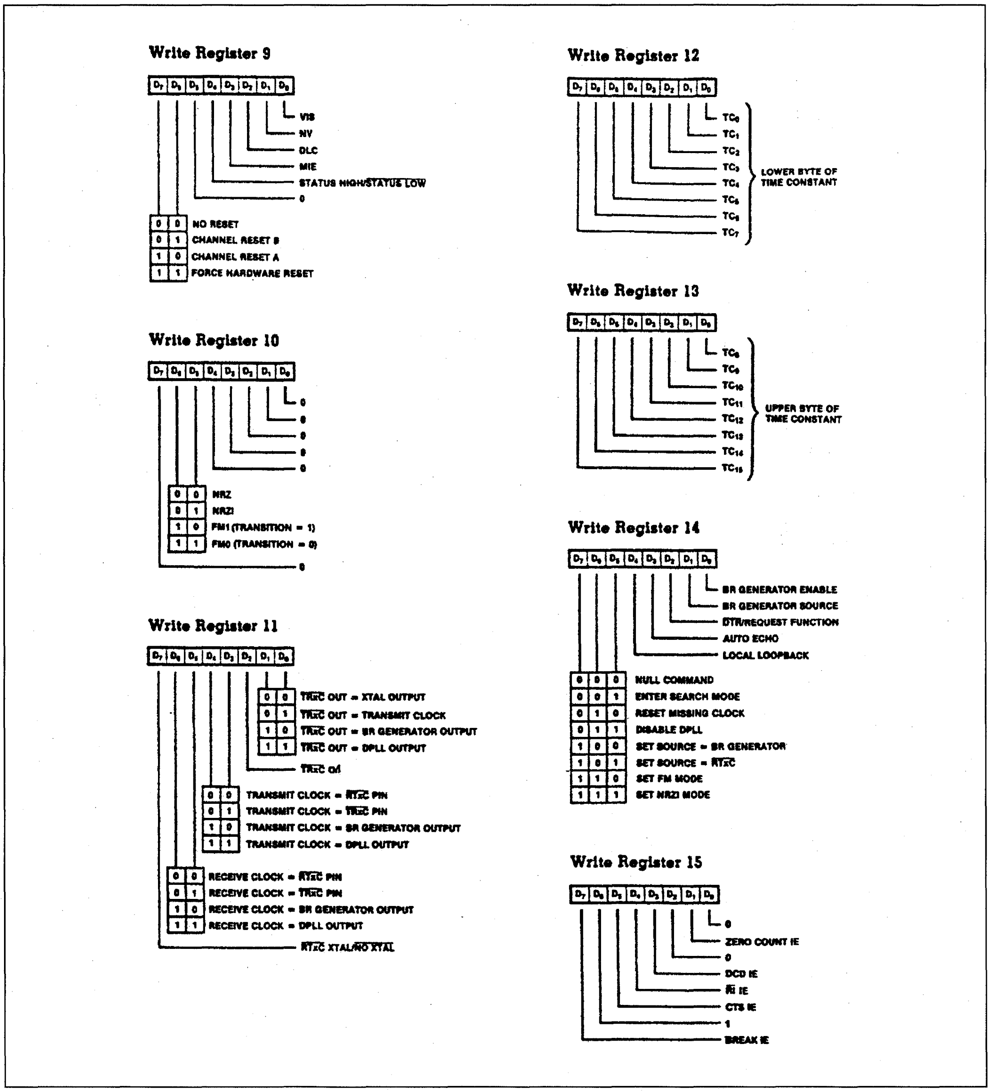
PROGRAMMING (continued)

Figure 9 : Write Register Bit Functions.



PROGRAMMING (continued)

Figure 9 : Write Register Bit Functions (continued).


Write Register 9
D7 D6 D5 D4 D3 D2 D1 D0
VIS
NV
DLC
MIE
STATUS HIGH/$\overline{\text{STATUS LOW}}$
0
0 0 NO RESET
0 1 CHANNEL RESET B
1 0 CHANNEL RESET A
1 1 FORCE HARDWARE RESET
Write Register 10
0
0
0
0
0
0 0 NRZ
0 1 NRZI
1 0 FM1 (TRANSITION = 1)
1 1 FM0 (TRANSITION = 0)
0
Write Register 11
0 0 $\overline{\text{TRxC}}$ OUT = XTAL OUTPUT
0 1 $\overline{\text{TRxC}}$ OUT = TRANSMIT CLOCK
1 0 $\overline{\text{TRxC}}$ OUT = BR GENERATOR OUTPUT
1 1 $\overline{\text{TRxC}}$ OUT = DPLL OUTPUT
$\overline{\text{TRxC}}$ O/I
0 0 TRANSMIT CLOCK = $\overline{\text{RTxC}}$ PIN
0 1 TRANSMIT CLOCK = $\overline{\text{TRxC}}$ PIN
1 0 TRANSMIT CLOCK = BR GENERATOR OUTPUT
1 1 TRANSMIT CLOCK = DPLL OUTPUT
0 0 RECEIVE CLOCK = $\overline{\text{RTxC}}$ PIN
0 1 RECEIVE CLOCK = $\overline{\text{TRxC}}$ PIN
1 0 RECEIVE CLOCK = BR GENERATOR OUTPUT
1 1 RECEIVE CLOCK = DPLL OUTPUT
$\overline{\text{RTxC}}$ XTAL/$\overline{\text{NO XTAL}}$
Write Register 12
TC0
TC1
TC2
TC3
TC4
TC5
TC6
TC7
LOWER BYTE OF TIME CONSTANT
Write Register 13
TC8
TC9
TC10
TC11
TC12
TC13
TC14
TC15
UPPER BYTE OF TIME CONSTANT
Write Register 14
BR GENERATOR ENABLE
BR GENERATOR SOURCE
$\overline{\text{DTR}}$/REQUEST FUNCTION
AUTO ECHO
LOCAL LOOPBACK
0 0 0 NULL COMMAND
0 0 1 ENTER SEARCH MODE
0 1 0 RESET MISSING CLOCK
0 1 1 DISABLE DPLL
1 0 0 SET SOURCE = BR GENERATOR
1 0 1 SET SOURCE = $\overline{\text{RTxC}}$
1 1 0 SET FM MODE
1 1 1 SET NRZI MODE
Write Register 15
0
ZERO COUNT IE
0
DCD IE
$\overline{\text{RI}}$ IE
CTS IE
1
BREAK IE

SGS-THOMSON MICROELECTRONICS

TIMING

The ASCC generates internal control signals from $\overline{WR}$ and $\overline{RD}$ that are related to PCLK. Since PLCK has no phase relationship with $\overline{WR}$ and $\overline{RD}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the ASCC. The recovery time required for proper operation is specified from the rising edge of $\overline{WR}$ or $\overline{RD}$ in the first transaction involving the ASCC to the falling edge of $\overline{WR}$ or $\overline{RD}$ in the second transaction involving the ASCC. This time must be at least 6 PLCK cycles plus 200 ns.

Read Cycle Timing. Figure 10 illustrates read cycle timing, Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{RD}$ falls, or rises before $\overline{RD}$ rises, the effective $\overline{RD}$ is shortened.

Write Cycle Timing. Figure 11 illustrates write cycle timing. Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{WR}$ falls or rises before $\overline{WR}$ rises, the effective $\overline{WR}$ is shortened.

Interrupt Acknowledge Cycle Timing. Figure 12 illustrates interrupt acknowledge cycle timing. Between the time $\overline{INTACK}$ goes low and the falling edge of $\overline{RD}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending the ASCC and IEI is High when $\overline{RD}$ falls, the acknowledge cycle was intended for the ASCC. In this case, the ASCC may be programmed to respond to $\overline{RD}$ Low by placing its interrupt vector on D_0-D_7 and sets the appropriate Interrupt-Under-Service latch internally.

Figure 10 : Read Cycle Timing.

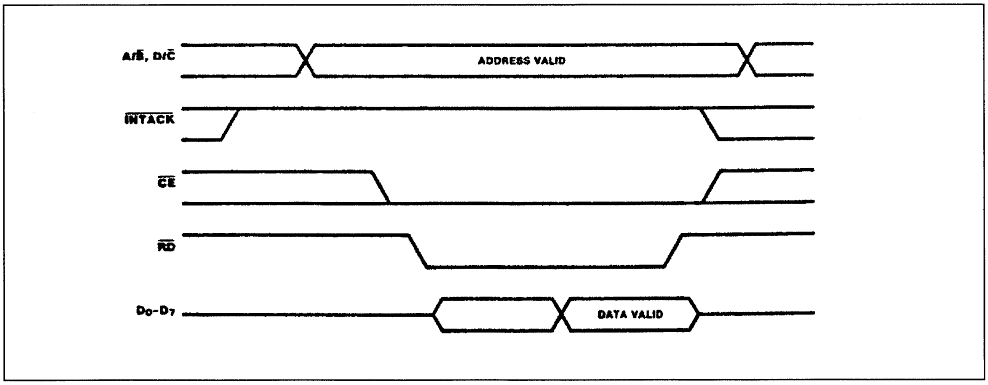
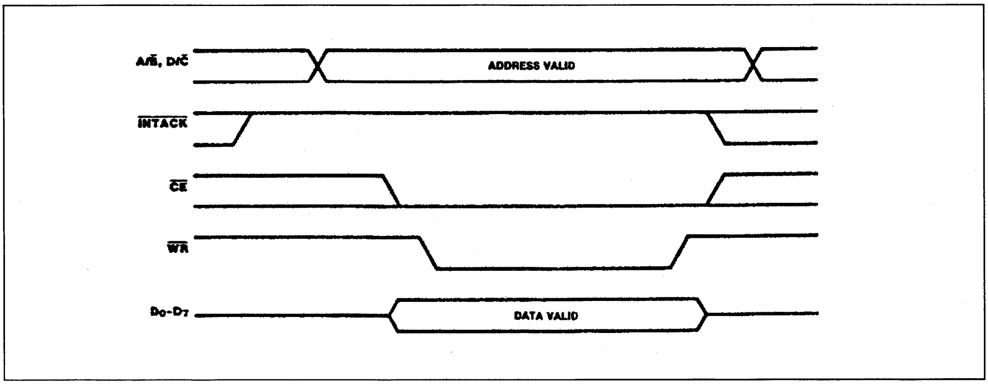
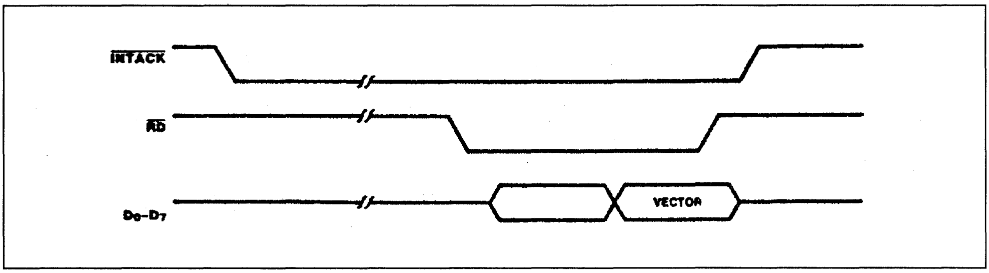


Figure 11 : Write Cycle Timing.



TIMING (continued)

Figure 12 : Interrupt Acknowledge Cycle Timing.



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_I	Voltages on all Pins with Respect to GND	– 0.3 to + 7.0	V
T_A	Operating Ambient Temperature	0 to + 70 – 40 to + 85 – 55 to + 125	°C
T_{stg}	Storage Temperature	– 65 to + 150	°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only ; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDART TEST CONDITIONS

The DC characteristics and capacitance sections below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are as follows :
- $+ 4.75V \leq V_{CC} \leq + 5.25V$
- GND = 0V
- T_A as specified in Order Codes

The Ordering Information section lists temperature ranges and product numbers. Package drawings are in the Package Information section in this book. Refer to the Literature List for additional documentation.

All ac parameters assume a load capacitance of 50 pf max.

Figure 13 : Standart Test Load.

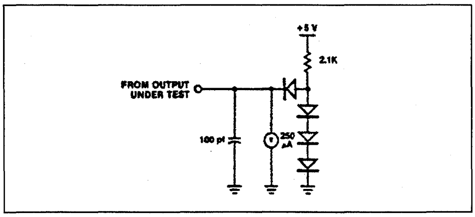
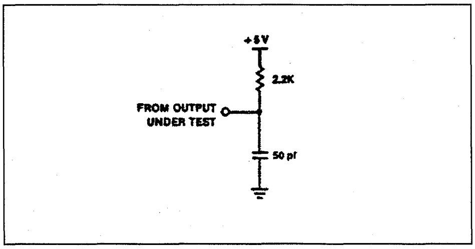


Figure 14 : Open-Drain Test Load.



DC CHARACTERISTICS

Symbol	Parameter	Test Conditions	Min.	Max.	Unit
V_{IH}	Input High Voltage	–	2.0	V_{CC} + 0.3	V
V_{IL}	Input Low Voltage	–	– 0.3	0.8	V
V_{OH}	Output High Voltage	I_{OH} = – 250 µA	2.4	–	V
V_{OL}	Output Low Voltage	I_{OL} = + 2 mA	–	0.4	V
I_{IL}	Input Leakage	$0.4 \leq V_{IN} \leq + 2.4$ V	–	± 10	µA
I_{OL}	Output Leakage	$0.4 \leq V_{OUT} \leq + 2.4$ V	–	± 10	µA
I_{CC}	V_{CC} Supply Current	–	–	250	mA

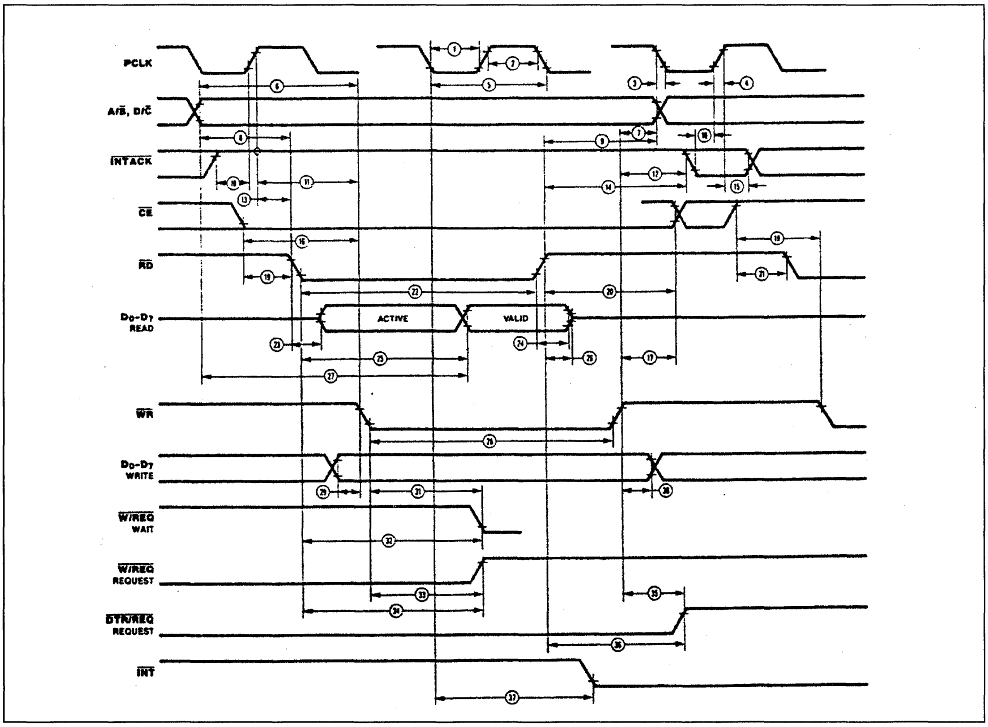
V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

CAPACITANCE

Symbol	Parameter	Test Conditions	Min.	Max.	Unit
C_{IN}	Input Capacitance	–	–	10	pf
C_{OUT}	Output Capacitance	–	–	15	pf
$C_{I/O}$	Bidirectional Capacitance	–	–	20	pf

f : 1 MHz, over specified temperature range.
Unmeasured pins returned to ground.

READ AND WRITE TIMING



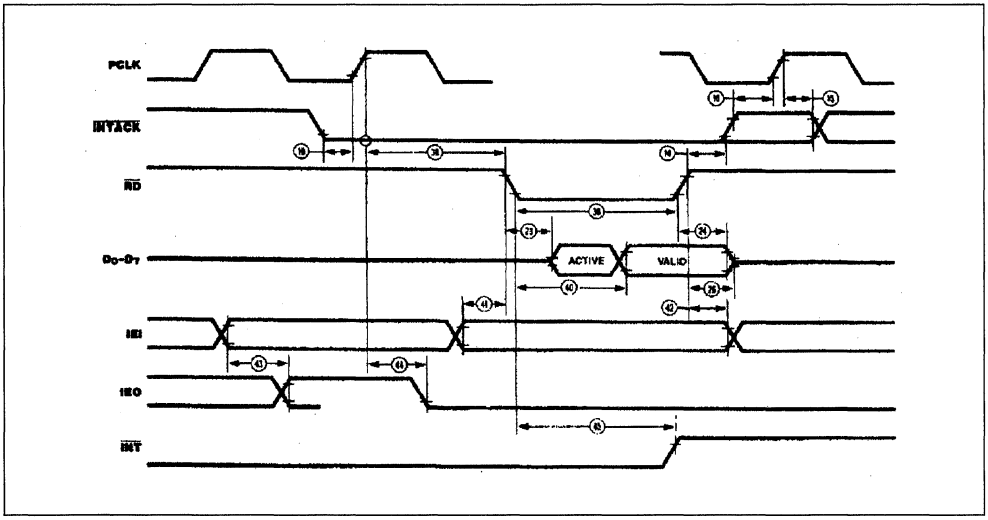
SGS-THOMSON MICROELECTRONICS

READ AND WRITE TIMING (continued)

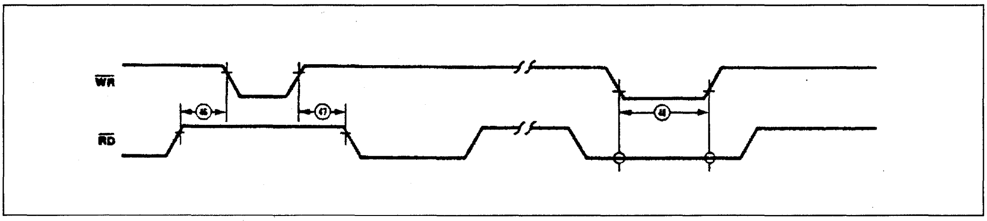
N°	Symbol	Parameter	4 MHz		6 MHz		8 MHz		Notes *
			Min.	Max.	Min.	Max.	Min.	Max.	†
1	TwPCl	PCLK Low Width	105	2000	70	1000	50	1000	–
2	TwPCh	PCLK High Width	105	2000	70	1000	50	1000	–
3	TfPC	PCLK Fall Time	–	– 20	–	10	–	10	–
4	TrPC	PCLK Rise Time	–	– 20	–	15	–	10	–
5	TcPC	PCLK Cycle Time	250	4000	165	2000	125	2000	–
6	TsA(WR)	Address to $\overline{\text{WR}}$ ↓ Setup Time	80	–	80	–	70	–	–
7	ThA(WR)	Address to $\overline{\text{WR}}$ ↑ Hold Time	0	–	0	–	0	–	–
8	TsA(RD)	Address to $\overline{\text{RD}}$ ↓ Setup Time	80	–	80	–	70	–	–
9	ThA(RD)	Address to $\overline{\text{RD}}$ ↑ Hold Time	0	–	0	–	0	–	–
10	TsIA(PC)	$\overline{\text{INTACK}}$ to PCLK ↑ Setup Time	10	–	10	–	10	–	–
11	ThIA(WR)	$\overline{\text{INTACK}}$ to $\overline{\text{WR}}$ ↓ Setup Time	200	–	160	–	145	–	1
12	ThIA(WR)	$\overline{\text{INTACK}}$ to $\overline{\text{WR}}$ ↑ Hold Time	0	–	0	–	0	–	–
13	TsIAi(RD)	$\overline{\text{INTACK}}$ to $\overline{\text{RD}}$ ↓ Setup Time	200	–	160	–	145	–	1
14	ThIA(RD)	$\overline{\text{INTACK}}$ to $\overline{\text{RD}}$ ↑ Hold Time	0	–	0	–	0	–	–
15	ThIA(PC)	$\overline{\text{INTACK}}$ to PCLK ↑ Hold Time	100	–	100	–	85	–	–
16	TsCEl(WR)	$\overline{\text{CE}}$ Low to $\overline{\text{WR}}$ ↓ Setup Time	0	–	0	–	0	–	–
17	ThCE(WR)	$\overline{\text{CE}}$ to $\overline{\text{WR}}$ ↑ Hold Time	0	–	0	–	0	–	–
18	TsCEh(WR)	$\overline{\text{CE}}$ High to $\overline{\text{WR}}$ ↓ Setup Time	100	–	70	–	60	–	–
19	TsCEl(RD)	$\overline{\text{CE}}$ Low to $\overline{\text{RD}}$ ↓ Setup Time	0	–	0	–	0	–	1
20	ThCE(RD)	$\overline{\text{CE}}$ to $\overline{\text{RD}}$ ↑ Hold Time	0	–	0	–	0	–	1
21	TsCEh(RD)	$\overline{\text{CE}}$ High to $\overline{\text{RD}}$ ↓ Setup Time	100	–	70	–	60	–	1
22	TwRDl	$\overline{\text{RD}}$ Low Width	240	–	200	–	150	–	1
23	TdRD(DRA)	$\overline{\text{RD}}$ ↓ to Read Data Active Delay	0	–	0	–	0	–	–
24	TdRDr(DR)	$\overline{\text{RD}}$ ↑ To Read Data Not Valid Delay	0	–	0	–	0	–	–
25	TdRDf(DR)	$\overline{\text{RD}}$ ↓ to Read Data Valid Delay	–	250	–	180	–	140	–
26	TdRD(DRz)	$\overline{\text{RD}}$ ↑ to Read Data Float Delay	–	70	–	45	–	40	2

Notes : 1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is defined as the time required for a ± 0.5 V change in the output with a maximum dc load and minimum ac load.
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

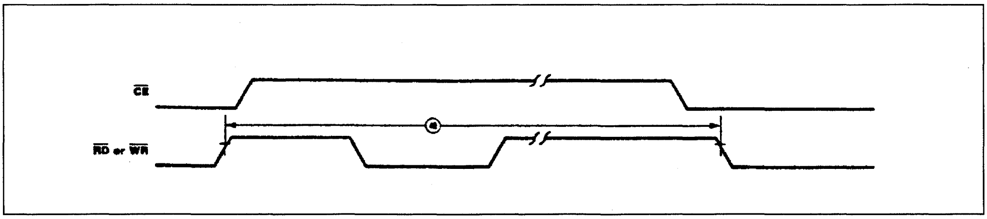
INTERRUPT ACKNOWLEDGE TIMING



RESET TIMING



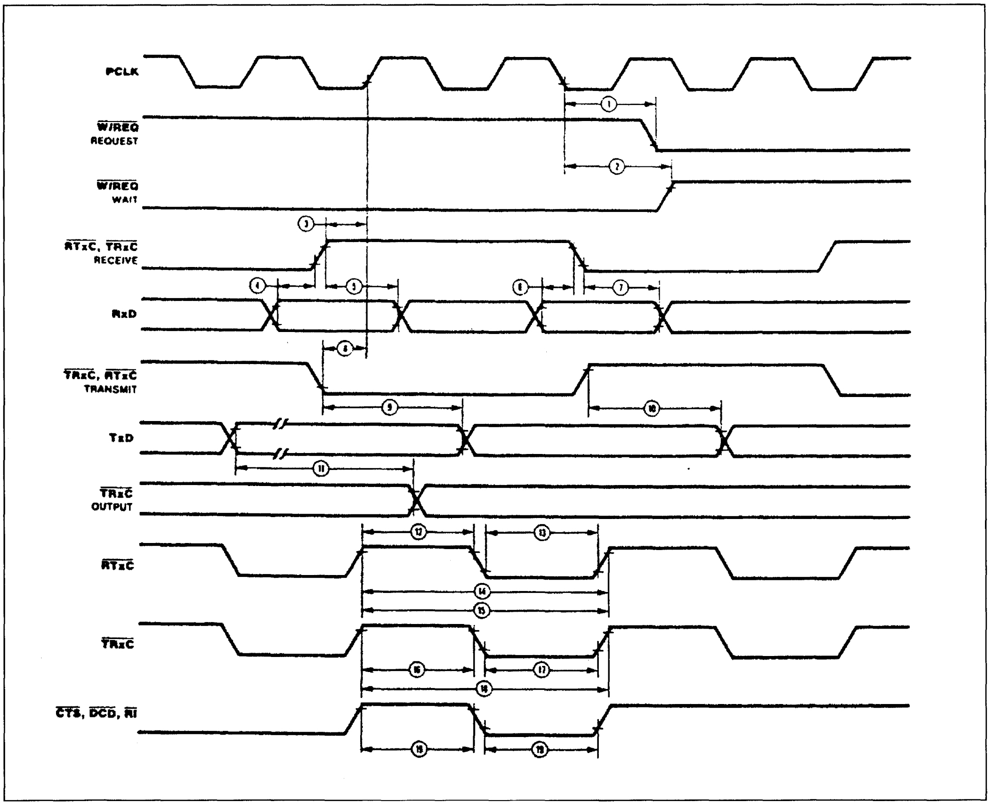
CYCLE TIMING



N°	Symbol	Parameter	4 MHz		6 MHz		8 MHz		Notes *
			Min.	Max.	Min.	Max.	Min.	Max.	†
27	TdA(DR)	Address Required Valid to Read Data Valid Delay	–	300	–	280	–	220	–
28	TwWRl	$\overline{\text{WR}}$ Low Width	240	–	200	–	150	–	–
29	TsDW(WR)	Write Data to $\overline{\text{WR}}$ ↓ Setup Time	10	–	10	–	10	–	–
30	ThDW(WR)	Write Data $\overline{\text{WR}}$ ↑ Hold time	0	–	0	–	–	–	–
31	TdWR(W)	$\overline{\text{WR}}$ ↓ to Wait Valid Delay	–	240	–	200	–	170	4
32	TdRD(W)	$\overline{\text{RD}}$ ↓ to Wait Valid Delay	–	240	–	200	–	170	4
33	TdWRf(REQ)	$\overline{\text{WR}}$ ↓ to $\overline{\text{W}}/\overline{\text{REQ}}$ Not Valid Delay	–	240	–	200	–	170	–
34	TdRDf(REQ)	$\overline{\text{RD}}$ ↓ to $\overline{\text{W}}/\overline{\text{REQ}}$ Not Valid Delay	–	240	–	200	–	170	–
35	TdWRr(REQ)	$\overline{\text{WR}}$ ↑ to $\overline{\text{DTR}}/\overline{\text{REQ}}$ not Valid Delay	–	4TcPC	–	4TcPC	–	4TcPC	–
36	TdRD(REQ)	RD ↑ $\overline{\text{DTR}}/\overline{\text{REQ}}$ Not Valid Delay	–	4TcPC	–	4TcPC	–	4TcPC	–
37	TdPC(INT)	PLCK ↓ To $\overline{\text{INT}}$ Valid Delay	–	500	–	500	–	500	4
38	TdA(RD)	$\overline{\text{INTACK}}$ to $\overline{\text{RD}}$ ↓ (acknowledge) Delay	250	–	200	–	150	–	5
39	TwRDA	$\overline{\text{RD}}$ (acknowledge) Width	250	–	200	–	150	–	–
40	TdRDA(DR)	$\overline{\text{RD}}$ ↓ (acknowledge) to Read Data Valid Delay	–	250	–	180	–	140	–
41	TsIEI(RDA)	IEI to $\overline{\text{RD}}$ ↓ (acknowledge) Setup Time	120	–	100	–	95	–	–
42	ThIEI(RDA)	IEI to $\overline{\text{RD}}$ ↑ (acknowledge) Hold Time	0	–	0	–	0	–	–
43	TdIEI(IEO)	IEI to IEO Delay Time	–	120	–	100	–	95	–
44	TdPC(IEO)	PLCK ↑ IEO Delay	–	250	–	250	–	200	–
45	TdRDA(INT)	$\overline{\text{RD}}$ ↓ to $\overline{\text{INT}}$ Inactive Delay	–	500	–	500	–	450	4
46	TdRD(WRQ)	$\overline{\text{RD}}$ ↑ $\overline{\text{WR}}$ ↓ Delay for no Reset	30	–	15	–	15	–	–
47	TdWRQ(RD)	$\overline{\text{WR}}$ ↑ to $\overline{\text{RD}}$ ↓ Delay for no Reset	30	–	30	–	20	–	–
48	TwRES	$\overline{\text{WR}}$ and $\overline{\text{RD}}$ Coincident Low for Reset	250	–	200	–	150	–	–
49	Trc	Valid Access Recovery Time	4TcPC	–	4TcPC	–	4TcPC	–	3

Notes : 3. Parameter applies only between transactions involving the ASCC.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any ASCC in the daisy chain, TdIAi(RD) must be greater than the sum of TdPC (IEO) for the highest priority device in the daisy chain, TsIEI(RDA) for the ASCC, and TdIEI(IEO) for each device separating them in the daisy chain.
* Timing are preliminary and subject to change.
† Units in nanoseconds (ns).

GENERAL TIMING

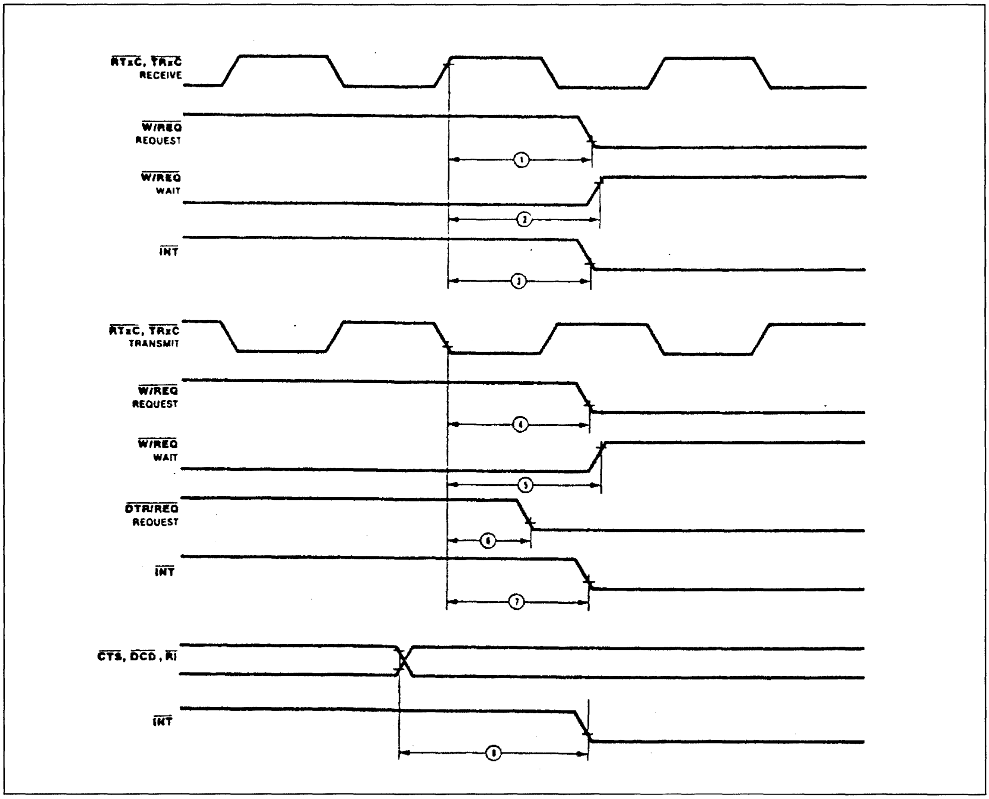


GENERAL TIMING (continued)

N°	Symbol	Parameter	4 MHz		6 MHz		8 MHz		Notes *
			Min.	Max.	Min.	Max.	Min.	Max.	†
1	TdPC(REQ)	PCLK ↓ to $\overline{W/REQ}$ Valid Delay	–	250	–	250	–	250	–
2	TdPC(W)	PCLK ↓ to Wait Inactive Delay	–	350	–	350	–	350	–
3	TsRXC(PC)	$\overline{RxC}$ ↑ to PCLK ↑ Setup Time (PCLK + 4 case only)	80	TwPCl	70	TwPCl	60	TwPCl	1.4
4	TsRXD(RXCr)	RxD to $\overline{RXC}$ ↑ Setup Time (XI mode)	0	–	0	–	0	–	1
5	ThRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Hold Time (XI mode)	150	–	150	–	150	–	1
6	TsRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Setup Time (X1 mode)	0	–	0	–	0	–	1.5
7	ThRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Hold Time (X1 Mode)	150	–	150	–	150	–	1.5
8	TsTXC(PC)	$\overline{TxC}$ ↓ to PCLK ↑ Setup Time	0	–	0	–	0	–	2.4
9	TdTXCf(TXD)	$\overline{TxC}$ ↓ to TxD Delay (X1 Mode)	–	300	–	230	–	200	2
10	TdTXCr(TXD)	$\overline{TxC}$ ↑ to TxD Delay (X1 mode)	–	300	–	230	–	200	2.5
11	TdTXD(TRX)	TxD to $\overline{TRxC}$ Delay (send clock echo)	–	200	–	200	–	200	–
12	TwRTXh	$\overline{RTxC}$ High Width	180	–	180	–	150	–	6
13	TwRTXl	$\overline{RTxC}$ Low Width	180	–	180	–	150	–	6
14	TcRTX	$\overline{RTxC}$ Cycle Time	1000	–	640	–	500	–	6
15	TcRTXX	Crystal Oscillator Period	250	1000	165	1000	125	1000	3
16	TwTRXh	$\overline{TRxC}$ High Width	180	–	180	–	150	–	6
17	TwTRXl	$\overline{TRxC}$ Low Width	180	–	180	–	150	–	6
18	TcTRX	$\overline{TRxC}$ Cycle Time	1000	–	640	–	500	–	6.7
19	TwEXT	$\overline{DCD}$ or $\overline{CTS}$ or RI Pulse Width	200	–	200	–	200	–	–

Notes : 1. RxC is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
2. TxC is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
3. Both RTxC and R1 have 30 pF capacitors ot ground connected to them.
4. Parameter applies only if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between $\overline{RxC}$ and PCLK or $\overline{TxC}$ and PCLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Parameter applies only for transmitter and receiver; DPLL and baud rate generator timing requirements are identical to chip PCLK requirements.
7. The maximum receive or transmit data is 1/4 PLCK.
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

SYSTEM TIMING



N°	Symbol	Parameter	4 MHz		6 MHz		8 MHz		Notes * †
			Min.	Max.	Min.	Max.	Min.	Max.	
1	TdRXC(REQ)	$\overline{RxC}$ ↑ to $\overline{W/REQ}$ Valid Delay	8	12	8	12	8	12	2
2	TdRXC(W)	$\overline{RxC}$ ↑ to Wait Inactive Delay	8	14	8	14	8	14	1, 2
3	TdRXC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ Valid Delay	10	16	10	16	10	16	1, 2
4	TdTXC(REQ)	$\overline{TxC}$ ↓ to $\overline{W/REQ}$ Valid Delay	5	8	5	8	5	8	3
5	TdTXC(W)	$\overline{TxC}$ ↓ to Wait Inactive Delay	5	11	5	11	5	11	1, 3
6	TdTXC(DRQ)	$\overline{TxC}$ ↓ to $\overline{DRT/REQ}$ Valid Delay	4	7	4	7	4	7	1, 3
7	tdTXC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ Valid Delay	6	10	6	10	6	10	1, 3
8	TdEXT(INT)	$\overline{DCD}$ or CTS Transition to INT Valid Delay	2	6	2	6	2	6	1

Notes : 1. Open-drain output, measured with open-drain test load.
2. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
3. TxC is TRxC or RTxC, whichever is supplying the transmit clock.
* Timings are preliminary and subject to change.
† Units equal to TcPC.

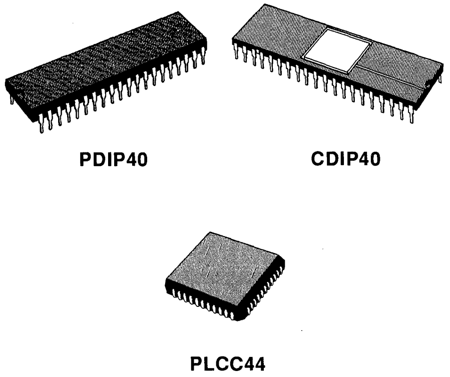
ORDERING INFORMATION

Sales Type	Frequency	Temp. Range	Package
Z8531B1V	4MHz	0 to + 70°C	PDIP-40
Z8531AB1V	6MHz	0 to + 70°C	PDIP-40
Z8531BB1V	8MHz	0 to + 70°C	PDIP-40
Z8531B6V	4MHz	– 40 to + 85°C	PDIP-40
Z8531AB6V	6MHz	– 40 to + 85°C	PDIP-40
Z8531BB6V	8MHz	– 40 to + 85°C	PDIP-40
Z8531C1V	4MHz	0 to + 70°C	PLCC44
Z8531AC1V	6MHz	0 to + 70°C	PLCC44
Z8531BC1V	8MHz	0 to + 70°C	PLCC44
Z8531C6V	4MHz	– 40 to + 85°C	PLCC44
Z8531AC6V	6MHz	– 40 to + 85°C	PLCC44
Z8531BC6V	8MHz	– 40 to + 85°C	PLCC44
Z8531D1N	4MHz	0 to + 70°C	CDIP-40
Z8531AD1N	6MHz	0 to + 70°C	CDIP-40
Z8531BD1N	8MHz	0 to + 70°C	CDIP-40
Z8531D6N	4MHz	– 40 to + 85°C	CDIP-40
Z8531AD6N	6MHz	– 40 to + 85°C	CDIP-40
Z8531BD6N	8MHz	– 40 to + 85°C	CDIP-40
Z8531D2N	4MHz	– 55 to + 125°C	CDIP-40
Z8531AD2N	6MHz	– 55 to + 125°C	CDIP-40

Note : PDIP = Plastic DIP ; CDIP = Ceramic Multilayer DIP ; PLCC = Plastic Leaded Chip Carrier.

CIO COUNTER/TIMER AND PARALLEL I/O UNIT

- TWO INDEPENDENT 8-BIT, DOUBLE-BUFFERED, BIDIRECTIONAL I/O PORTS PLUS A 4-BIT SPECIAL-PURPOSE I/O PORTS. I/O PORTS FEATURE PROGRAMMABLE POLARITY, PROGRAMMABLE DIRECTION (Bit mode), "PULSE CATCHERS", AND PROGRAMMABLE OPEN-DRAIN OUTPUTS
- FOUR HANDSHAKE MODES, INCLUDING 3-WIRE (like the IEEE-488)
- REQUEST/$\overline{\text{WAIT}}$ SIGNAL FOR HIGH-SPEED DATA TRANSFER
- FLEXIBLE PATTERN-RECOGNITION LOGIC, PROGRAMMABLE AS A 16-VECTOR INTERRUPT CONTROLLER
- THREE INDEPENDENT 16-BIT COUNTER/TIMERS WITH UP TO FOUR EXTERNAL ACCESS LINES PER COUNTER/TIMER (count input, output, gate, and trigger), AND THREE OUTPUT DUTY CYCLES (pulsed, one-shot, and square-wave), PROGRAMMABLE AS RETRIGGERABLE OR NONRETRIGGERABLE
- EASY TO USE SINCE ALL REGISTERS ARE READ/WRITE



(Ordering Information at the end of the datasheet)

DESCRIPTION

The Z8536 CIO Counter/Timer and Parallel I/O element is a general-purpose peripheral circuit, satisfying most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications. The use of the device is simplified by making all internal registers (command, status, and data) readable and (except for status bits) writable. In addition, each register is given its own unique internal address, so that any register can be accessed in two operations. All data registers can be directly accessed in a single operation. The CIO is easily interfaced to all popular microprocessors.

Figure 1 : Logic Functions.

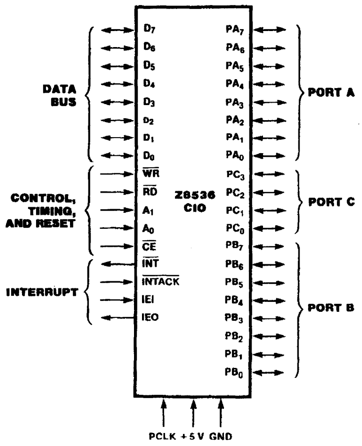


Figure 2a : DIP, Pin Connections.

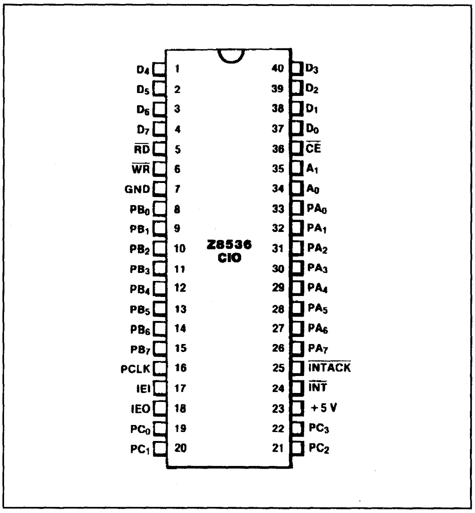
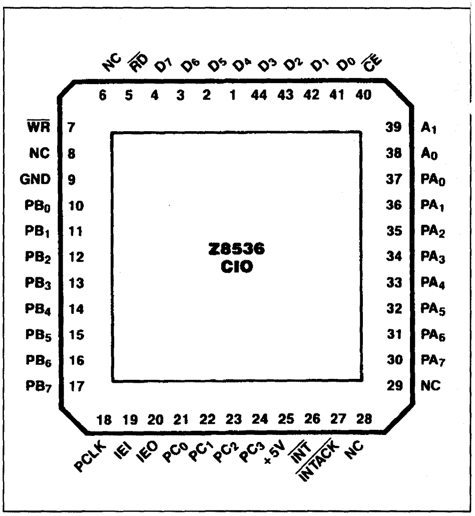


Figure 2b : Chip Carrier, Pin Connections.



PIN DESCRIPTION

A_0-A_1. *Address Lines* (input). These two lines are used to select the register involved in the CPU transaction : Port A's Data register, Port B's Data register, Port C's Data register, or a control register.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level on this input enables the CIO to be read from or written to.

D_0-D_7. *Data Bus* (bidirectional 3-state). These eight data lines are used for transfers between the CPU and the CIO.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting CIO or is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is pulled Low when the CIO requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This input indicates to the CIO that an Interrupt Acknowledge cycle is in progress. $\overline{INTACK}$ must be synchronized to PCLK, and it must be stable throughout the Interrupt Acknowledge cycle.

PA_0-PA_7. *Port A I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the CIO's Port A and external devices.

PB_0-PB_7. *Port B I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the CIO's Port B and external devices. May also be used to provide external access to Counter/Timers 1 and 2.

PC_0-PC_3. *Port C I/O lines* (bidirectional, 3-state, or open-drain). These four I/O lines are used to provide handshake, $\overline{WAIT}$, and $\overline{REQUEST}$ lines for Ports A and B or to provide external access to Counter/Timer 3 or access to the CIO's Port C.

PCLK. *Peripheral Clock* (input, TTL-compatible). This is the clock used by the internal control logic and the counter/timers in timer mode. It does not have to be the CPU clock.

$\overline{RD}$*. *Read* (input, active Low). This signal indicates that a CPU is reading from the CIO. During an Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the data bus if the CIO is the highest priority device requesting an interrupt.

$\overline{WR}$*. *Write* (input, active Low). This signal indicates a CPU write to the CIO.

* When $\overline{RD}$ and $\overline{WR}$ are delected low at the same time (normally an illegal condition), the CIO is reset.

ARCHITECTURE

The CIO Counter/Timer and Parallel I/O element (figure 3) consists of a CPU interface, three I/O ports (two general-purpose 8-bit ports and one special-purpose 4-bit port), three 16-bit counter/timers, an interrupt-control logic block, and the internal-control logic block. An extensive number of programmable options allow the user to tailor the configuration to best suit the specific application.

The two general-purpose 8-bit I/O ports (figure 4) are identical, except that Port B can be specified to provide external access to Counter/Timers 1 and 2. Either port can be programmed to be a handshake-driven, double-buffered port (input, output, or bidirectional) or a control-type port with the direction of each bit individually programmable. Each port includes pattern-recognition logic, allowing interrupt generation when a specific pattern is detected. The pattern-recognition logic can be programmed so the port functions like a priority-interrupt controller. Ports A and B can also be linked to form a 16-bit I/O port.

To control these capabilities, both ports contain 12 registers. Three of these registers, the Input, Output, and Buffer registers, comprise the data path registers. Two registers, the Mode Specification and Handshake Specification registers, are used to define the mode of the port and to specify which handshake, if any, is to be used. The reference pattern for the pattern-recognition logic is defined via three registers : the Pattern Polarity, Pattern Transition, and Pattern Mask registers. The detailed characteristics of each bit path (for example, the direction of data flow or whether a path is inverting or noninverting) are programmed using the Data Path Polarity, Data Direction, and Special I/O Control registers.

The primary control and status bits are grouped in a single register, the Command and Status register, so that after the port is initially configured, only this register must be accessed frequently. To facilitate initialization, the port logic is designed so that registers associated with an unrequired capability are ignored and do not have to be programmed.

Figure 3 : CIO Block Diagram.

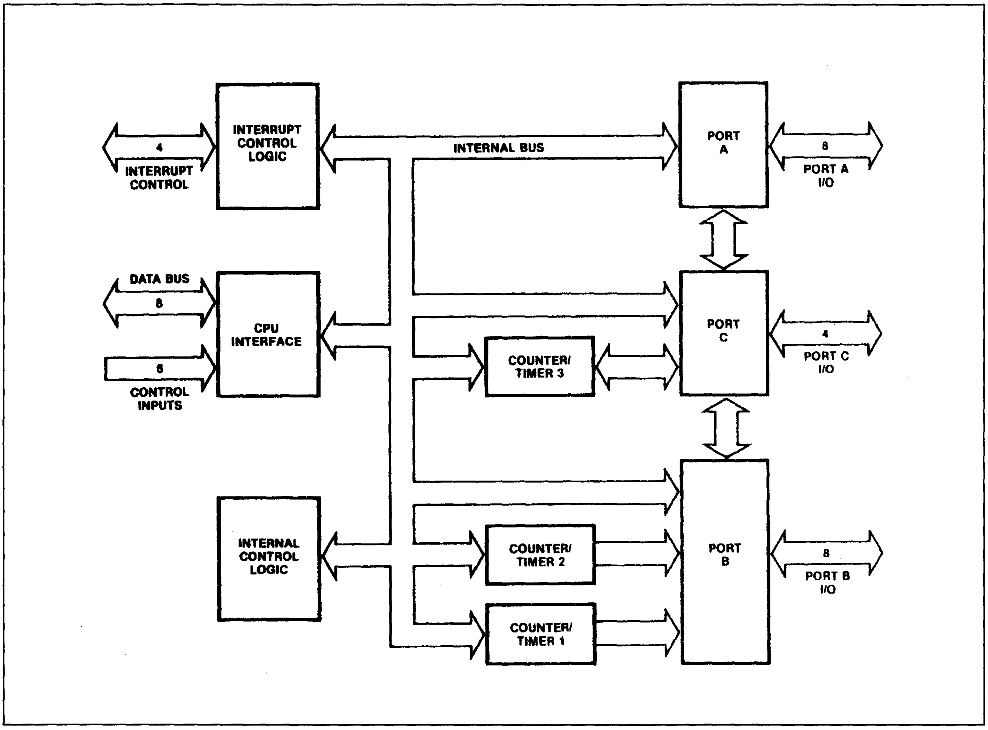
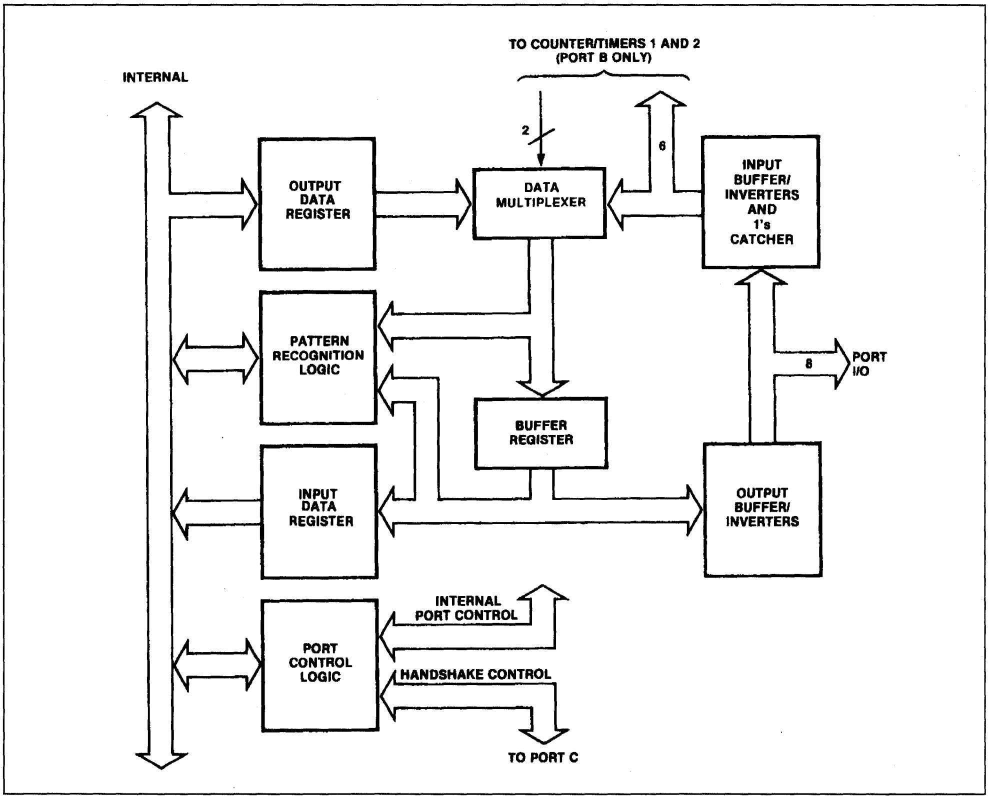


Figure 4 : Ports A and B Block Diagram.



The function of the special-purpose 4-bit port, Port C (figure 5), depends upon the roles of' Ports A and B. Port C provides the required handshake lines. Any bits of Port C not used as handshake lines can be used as I/O lines or to provide external access for the third counter/timer.

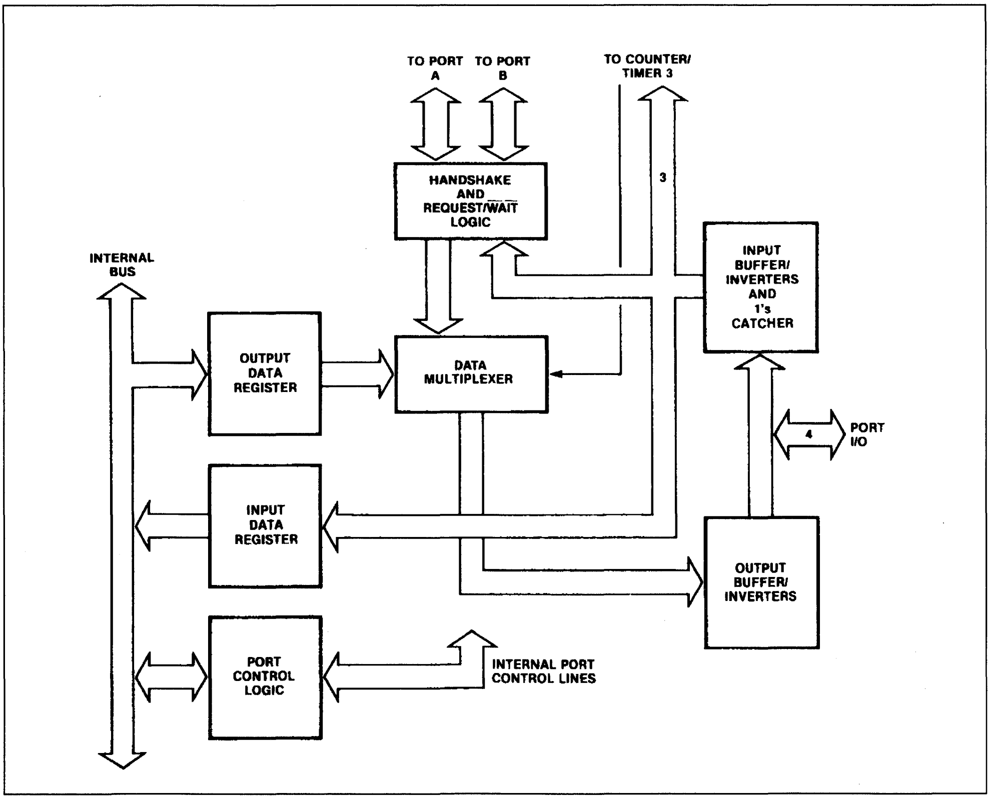
Since Port C's function is defined primarily by Ports A and B, only three registers (besides the Data Input and Output registers) are needed. These registers specify the details of each bit path : the Data Path Polarity, Data Direction, and Special I/O Control registers.

The three counter/timers (figure 6) are all identical. Each is comprised of a 16-bit down-counter, a 16-bit Time Constant register (which holds the value loaded into the down-counter), a 16-bit Current Count register (used to read the contents of the down-counter), and two 8-bit registers for control and status (the Mode Specification and the Command and Status registers).

The capabilities of the counter/timer are numerous. Up to four port I/O lines can be dedicated as external access lines for each counter/timer : counter input, gate input, trigger input, and counter/timer output. Three different counter/timer output duty cycles are available : pulse, one-shot, or square-wave. The operation of the counter/timer can be programmed as either retriggerable or non-retriggerable. With these and other options, most counter/timer applications are covered.

There are five registers (Master Interrupt Control register, three Interrupt Vector registers, and the Current Vector register) associated with the interrupt logic. In addition, the ports' Command and Status registers and the counter/timers' Command and Status registers include bits associated with the interrupt logic. Each of these registers contains three bits for interrupt control and status : Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE).

Figure 5 : Port C Block Diagram.



FUNCTIONAL DESCRIPTION

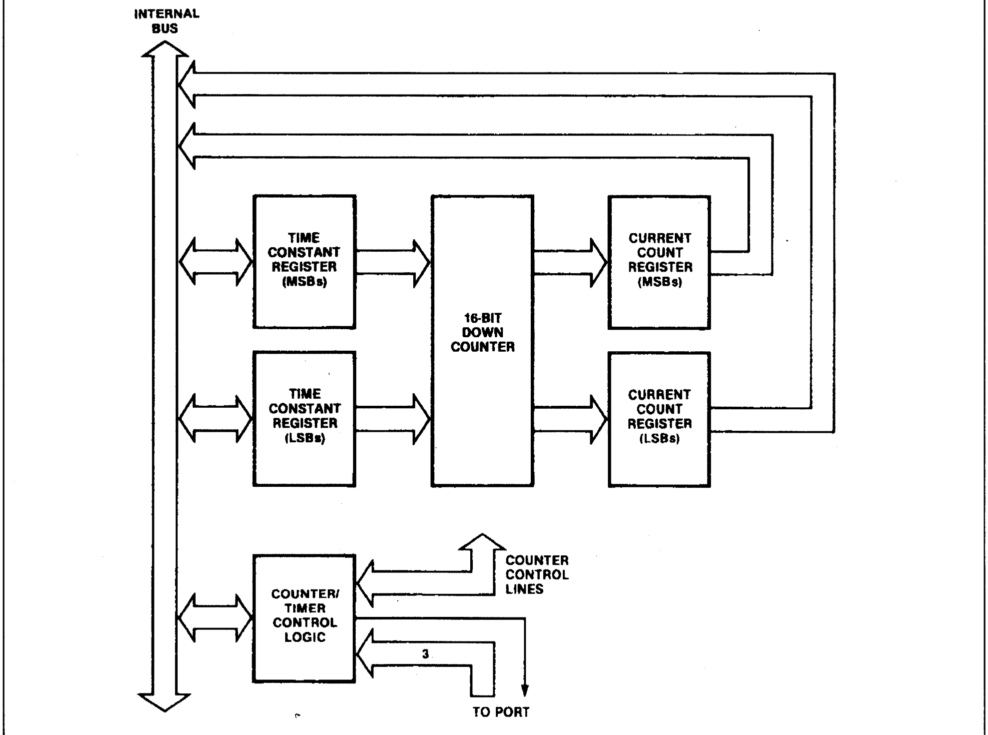
The following describes the functions of the ports, pattern-recognition logic, counter/timers, and interrupt logic.

I/O Port Operations. Of the CIO's three I/O ports, two (Ports A and B) are general-purpose, and the third (Port C) is a special-purpose 4-bit port. Ports A and b can be configured as input, output, or bidirectional ports with handshake. (Four different handshakes are available). They can also be linked to form a single 16-bit port. If they are not used as ports with handshake, they provide 16 input or output bits with the data direction programmable on a bit-by-bit basis. Port B also provides access for Counter/Timers 1 and 2. In all configurations, Ports A and B can be programmed to recognize specific data patterns and to generate interrupts when the pattern is encountered.

The four bits of Port C provide the handshake lines for Ports A and B when required. A REQUEST/WAIT line can also be provided so that CIO transfers can be synchronized with DMAs or CPUs. Any Port C bits not used for handshake or REQUEST/WAIT can be used as input or output bits (individually data-direction programmable) or external access lines for Counter/Timer 3. Port C does not contain any pattern-recognition logic. It is, however, capable of bit-addressable writes. With this feature, any combination of bits can be set and/or cleared while the other bits remain undisturbed without first reading the register.

Bit Port Operations. In bit port operations, the port's Data Direction register specifies the direction of data flow for each bit. A 1 specifies an input bit, and a 0 specifies an output bit. If bits are used as I/O bits for a counter/timer, they should be set as input or output, as required.

Figure 6 : Counter/Timer Block Diagram.


INTERNAL BUS
TIME CONSTANT REGISTER (MSBs)
TIME CONSTANT REGISTER (LSBs)
16-BIT DOWN COUNTER
CURRENT COUNT REGISTER (MSBs)
CURRENT COUNT REGISTER (LSBs)
COUNTER/ TIMER CONTROL LOGIC
COUNTER CONTROL LINES
3
TO PORT

The Data Path Polarity register provides the capability of inverting the data path. A 1 specifies inverting, and a 0 specifies noninverting. All discussions of the port operations assume that the path is noninverting.

The value returned when reading an input bit reflects the state of the input just prior to the read. A 1's catcher can be inserted into the input data path by programming a 1 to the corresponding bit position of the port's Special I/O Control register. When a 1 is detected at the 1's catcher input, its output is set to 1 until it is cleared. The 1's catcher is cleared by writing a 0 to the bit. In all other cases, attempted writes to input bits are ignored.

When Ports A and B include output bits, reading the Data register returns the value being output. Reads of Port C return the state of the pin. Outputs can be specified as open-drain by writing a 1 to the corresponding bit of the port's Special I/O Control register. Port C has the additional feature of bit-addressable writes. When writing to Port C, the four most significant bits are used as a write protect mask for the least significant bits (0-4, 1-5, 2-6, and 3-7). If the write protect bit is written with a 1, the state of the corresponding output bit is not changed.

Ports with Handshake Operation. Ports A and B can be specified as 8-bit input, output, or bidirectional ports with handshake. The CIO provides four different handshakes for its ports : Interlocked, Strobed, Pulsed, and 3-Wire. When specified as a port with handshake, the transfer of data into and out of the port and interrupt generation is under control of the handshake logic. Port C provides the handshake lines as shown in table 1. Any Port C lines not used for handshake can be used as simple I/O lines or as access lines for Counter/Timer 3;

When Ports A and B are configured as ports with handshake, they are double-buffered. This allows for more relaxed interrupt service routine response time. A second byte can be input to or output from

FUNCTIONAL DESCRIPTION (cont'd)

the port before the interrupt for the first byte is serviced. Normally, the Interrupt Pending (IP) bit is set and an interrupt is generated when data is shifted into the Input register (input port) or out of the Output register (output port). For input and output ports, the IP is automatically cleared when the data is read or written. In bidirectional ports, IP is cleared only by command. When the Interrupt on Two Bytes (ITB) control bit is set to 1, interrupts are generated only when two bytes of data are available to be read or written. This allows a minimum of 16 bits of information to be transferred on each interrupt. With ITB set, the IP is not automatically cleared until the second byte of data is read or written.

When the Single Buffer (SB) bit is set to 1, the port acts as if it is only single-buffered. This is useful if the handshake line must be stopped on a byte-by-byte basis.

Ports A and B can be linked to form a 16-bit port by programming a 1 in the Port Link Control (PLC) bit. In this mode, only Port A's Handshake Specification and Command and Status registers are used. Port B must be specified as a bit port. When linked, only Port A has pattern-match capability. Port B's pattern-match capability must be disabled. Also, when the ports are linked. Port B's Data register must be read or written before Port A's.

When a port is specified as a port with handshake, the type of port it is (input, output, or bidirectional) determines the direction of data flow. The data direction for the bidirectional port is determined by a bit in Port C (table 1). In all cases, the contents of the Data Direction register are ignored. The contents of the Special I/O Control register apply only to output bits (3-state or open-drain). Inputs may not have 1's catchers ; therefore, those bits in the Special I/O Control register are ignored. Port C lines used for handshake should be programmed as inputs. The handshake specification overrides Port C's Data Direction register for bits that must be outputs. The contents of Port C's Data Path Polarity register still apply.

Interlocked Handshake. In the Interlocked Handshake mode, the action of the CIO must be acknowledged by the external device before the next action can take place. Figure 7 shows timing for Interlocked Handshake. An output port does not indicate that new data is available until the external device indicates it is ready for the data. Similarly, an input port does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging the input port's acceptance of the last byte. This allows the CIO to interface directly to the port of a Z8 microcomputer, a UPC, an FIO, an FIFO, or to another CIO port with no external logic.

A 4-bit deskew timer can be inserted in the Data Available ($\overline{\text{DAV}}$) output for output ports. As data is transferred to the Buffer register, the deskew timer is triggered. After the number of PCLK cycles specified by the deskew timer time constant plus one, $\overline{\text{DAV}}$ is allowed to go Low. The deskew timer therefore guarantees that the output data is valid for a specified minimum amount of time before $\overline{\text{DAV}}$ goes Low. Deskew timers are available for output ports independent of the type of handshake employed.

Strobed Handshake. In the Strobed Handshake mode, data is "strobed" into or out of the port by the external logic. The falling edge of the Acknowledge Input (ACKIN) strobes data into or out of the port. Figure 7 shows timing for the Strobed Handshake. In contrast to the Interlocked handshake, the signal indicating the port is ready for another data transfer operates independently of the ACKIN input. It is up to the external logic to ensure that data overflows or underflows do not occur.

3-Wire Handshake. The 3-Wire Handshake is designed for the situation in which one output port is communicating with many input ports simultaneously. It is essentially the same as the Interlocked Handshake, except that two signals are used to indicate if an input port is ready for new data or if it has accepted the present data. In the 3-Wire Handshake (figure 8), the rising edge of one status line indicates that the port is ready for data, and the rising edge of another status line indicates that the data has been accepted. With the 3-Wire Handshake, the output lines of many input ports can be bussed together with open-drain drivers ; the output port knows when all the ports have accepted the data and are ready. This is the same handshake as is used on the IEEE-488 bus. Because this handshake requires three lines, only one port (either A or B) can be a 3-Wire Handshake port at a time. The 3-Wire Handshake is not available in the bidirectional mode. Because the port's direction can be changed under software control, however, bidirectional IEEE-488-type transfers can be performed.

Pulsed Handshake. The Pulsed Handshake (figure 9) is designed to interface to mechanical-type devices that require data to be held for long periods of time and need relatively wide pulses to gate the data into or out of the device. The logic is the same as the Interlocked Handshake mode, except that an internal counter/timer is linked to the handshake logic. If the port is specified in the input mode, the timer is inserted in the ACKIN path. The external ACKIN input triggers the timer and its output is used

FUNCTIONAL DESCRIPTION (cont'd)

as the Interlocked Handshake's normal acknowledge input. If the port is an output port, the timer is placed in the Data Available ($\overline{\text{DAV}}$) output path. The timer is triggered when the normal Interlocked Handshake $\overline{\text{DAV}}$ output goes Low and the timer output is used as the actual $\overline{\text{DAV}}$ output. The counter/timer maintains all of its normal capabilities. This handshake is not available to bidirectional ports.

REQUEST/$\overline{\text{WAIT}}$ Line Operation. Port C can be programmed to provide a status signal output in addition to the normal handshake lines for either Port A or B when used as a port with handshake. The additional signal is either a REQUEST or $\overline{\text{WAIT}}$ signal. The REQUEST signal indicates when a port is ready to perform a data transfer via the CPU interface. It is intended for use with a DMA-type device. The $\overline{\text{WAIT}}$ signal provides synchronization for transfers with a CPU. Three bits in the Port Handshake Specification register provide controls for the REQUEST/$\overline{\text{WAIT}}$ logic. Because the extra Port C line is used, only one port can be specified as a port with a handshake and a REQUEST/$\overline{\text{WAIT}}$ line. The other port must be a bit port.

Operation of the REQUEST line is modified by the state of the port's Interrupt on Two Bytes (ITB) control bit. When ITB is 0, the REQUEST line goes active as soon as the CIO is ready for a data transfer. If ITB is 1, REQUEST does not go active until two bytes can be transferred. REQUEST stays active as long as a byte is available to be read or written.

The SPECIAL REQUEST function is reserved for use with bidirectional ports only. In this case, the REQUEST line indicates the status of the register not being used in the data path at that time. If the IN/$\overline{\text{OUT}}$ line is High, the REQUEST line is High when the Output register is empty. If IN/$\overline{\text{OUT}}$ is Low, the REQUEST line is High when the Input register is full.

Pattern-Recognition Logic Operation. Both Ports A and B can be programmed to generate interrupts when a specific pattern is recognized at the port. The pattern-recognition logic is independent of the port application, thereby allowing the port to recognize patterns in all of its configurations. The pattern can be independently specified for each bit as 1, 0, rising edge, falling edge, or any transition. Individual bits may be masked off. A pattern-match is defined as the simultaneous satisfaction of all non-masked bit specifications in the AND mode or the satisfaction of any non-masked bit specifications in either of the OR or OR-Priority Encoded Vector modes.

Table 1 : Port C Bit Utilization.

Port A/B Configuration	**PC_3**	**PC_2**	**PC_1**	**PC_0**
Ports A and B : Bits Ports	Bit I/O	Bit I/O	Bit I/O	Bit I/O
Port A : Input or Output Port (interlocked, strobed, or pulsed handshake) *	RFD or $\overline{\text{DAV}}$	$\overline{\text{ACKIN}}$	REQUEST/$\overline{\text{WAIT}}$ or Bit I/O	Bit I/O
Port B : Input or Output Port (interlocked, strobed, or pulsed handshake) *	REQUEST/$\overline{\text{WAIT}}$ or Bit I/O	Bit I/O	RFD or $\overline{\text{DAV}}$	$\overline{\text{ACKIN}}$
Port A or B : Input Port (3-wire handshake)	RFD (output)	$\overline{\text{DAV}}$ (input)	REQUEST/$\overline{\text{WAIT}}$ or Bit I/O	DAC (output)
Port A or B : Output Port (3-wire handshake)	$\overline{\text{DAV}}$ (output)	DAC (input)	REQUEST/$\overline{\text{WAIT}}$ or Bit I/O	RFD (input)
Port A or B : Bidirectional Port (interlocked or strobed handshake)	RFD or $\overline{\text{DAV}}$	$\overline{\text{ACKIN}}$	REQUEST/$\overline{\text{WAIT}}$ or Bit I/O	IN/$\overline{\text{OUT}}$

* Both Ports A and B can be specified input or output with interlocked, Strobed, or Pulsed Handshake at the same time if neither uses REQUEST/$\overline{\text{WAIT}}$.

Figure 7 : Interlocked and Strobed Handshakes.

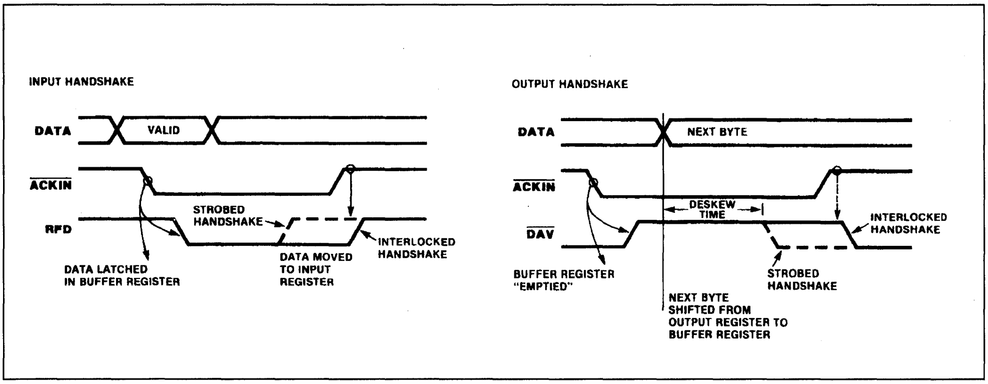


Figure 8 : 3-wire Handshake.

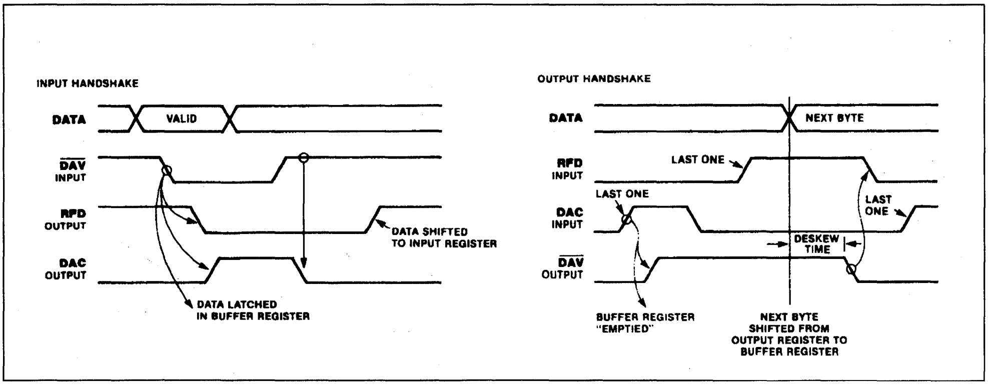
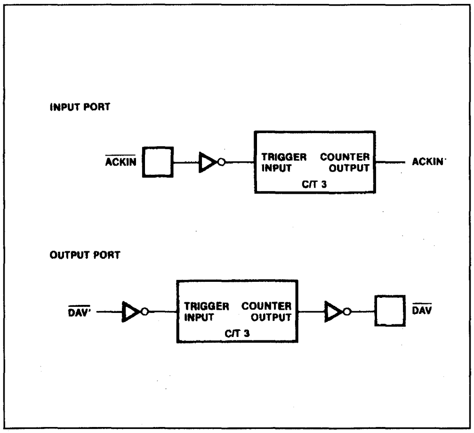


Figure 9 : Pulsed Handshake.



The pattern specified in the Pattern Definition register assumes that the data path is programmed to be noninverting. If an input bit in the data path is programmed to be inverting, the pattern detected is the opposite of the one specified. Output bits used in the pattern-match logic are internally sampled before the invert/noninvert logic.

Bit Port Pattern-Recognition Operations. During bit port operations, pattern-recognition may be performed on all bits, including those used as I/O for the counter/timers. The input to the pattern-recognition logic follows the value at the pins (through the invert/noninvert logic) in all cases except for simple inputs with 1's catchers. In this case, the output of the 1's catcher is used. When operating in the AND or OR mode, it is the transition from a no-match to a match state that causes the interrupt. In the "OR" mode, if a second match occurs before the first match goes away, it does not cause an interrupt.

FUNCTIONAL DESCRIPTION (cont'd)

Since a match condition only lasts a short time when edges are specified, care must be taken to avoid losing a match condition. Bit ports specified in the OR-Priority Encoded Vector mode generate interrupts as long as any match state exists. A transition from a no-match to a match state is not required.

The pattern-recognition logic of bit ports operates in two basic modes : transparent and latched. When the Latch on Pattern Match (LPM) bit is set to 0 (Transparent mode), the interrupt indicates that a specified pattern has occurred, but a read of the Data register does not necessarily indicate the state of the port at the time the interrupt was generated. In the Latched mode (LPM = 1), the state of all the port inputs at the time the interrupt was generated is latched in the input register and held until IP is cleared. In all cases, the PMF indicates the state of the port at the time it is read.

If a match occurs while IP is already set, an error condition exists. If the Interrupt On Error bit (IOE) is 0, the match is ignored. However, if IOE is 1 after the first IP is cleared, it is automatically set to 1 along with the Interrupt Error (ERR) flag. Matches occurring while ERR is set are ignored. ERR is cleared when the corresponding IP is cleared.

When a pattern-match is present in the OR-Priority Encoded Vector mode, IP is set to 1. The IP cannot be cleared until a match is no longer present. If the interrupt vector is allowed to include status, the vector returned during Interrupt Acknowledge indicates the highest priority bit matching its specification at the time of the Acknowledge cycle. Bit 7 is the highest priority and bit 0 is the lowest. The bit initially causing the interrupt may not be the one indicated by the vector if a higher priority bit matches before the Acknowledge. One the Acknowledge cycle is initiated, the vector is frozen until the corresponding IP is cleared. Where inputs that cause interrupt is serviced, the 1's catcher can be used to hold the value.

Because a no-match to match transition is not required, the source of the interrupt must be cleared before IP is cleared or else a second interrupt is generated. No error detection is performed in this mode, and the Interrupt On Error bit should be set to 0.

Ports with Handshake Pattern-Recognition Operation. In this mode, the handshake logic normally controls the setting of IP and, therefore, the generation of interrupt requests. The pattern-match logic controls the Pattern-Match Flag (PMF). The data is compared with the match pattern when it is shifted from the Buffer register (input port) or when it is shifted from the Output register to the Buffer register (output port). The pattern match logic can override the handshake logic in certain situations. If the port is programmed to interrupt when two bytes of data are available to be read or written, but the first byte matches the specified pattern, the pattern-recognition logic sets IP and generates an interrupt. While PMF is set, IP cannot be cleared by reading or writing the data registers. IP must be cleared by command. The input register is not emptied while IP is set, nor is the output register filled until IP is cleared.

If the Interrupt on Match Only (IMO) bit is set, IP is set only when the data matches the pattern. This is useful in DMA-type application when interrupts are required only after a block of data is transferred.

Counter/Timer Operation. The three independent 16-bit counter/timers consist of a presettable 16-bit down counter, a 16-bit Time Constant register, a 16-bit Current Counter register, an 8-bit Mode Specification register, an 8-bit Command and Status register, and the associated control logic that links these registers.

Table 2 : Counter/Timer External Access.

Function	C/T_1	C/T_2	C/T_3
Counter/Timer Output	PB 4	PB 0	PC 0
Counter Input	PB 5	PB 1	PC 1
Trigger Input	PB 6	PB 2	PC 2
Gate Input	PB 7	PB 3	PC 3

The flexibility of the counter/timers is enhanced by the provision of up to four lines per counter/timer (counter input, gate input, trigger input, and counter/timer output) for direct external control and status. Counter/Timer 1's external I/O lines are provided by the four most significant bits of Port B. Counter/Timer 2's are provided by the four significant bits of Port B. Counter/Timer 3's external I/O lines are provided by the four bits of Port C. The utilization of these lines (table 2) is programmable on a bit-by-bit basis via the Counter/Timer Mode Specification registers.

When external counter/timer I/O lines are to be used, the associated port lines must be vacant and programmed in the proper data direction. Lines used for counter/timer I/O have the same characteristics as simple input lines. They can be specified as inverting or noninverting ; they can be read and used with the pattern-recognition logic. They can also include the 1's catcher input.

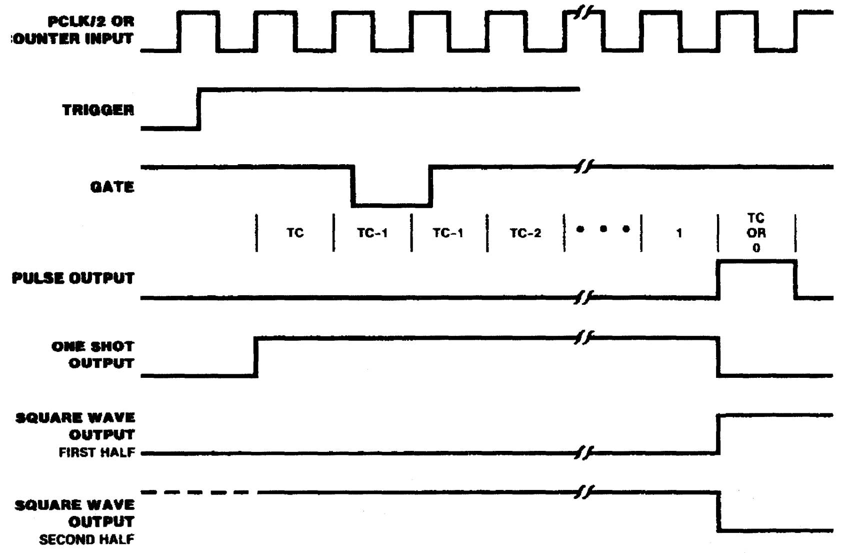
Counter/Timers 1 and 2 can be linked internally in three different ways. Counter/Timer 1's output (inverted) can be used as Counter/Timer 2's trigger, gate, or counter input. When linked, the counter/ timers have the same capabilities as when used separately. The only restriction is that when Counter/Timer 1 drives Counter/Timer 2's count input, Counter/Timer 2 must be programmed with its external count input disabled.

There are three duty cycles available for the timer/counter output : pulse, one-shot, and square-wave. Figure 10 shows the counter/timer waveforms. When the Pulse mode is specified, the output goes High for one clock cycle, beginning when the down-counter leaves the count of 1. In the One-Shot mode, the output goes High when the counter/timer is triggered and goes Low when the down-counter reaches 0. When the square-wave output duty cycle is specified, the counter/timer goes through two full sequences for each cycle. The initial trigger causes the down-counter to be loaded and the normal count-down sequence to begin. If a 1 count is detected on the down-counter's clocking edge, the output goes High and the time constant value is reloaded. On the clocking edge, when both the down-counter and the output are 1's, the output is pulled back Low.

The Continuous/Single Cycle (C/$\overline{SC}$) bit in the Mode Specification register controls operation of the down-counter when it reaches terminal count. If C/$\overline{SC}$ is 0 when a terminal count is reached, the countdown sequence stops. If the C/$\overline{SC}$ bit is 1 each time the count-down counter reaches 1, the next cycle causes the time constant value to be reloaded. The time constant value may be changed by the CPU, and on reload, the new time constant value is loaded.

Counter/timer operations require loading the time constant value in the Time Constant register and initiating the countdown sequence by loading the down-counter with the time constant value. The Time Constant register is accessed as two 8-bit registers. The registers are readable as well as writable, and the access order is irrelevant. A 0 in the Time Constant register specifies a time constant of 65.536. The down-counter is loaded in one of three ways : by writing a 1 to the Trigger Command Bit (TCB) of the Command and Status register, on the rising edge of the external trigger input, or, for Counter/Timer 2 only, on the rising edge of Counter/Timer 1's internal outpout if the counters are linked via the trigger input. The TCB is write-only, and read always returns 0.

Figure 10 : Counter/Timer Waveforms.



Once the down-counter is loaded, the count-down sequence continues toward terminal count as long as all the counter/timer's hardware and software gate inputs are High. If any of the gate inputs goes Low (0), the countdown halts. It resumes when all gate inputs are 1 again.

The reaction to triggers occurring during a count-down sequence is determined by the state of the Retrigger Enable Bit (REB) in the Mode Specification register. If REB is 0, retriggers are ignored and the countdown continues normally. If REB is 1, each trigger causes the down-counter to be reloaded and the count-down sequence starts over again. If the output is programmed in the Square-Wave mode, retrigger causes the sequence to start over from the initial load of the time constant.

The rate at which the down-counter counts is determined by the mode of the counter/timer. In the Timer mode (the External Count Enable [ECE] bit is 0), the down-counter is clocked internally by a signal that is half the frequency of the PCLK input to the chip. In the Counter mode (ECE is 1), the down-counter is decremented on the rising edge of the counter/timer's counter input.

Each time the counter reaches terminal count, its Interrupt Pending (IP) bit is set to 1, and if interrupts are enabled (IE = 1), an interrupt is generated. If a terminal count occurs while IP is already set, an internal error flag is set. As soon as IP is cleared, it is forced to 1 along with the Interrupt Error (ERR) flag. Errors that occur after the internal flag is set are ignored.

The state of the down-counter can be determined in two ways : by reading the contents of the down-counter via the Current Count register or by testing the Count In Progress (CIP) status bit in the Command and Status register. The CIP status bit is set when the down-counter is loaded ; it is reset when the down-counter reaches 0. The Current Count register is a 16-bit register, accessible as two 8-bit registers, which mirrors the contents of the down-counter. This register can be read anytime. However, reading the register is asynchronous to the counter's counting, and the value returned is valid only if the counter is stopped. The down-counter can be reliably read "on the fly" by the first writing of a 1 to the Read Counter Control (RCC) bit in the counter/timer's Command and Status register. This freezes the value in the Current Count register until a read of the least significant byte is performed.

Interrupt Logic Operation. The CIO has five potential sources of interrupts : the three counter/timers and Ports A and B. The priorities of these sources are fixed in the following order : Counter/Timer 3, Port A, Counter/Timer 2, Port B, and Counter/Timer 1. Since the counter/timers all have equal capabilities and Ports A and B have equal capabilities, there is no adverse impact from the relative priorities.

The CIO interrupt priority, relative to other components within the system, is determined by an interrupt daisy chain. Two pins, Interrupt Enable In (IEI) and Interrupt Enable Out (IEO), provide the input and output necessary to implement the daisy chain. When IEI is pulled Low by a higher priority device, the CIO cannot request an interrupt of the CPU. The following discussion assumes that the IEI line is High.

Each source of interrupt in the CIO contains three bits for the control and status of the interrupt logic : an Interrupt Pending (IP) status bit, an Interrupt Under Service (IUS) status bit, and an Interrupt Enable (IE) control bit. IP is set when an event requiring CPU intervention occurs. The setting of IP results in forcing the Interrupt ($\overline{\text{INT}}$) output Low, if the associated IE is 1.

The IUS status bit is set as a result of the Interrupt Acknowledge cycle by the CPU and is set only if its IP is of highest priority at the time the Interrupt Acknowledge commences. It can also be set directly by the CPU. Its primary function is to control the interrupt daisy chain. When set, it disables lower priority sources in the daisy chain, so that lower priority interrupt sources do not request servicing while higher priority devices are being serviced.

The IE bit provides the CPU with a means of masking off individual sources of interrupts. When IE is set to 1, interrupt is generated normally. When IE is set to 0, the IP bit is set when an event occurs that would normally require service ; however, the $\overline{\text{INT}}$ output is not forced Low.

The Master Interrupt Enable (MIE) bit allows all sources of interrupts within the CIO to be disabled without having to individually set each IE to 0. If MIE is set to 0, all IPs are masked off and no interrupt can be requested or acknowledged. The Disable Lower Chain (DLC) bit is included to allow the CPU to modify the system daisy chain. When the DLC bit is set to 1, the CIO's IEO is forced Low, independent of the state of the CIO or its IEI input, and all lower priority device's interrupts are disabled.

As part of the Interrupt Acknowledge cycle, the CIO is capable of responding with an 8-bit interrupt vector that specifies the source of the interrupt. The CIO contains three vector registers : one for Port A, one for Port B, and one shared by the three counter/timers. The vector output is inhibited by setting the No Vector (NV) control bit to 1.

The vector output can be modified to include status information to pinpoint more precisely the cause of interrupt. Whether the vector includes status or not is controlled by a Vector Includes Status (VIS) control bit. Each base vector has its own VIS bit and is controlled independently. When MIE = 1, reading the base vector register always includes status, independent of the state of the VIS bit. In this way, all the information obtained by the vector, including status, can be obtained with one additional instruction when VIS is set to 0. When MIE = 0, reading the vector register returns the unmodified base vector so that it can be verified. Another register, the Current Vector register, allows use of the CIO in a polled environment. When read, the data returned is the same as the interrupt vector that would be output in an acknowledge, based on the highest priority IP set. If no unmasked IPs are set, the value FF_H is returned. The Current Vector register is read-only.

PROGRAMMING

The data registers within the CIO are directly accessed by address lines A_0 and A_1 (table 3). All other internal registers are accessed by the following two-step sequence, with the address lines specifying a control operation. First, write the address of the target register to an internal 6-bit Pointer Register ; then read from or write to the target register. The Data registers can also be accessed by this method.

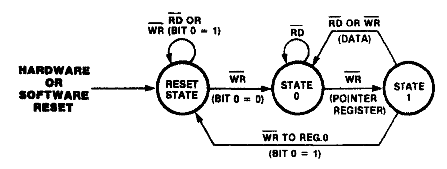
An internal state machine determines if accesses with A_0 and A_1 equalling 1 are to the Pointer Register or to an internal control register (figure 11). Following any control read operation, the state machine is in State 0 (the next control access is to the Pointer Register). This can be used to force the state machine into a known state. Control reads in State 0 return the contents of the last register pointed to. Therefore, a register can be read continuously without writing to the Pointer. While the CIO is in State 1 (next control access is to the register pointed to), many internal operations are suspensed - no IPs are set and internal status is frozen. Therefore, to minimize interrupt latency and to allow continuous status updates, the CIO should not be left in State 1.

The CIO is reset by forcing $\overline{RD}$ and $\overline{WR}$ Low simultaneously (normally an illegal condition) or by writting a 1 to the Reset bit. Reset disables all functions except a read from or write to the Reset bit ; writes to all other bits are ignored, and all reads return 01_H. In this state, all control bits are forced to 0 and may be programmed only after clearing the Reset bit (by writing a 0 to it).

Table 3 : Register Selection.

A_1	A_0	Register
0	0	Port C's Data Register
0	1	Port B's Data Register
1	0	Port A's Data Register
1	1	Control Registers

Figure 11 : State Machine Operation.



Note : State changes occur only when A0 = A1 = 1. No other accesses have effect.

REGISTERS

Figure 12 : Master Control Registers.

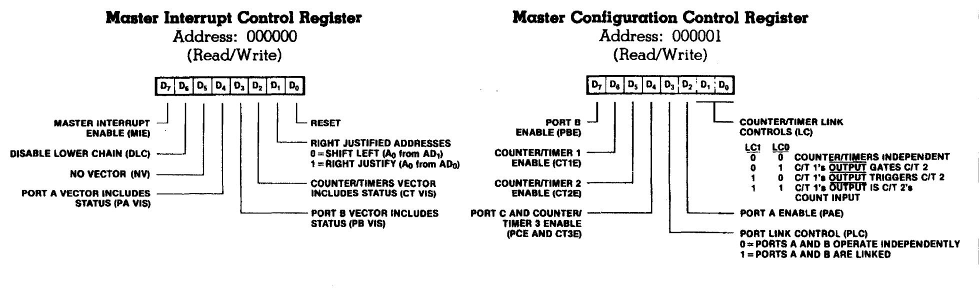
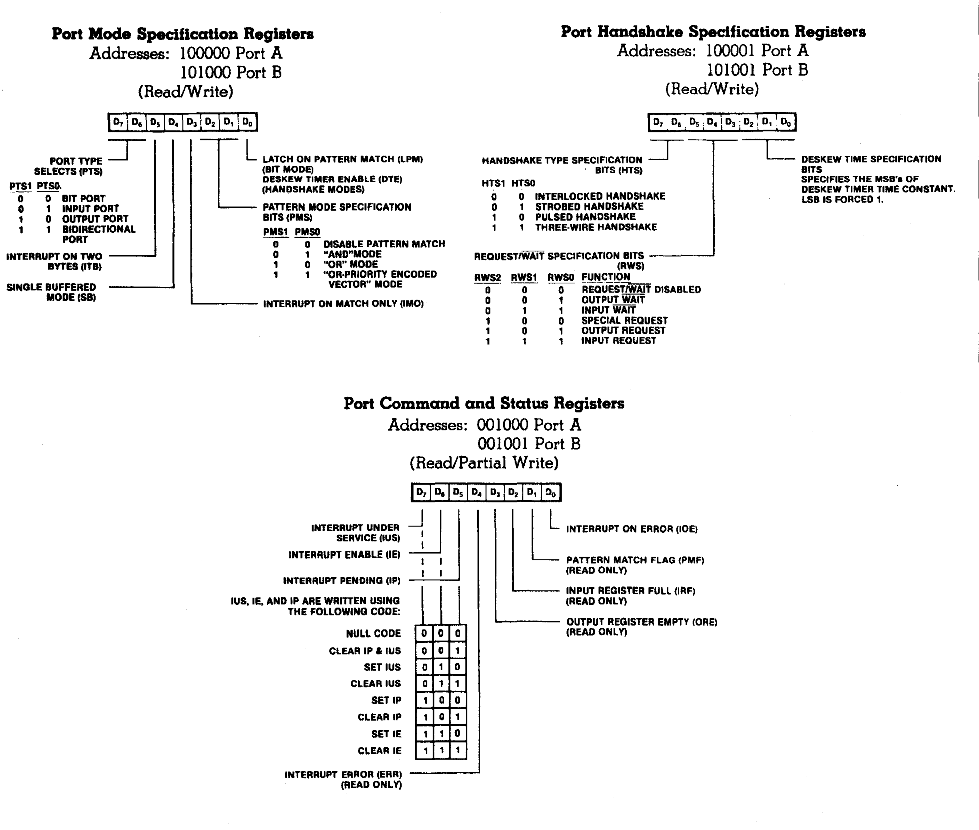

Master Interrupt Control Register
Address: 000000
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
MASTER INTERRUPT ENABLE (MIE)
DISABLE LOWER CHAIN (DLC)
NO VECTOR (NV)
PORT A VECTOR INCLUDES STATUS (PA VIS)
RESET
RIGHT JUSTIFIED ADDRESSES
0 = SHIFT LEFT (A0 from AD1)
1 = RIGHT JUSTIFY (A0 from AD0)
COUNTER/TIMERS VECTOR INCLUDES STATUS (CT VIS)
PORT B VECTOR INCLUDES STATUS (PB VIS)
Master Configuration Control Register
Address: 000001
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
PORT B ENABLE (PBE)
COUNTER/TIMER 1 ENABLE (CT1E)
COUNTER/TIMER 2 ENABLE (CT2E)
PORT C AND COUNTER/TIMER 3 ENABLE (PCE AND CT3E)
COUNTER/TIMER LINK CONTROLS (LC)
LC1 LC0
0 0 COUNTER/TIMERS INDEPENDENT
0 1 C/T 1's OUTPUT GATES C/T 2
1 0 C/T 1's OUTPUT TRIGGERS C/T 2
1 1 C/T 1's OUTPUT IS C/T 2's COUNT INPUT
PORT A ENABLE (PAE)
PORT LINK CONTROL (PLC)
0 = PORTS A AND B OPERATE INDEPENDENTLY
1 = PORTS A AND B ARE LINKED

Figure 13 : Port Specifications Registers.

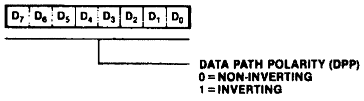

Port Mode Specification Registers
Addresses: 100000 Port A
101000 Port B
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
PORT TYPE SELECTS (PTS)
PTS1 PTS0
0 0 BIT PORT
0 1 INPUT PORT
1 0 OUTPUT PORT
1 1 BIDIRECTIONAL PORT
INTERRUPT ON TWO BYTES (ITB)
SINGLE BUFFERED MODE (SB)
LATCH ON PATTERN MATCH (LPM) (BIT MODE)
DESKEW TIMER ENABLE (DTE) (HANDSHAKE MODES)
PATTERN MODE SPECIFICATION BITS (PMS)
PMS1 PMS0
0 0 DISABLE PATTERN MATCH
0 1 "AND" MODE
1 0 "OR" MODE
1 1 "OR-PRIORITY ENCODED VECTOR" MODE
INTERRUPT ON MATCH ONLY (IMO)
Port Handshake Specification Registers
Addresses: 100001 Port A
101001 Port B
(Read/Write)
D7 D6 D5 D4 D3 D2 D1 D0
HANDSHAKE TYPE SPECIFICATION BITS (HTS)
HTS1 HTS0
0 0 INTERLOCKED HANDSHAKE
0 1 STROBED HANDSHAKE
1 0 PULSED HANDSHAKE
1 1 THREE-WIRE HANDSHAKE
REQUEST/WAIT SPECIFICATION BITS (RWS)
RWS2 RWS1 RWS0 FUNCTION
0 0 0 REQUEST/WAIT DISABLED
0 0 1 OUTPUT WAIT
0 1 1 INPUT WAIT
1 0 0 SPECIAL REQUEST
1 0 1 OUTPUT REQUEST
1 1 1 INPUT REQUEST
DESKEW TIME SPECIFICATION BITS
SPECIFIES THE MSB's OF DESKEW TIMER TIME CONSTANT.
LSB IS FORCED 1.
Port Command and Status Registers
Addresses: 001000 Port A
001001 Port B
(Read/Partial Write)
D7 D6 D5 D4 D3 D2 D1 D0
INTERRUPT UNDER SERVICE (IUS)
INTERRUPT ENABLE (IE)
INTERRUPT PENDING (IP)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING CODE:
NULL CODE 0 0 0
CLEAR IP & IUS 0 0 1
SET IUS 0 1 0
CLEAR IUS 0 1 1
SET IP 1 0 0
CLEAR IP 1 0 1
SET IE 1 1 0
CLEAR IE 1 1 1
INTERRUPT ERROR (ERR) (READ ONLY)
INTERRUPT ON ERROR (IOE)
PATTERN MATCH FLAG (PMF) (READ ONLY)
INPUT REGISTER FULL (IRF) (READ ONLY)
OUTPUT REGISTER EMPTY (ORE) (READ ONLY)

SGS-THOMSON MICROELECTRONICS

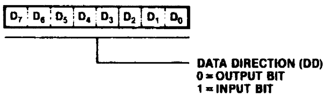
REGISTERS (cont'd)

Figure 14 : Bit Path Definition Registers.

Data Path Polarity Registers
Addresses: 100010 Port A
101010 Port B
000101 Port C (4 LSBs only)
(Read/Write)



Data Direction Registers
Addresses: 100011 Port A
101011 Port B
000110 Port C (4 LSBs only)
(Read/Write)



Special I/O Control Registers
Addresses: 100100 Port A
101100 Port B
000111 Port C (4 LSBs only)
(Read/Write)

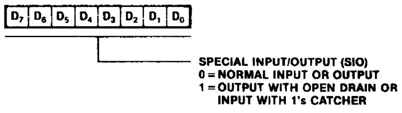
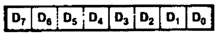

D7 D6 D5 D4 D3 D2 D1 D0
SPECIAL INPUT/OUTPUT (SIO)
0 = NORMAL INPUT OR OUTPUT
1 = OUTPUT WITH OPEN DRAIN OR
INPUT WITH 1's CATCHER

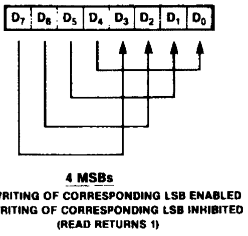
Figure 15 : Port Data Registers.

Port Data Registers
Addresses: 001101 Port A*
001110 Port B*
(Read/Write)



*These registers can be addressed directly.

Port C Data Register
Address: 001111*
(Read/Write)


D7 D6 D5 D4 D3 D2 D1 D0
4 MSBs
0 = WRITING OF CORRESPONDING LSB ENABLED
1 = WRITING OF CORRESPONDING LSB INHIBITED
(READ RETURNS 1)

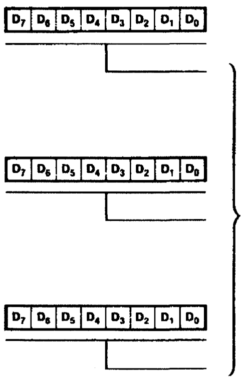
REGISTERS (cont'd)

Figure 16 : Pattern Definition Registers.

Pattern Polarity Registers (PP)
Addresses: 100101 Port A
101101 Port B
(Read/Write)

Pattern Transition Registers (PT)
Addresses: 100110 Port A
101110 Port B
(Read/Write)

Pattern Mask Registers (PM)
Addresses: 100111 Port A
101111 Port B
(Read/Write)

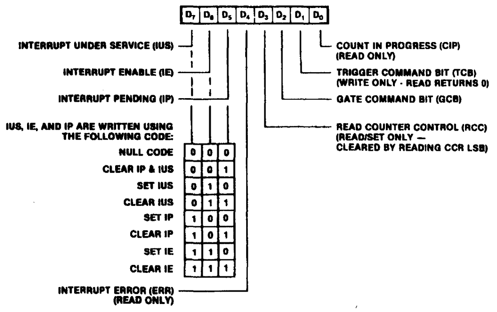


PM	PT	PP	PATTERN SPECIFICATION
0	0	X	BIT MASKED OFF
0	1	X	ANY TRANSITION
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE TO ZERO TRANSITION (↘)
1	1	1	ZERO-TO-ONE TRANSITION (↗)

Figure 17 : Counter/Timer Registers.

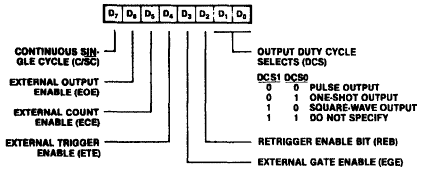
Counter/Timer Command and Status Registers

Addresses: 001010 Counter/Timer 1
001011 Counter/Timer 2
001100 Counter/Timer 3
(Read/Partial Write)


D7 D6 D5 D4 D3 D2 D1 D0
INTERRUPT UNDER SERVICE (IUS)
INTERRUPT ENABLE (IE)
INTERRUPT PENDING (IP)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING CODE:
NULL CODE 0 0 0
CLEAR IP & IUS 0 0 1
SET IUS 0 1 0
CLEAR IUS 0 1 1
SET IP 1 0 0
CLEAR IP 1 0 1
SET IE 1 1 0
CLEAR IE 1 1 1
INTERRUPT ERROR (ERR) (READ ONLY)
COUNT IN PROGRESS (CIP) (READ ONLY)
TRIGGER COMMAND BIT (TCB) (WRITE ONLY - READ RETURNS 0)
GATE COMMAND BIT (GCB)
READ COUNTER CONTROL (RCC) (READ/SET ONLY — CLEARED BY READING CCR LSB)

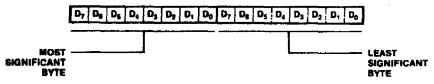
Counter/Timer Mode Specification Registers

Addresses: 011100 Counter/Timer 1
011101 Counter/Timer 2
011110 Counter/Timer 3
(Read/Write)


D7 D6 D5 D4 D3 D2 D1 D0
CONTINUOUS SINGLE CYCLE (C/SC)
EXTERNAL OUTPUT ENABLE (EOE)
EXTERNAL COUNT ENABLE (ECE)
EXTERNAL TRIGGER ENABLE (ETE)
OUTPUT DUTY CYCLE SELECTS (DCS)
DCS1 DCS0
0 0 PULSE OUTPUT
0 1 ONE-SHOT OUTPUT
1 0 SQUARE-WAVE OUTPUT
1 1 DO NOT SPECIFY
RETRIGGER ENABLE BIT (REB)
EXTERNAL GATE ENABLE (EGE)

Counter/Timer Current Count Registers

Addresses: 010000 Counter/Timer 1's MSB
010001 Counter/Timer 1's LSB
010010 Counter/Timer 2's MSB
010011 Counter/Timer 2's LSB
010100 Counter/Timer 3's MSB
010101 Counter/Timer 3's LSB
(Read Only)



Counter/Timer Time Constant Registers

Addresses: 010110 Counter/Timer 1's MSB
010111 Counter/Timer 1's LSB
011000 Counter/Timer 2's MSB
011001 Counter/Timer 2's LSB
011010 Counter/Timer 3's MSB
011011 Counter/Timer 3's LSB
(Read/Write)

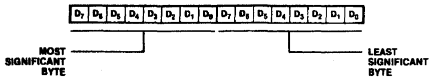
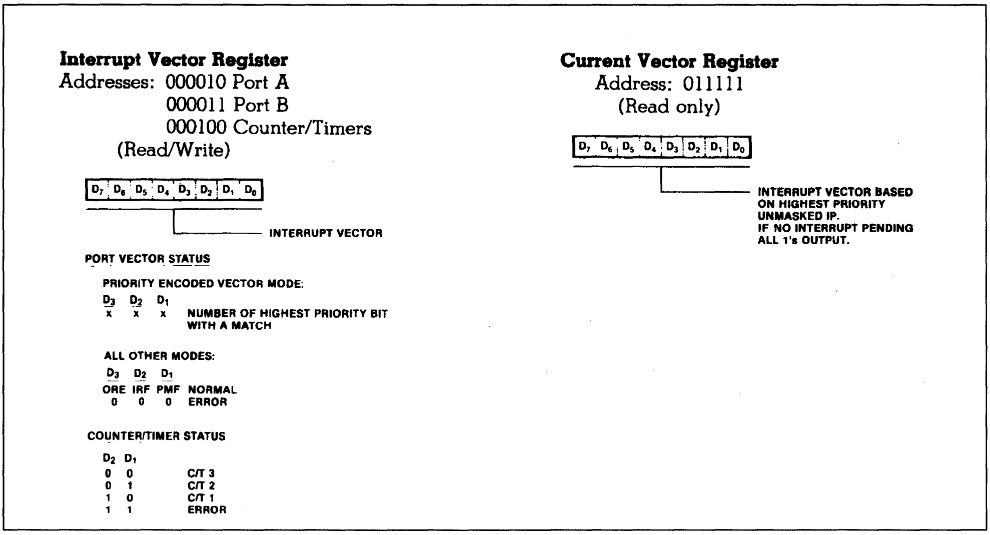


Figure 18 : Interrupt Vector Registers.


D7 D6 D5 D4 D3 D2 D1 D0
INTERRUPT VECTOR
D7 D6 D5 D4 D3 D2 D1 D0
INTERRUPT VECTOR BASED ON HIGHEST PRIORITY UNMASKED IP. IF NO INTERRUPT PENDING ALL 1's OUTPUT.

Interrupt Vector Register
Addresses: 000010 Port A
000011 Port B
000100 Counter/Timers
(Read/Write)

Current Vector Register
Address: 011111
(Read only)

PORT VECTOR STATUS

PRIORITY ENCODED VECTOR MODE:

D3	D2	D1	
x	x	x	NUMBER OF HIGHEST PRIORITY BIT WITH A MATCH

ALL OTHER MODES:

D3	D2	D1	
ORE	IRF	PMF	NORMAL
0	0	0	ERROR

COUNTER/TIMER STATUS

D2	D1	
0	0	C/T 3
0	1	C/T 2
1	0	C/T 1
1	1	ERROR

REGISTER ADDRESS SUMMARY

Main Control Registers

Address	Register Name
000000	Master Interrupt Control
000001	Master Configuration Control
000010	Port A's Interrupt Vector
000011	Port B's Interrupt Vector
000100	Counter/Timer's Interrupt Vector
000101	Port C's Data Path Polarity
000110	Port C's Data Direction
000111	Port C's Special I/O Control

Most Often Accessed Registers

Address	Register Name
001000	Port A's Command and Status
001001	Port B's Command and Status
001010	Counter/Timer 1's Command and Status
001011	Counter/Timer 2's Command and Status
001100	Counter/Timer 3's Command and Status
001101	Port A's Data (can be accessed directly)
001110	Port B's Data (can be accessed directly)
001111	Port C's Data (can be accessed directly)

Counter/Timer Related Registers

Address	Register Name
010000	Counter/Timer 1's Current Count-MSBs
010001	Counter/Timer 1's Current Count-LSBs
010010	Counter/Timer 2's Current Count-MSBs
010011	Counter/Timer 2's Current Count-LSBs
010100	Counter/Timer 3's Current Count-MSBs
010101	Counter/Timer 3's Current Count-LSBs
010110	Counter/Timer 1's Time Constant-MSBs
010111	Counter/Timer 1's Time Constant-LSBs
011000	Counter/Timer 2's Time Constant-MSBs
011001	Counter/Timer 2's Time Constant-LSBs
011010	Counter/Timer 3's Time Constant-MSBs
011011	Counter/Timer 3's Time Constant-LSBs
011100	Counter/Timer 1's Mode Specification
011101	Counter/Timer 2's Mode Specification
011110	Counter/Timer 3's Mode Specification
011111	Current Vector

Port A Specification Registers

Address	Register Name
100000	Port A's Mode Specification
100001	Port A's Handshake Specification
100010	Port A's Data Path Polarity
100011	Port A's Data Direction
100100	Port A's Special I/O Control
100101	Port A's Pattern Polarity
100110	Port A's Pattern Transition
100111	Port A's Pattern Mask

Port B Specification Registers

Address	Register Name
101000	Port B's Mode Specification
101001	Port B's Handshake Specification
101010	Port B's Data Path Polarity
101011	Port B's Data Direction
101100	Port B's Special I/O Control
101101	Port B's Pattern Polarity
101110	Port B's Pattern Transition
101111	Port B's Pattern Mask

TIMING

Read cycle. At the beginning of a read cycle, the CPU places an address on the address bus. Bits A_0 and A_1 specify a CIO register ; the remaining address bits and status information are combined and decoded to generate a Chip Enable (CE) signal that selects the CIO. When Read (RD) goes Low, data from the specified register is gated onto the data bus.

Write cycle. At the beginning of a write cycle, the CPU places an address on the data bus. Bits A_0 and A_1 specify a CIO register ; the remaining address bis and status information are combined and decoded to generate a Chip Enable (CE) signal that selects the CIO. When WR goes Low, data placed on the bus by the CPU is strobed into the specified CIO register.

Interrupt acknowledge. The CIO pulls its Interrupt Request (INT) line Low, requesting interrupt service from the CPU, if an Interrupt Pending (IP) bit is set and interrupts are enabled. The CPU responds with an Interrupt Acknowledge cycle. When Interrupt Acknowledge (INTACK) goes true and the IP is set, the CIO forces Interrupt Enable Out (IEO) Low, disabling all lower priority devices in the interrupt daisy chain. If the CIO is the highest priority device requesting service (IEI is High), it places its interrupt vector on the data bus and sets the Interrupt Under Service (IUS) bit when Read (RD) goes Low.

Figure 19 : Read Cycle Timing.

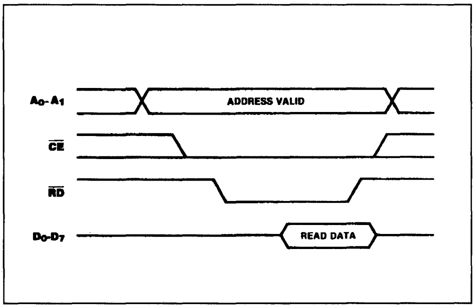


Figure 20 : Write Cycle Timing.

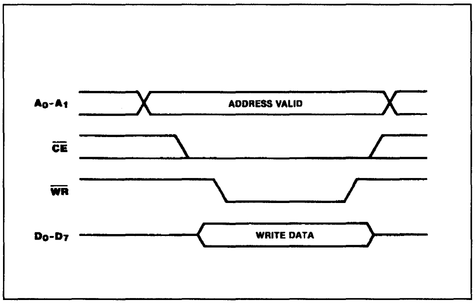
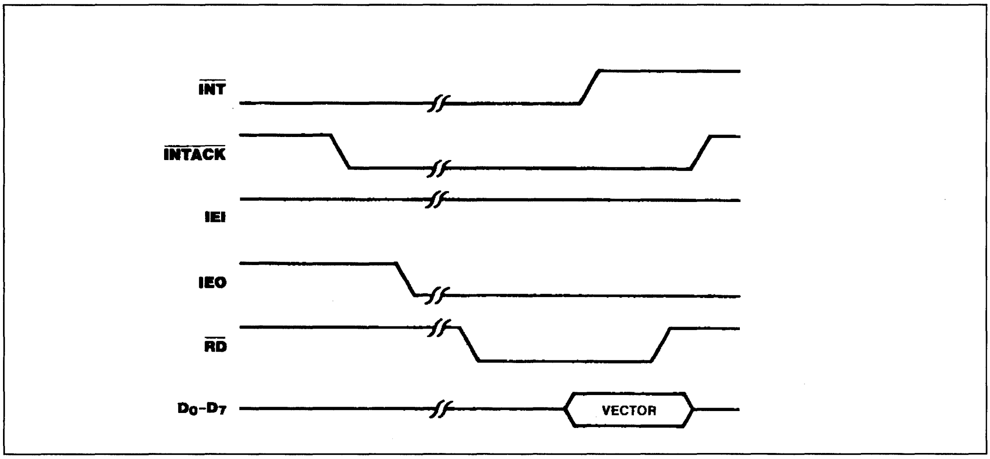


Figure 21 : Interrupt Acknowledge Timing.



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_I	Voltages on all Pins with Respect to GND	– 0.3 to + 7	V
T_A	Operating Ambient Temperature	0 to + 70 – 40 to + 85 – 55 to + 125	°C
T_{stg}	Storage Temperature	– 65 to + 150	°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only ; operation of the device at any codition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The DC characteristics and capacitance sections below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are as follows :

- $+ 4.75V \le V_{CC} \le + 5.25V$
- GND = 0V
- T_A as specified in Ordering Information

The Ordering Information section lists temperature ranges and product numbers. Package drawings are in the Package Information section in this book. Refer to the Literature List for additional documentation.

All ac parameters assume a load capacitance of 50pf max.

Figure 22 : Standard Test Load.

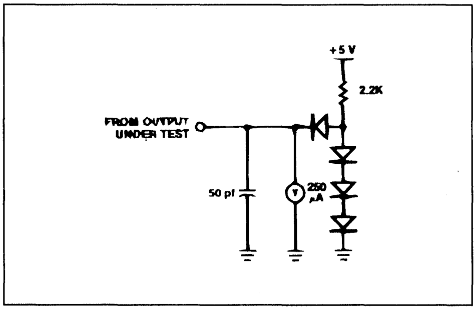
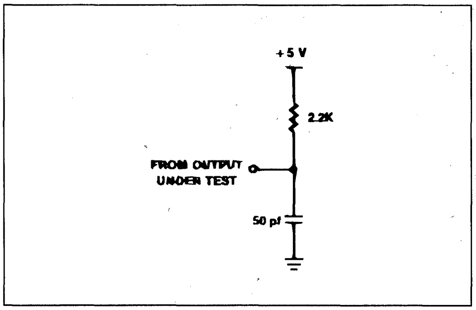


Figure 23 : Open-drain Test Load.



DC CHARACTERISTICS

Symbol	Parameter	Test Conditions	Min.	Max.	Unit
V_{IH}	Input High Voltage		2	V_{CC} + 0.3	V
V_{IL}	Input Low Voltage		– 0.3	0.8	V
V_{OH}	Output High Voltage	$I_{OH} = – 250\mu A$	2.4		V
V_{OL}	Output Low Voltage	$I_{OL} = + 2mA$		0.4	V
		$I_{OL} = + 3.2mA$		0.5	V
I_{IL}	Input Leakage	$0.4 \le V_{IN} \le + 2.4V$		± 10	µA
I_{OL}	Output Leakage	$0.4 \le V_{OUT} \le + 2.4V$		± 10	µA
I_{CC}	V_{CC} Supply Current			200	mA

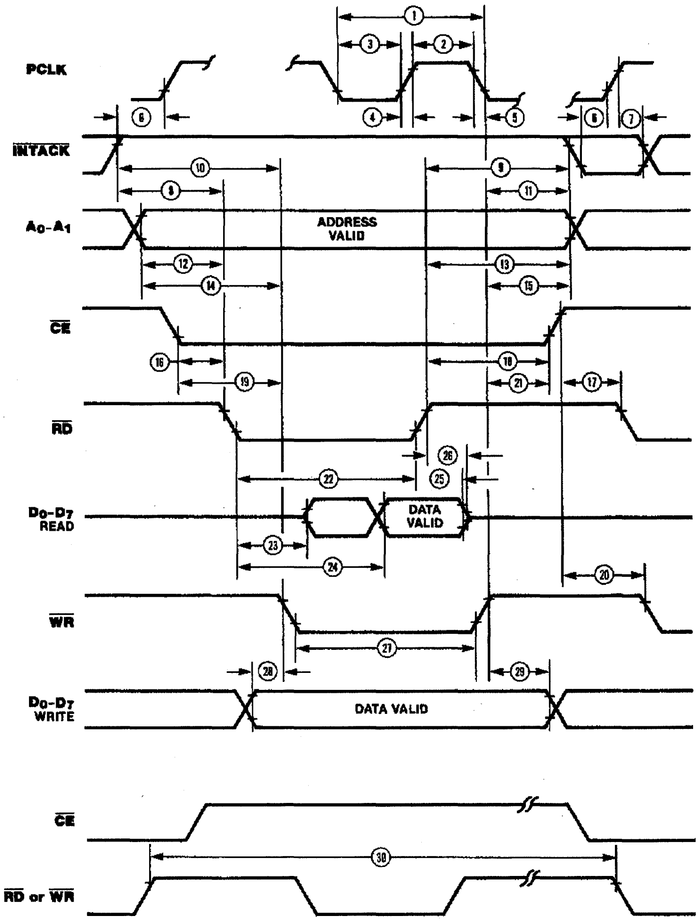
VCC = 5V ± 5% unless otherwise specified, over specified temperature range.

CAPACITANCE

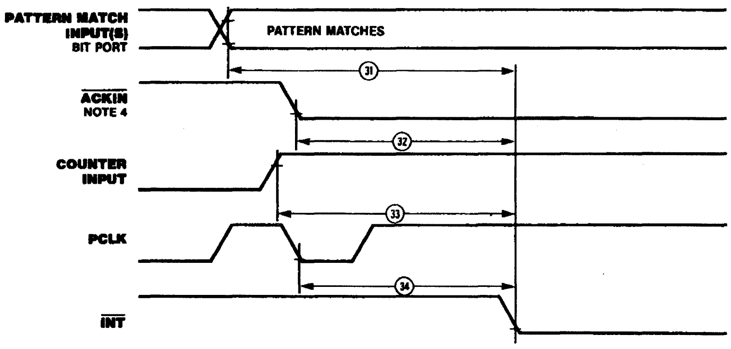
Symbol	Parameter	Test Conditions	Min.	Max.	Unit
C_{IN}	Input Capacitance			10	pf
C_{OUT}	Output Capacitance			15	pf
$C_{I/O}$	Bidirectional Capacitance			20	pf

f = 1MHz, over specified temperature range.
Unmeasured pins returned to ground.

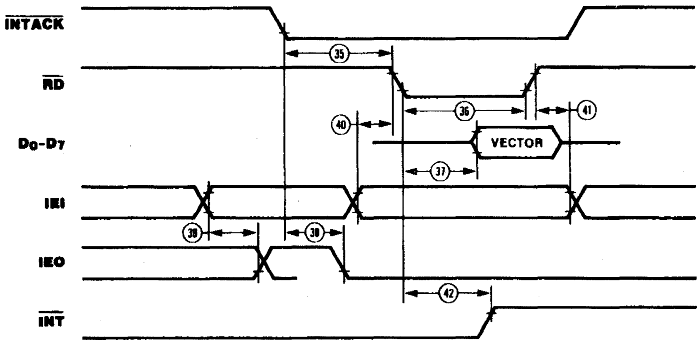
CPU INTERFACE TIMING



INTERRUPT TIMING



INTERRUPT ACKNOWLEDGE TIMING



SGS-THOMSON MICROELECTRONICS

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	TcPC	PCLK Cycle Time	250	4000	165	4000	
2	TwPCh	PCLK Width (high)	105	2000	70	2000	
3	TwPCl	PCLK Width (low)	105	2000	70	2000	
4	TrPC	PCLK Rise Time		20		10	
5	TfPC	PCLK Fall Time		20		15	
6	TsIA(PC)	$\overline{\text{INTACK}}$ to PCLK ↑ Setup Time	100		100		
7	ThIA(PC)	$\overline{\text{INTACK}}$ to PCLK ↑ Hold Time	0		0		
8	TsIA(RD)	$\overline{\text{INTACK}}$ to $\overline{\text{RD}}$ ↓ Setup Time	200		200		
9	ThIA(RD)	$\overline{\text{INTACK}}$ to $\overline{\text{RD}}$ ↑ Hold Time	0		0		
10	TsIA(WR)	$\overline{\text{INTACK}}$ to $\overline{\text{WR}}$ ↓ Setup Time	200		200		
11	ThIA(WR)	$\overline{\text{INTACK}}$ to $\overline{\text{WR}}$ ↑ Hold Time	0		0		
12	TsA(RD)	Address to $\overline{\text{RD}}$ ↓ Setup Time	80		80		
13	ThA(RD)	Address to $\overline{\text{RD}}$ ↑ Hold Time	0		0		
14	TsA(WR)	Address to $\overline{\text{WR}}$ ↓ Setup Time	80		80		
15	ThA(WR)	Address to $\overline{\text{WR}}$ ↑ Hold Time	0		0		
16	TsCEl(RD)	$\overline{\text{CE}}$ Low to $\overline{\text{RD}}$ ↓ Setup Time	0		0		1
17	TsCEh(RD)	$\overline{\text{CE}}$ High to $\overline{\text{RD}}$ ↓ Setup Time	100		70		1
18	ThCE(RD)	$\overline{\text{CE}}$ to $\overline{\text{RD}}$ ↑ Hold Time	0		0		1
19	TsCEl(WR)	$\overline{\text{CE}}$ Low to $\overline{\text{WR}}$ ↓ Setup Time	0		0		
20	TsCEh(WR	$\overline{\text{CE}}$ High to $\overline{\text{WR}}$ ↓ Setup Time	10		70		
21	ThCE(WR)	$\overline{\text{CE}}$ to $\overline{\text{WR}}$ ↑ Hold Time	0		0		
22	TwRDl	$\overline{\text{RD}}$ Low Width	390		250		1
23	TdRD(DRA)	$\overline{\text{RD}}$ ↓ to Read Data Active Delay	0		0		
24	TdRDf(DR)	$\overline{\text{RD}}$ ↓ To Read Data Valid Delay		255		180	
25	TdRDr(DR)	$\overline{\text{RD}}$ ↑ to Read Data not Valid Delay	0		0		
26	TdRD(DRz)	$\overline{\text{RD}}$ ↑ to Read Data Float Delay		70		45	2
27	TwWRl	$\overline{\text{WR}}$ Low Width	390		250		
28	TsDW(WR)	Write Data to $\overline{\text{WR}}$ ↓ Setup Time	0		0		
29	ThDW(WR)	Write Data to $\overline{\text{WR}}$ ↑ Hold Time	0		0		
30	Trc	Valid Accesss Recovery Time	1000 *		650		3
31	TdPM(INT)	Pattern Match to $\overline{\text{INT}}$ Delay (bit port)		2 + 800		2	6
32	TdACK(INT)	ACKIN to $\overline{\text{INT}}$ Delay (port with handshake)		10 + 600		10	4,6
33	TdCI(INT)	Counter Input to $\overline{\text{INT}}$ Delay (counter mode)		2 + 700		2	6
34	TdPC(INT)	PCLK to $\overline{\text{INT}}$ Delay (timer mode)		3 + 700		3	6
35	TsIA(RDA)	$\overline{\text{INTACK}}$ to $\overline{\text{RD}}$ $\overline{\text{WR}}$ (acknowledge) Setup Time	350		250		5
36	TwRDA	RD (acknowledge)Width	350		250		

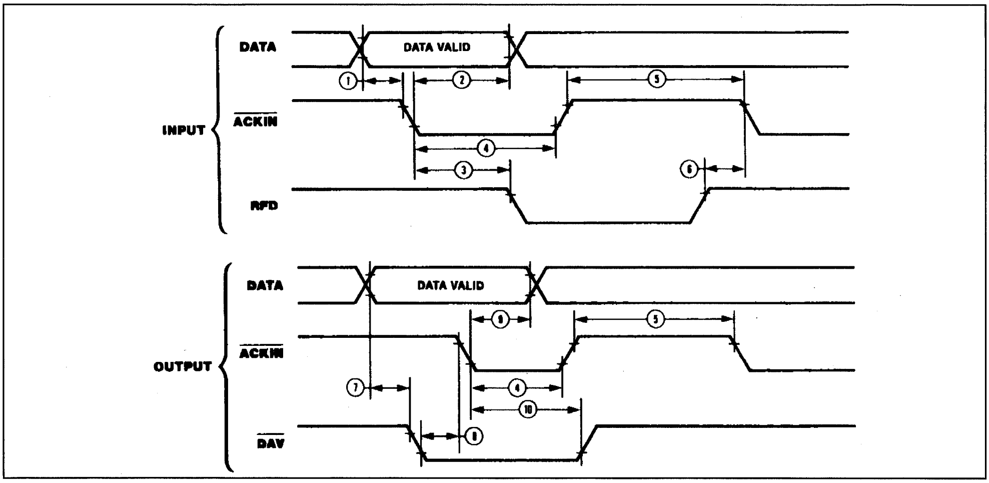
Notes :
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5V with minimum ac load and maximum dc load. 3.
3. Trc is the specified number or 3 TcPC, whichever is longer.
4. The delay is from DAV for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Ouput Handshake.
5. The parameters for the devices in any particular daisy chain must meet the following constraint : The delay from INTACK to RD must be greater than the sum of TdIA(IEO) for the highest priority peripheral, TsIEI(RDA) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.
6. Units are equal to TcPC plus ns.

* Timing are preliminary and subject to change. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0".

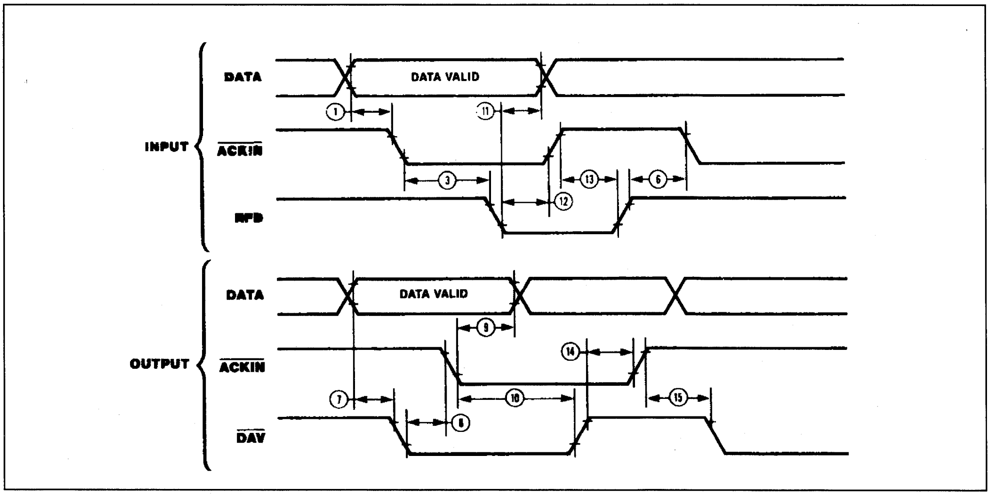
† Units in nanoseconds (ns), except as noted.

N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
37	TdRDA(IEO)	$\overline{RD}$ ↓ (acknowledge) to Read Data Valid Delay		250		180	
38	TdIA(IEO)	$\overline{INTACK}$ ↓ to IEO ↓ Delay		350		250	5
39	TdIEI(IEO)	IEI to IEO Delay		150		100	5
40	TsIEI(RDA)	IEI to RD ↓ (acknowledge) Setup Time	100		70		5
41	ThIEI(RDA)	IEI to $\overline{RD}$ ↑ (acknowledge) Hold Time	100		70		
42	TdRDA(INT)	$\overline{RD}$ ↓ (acknowledge) to $\overline{INT}$ ↑ Delay		600		600	

STROBED HANDSHAKE

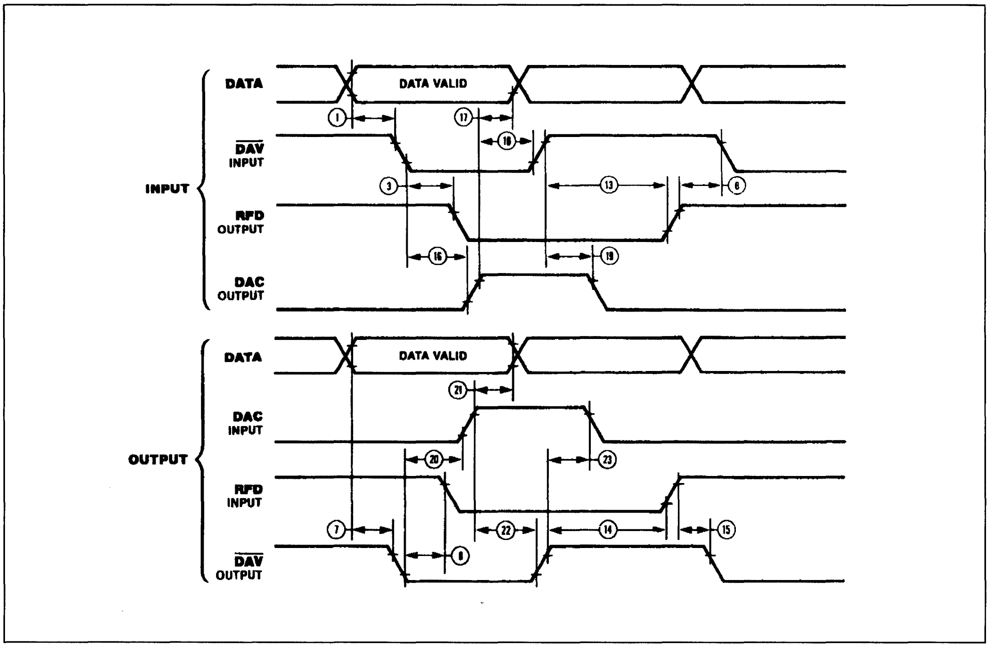


INTERLOCKED HANDSHAKE



SGS-THOMSON MICROELECTRONICS

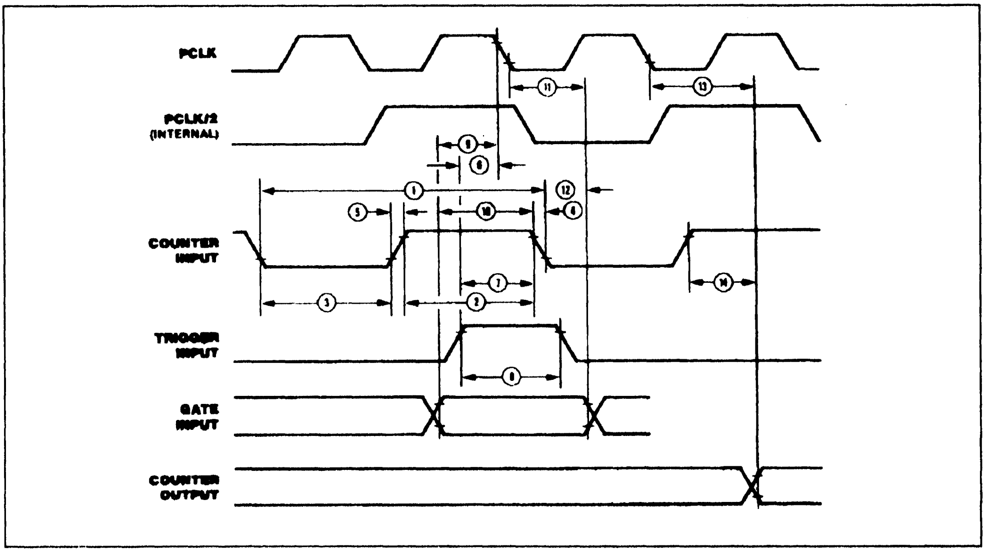
3-WIRE HANDSHAKE



N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	TsDI(ACK)	Data Input to $\overline{\text{ACKIN}}$ ↓ Setup Time	0		0		
2	ThDI(ACK)	Data Input to $\overline{\text{ACKIN}}$ ↓ Hold Time Strobed Handshake	500				
3	TdACKf(RFD)	$\overline{\text{ACKIN}}$ ↓ to RFD ↓ Delay	0		0		
4	TwACKl	$\overline{\text{ACKIN}}$ Low Width-Strobed Handshake	250				
5	TwACKh	$\overline{\text{ACKIN}}$ High Width-Strobed Handshake	250				
6	TdRFDr(ACK)	RFD ↑ to $\overline{\text{ACKIN}}$ ↓ Delay	0		0		
7	TsDO(DAV)	Data Out to $\overline{\text{DAV}}$ ↓ Setup Time	25		20		1
8	TdDAVf(ACK)	$\overline{\text{DAV}}$ ↓ to $\overline{\text{ACKIN}}$ ↓ Delay	0		0		
9	ThDO(ACK)	Data Out to $\overline{\text{ACKIN}}$ ↓ Hold Time	2		2		2
10	TdACK(DAV)	$\overline{\text{ACKIN}}$ ↓ to $\overline{\text{DAV}}$ ↑ Delay	2		2		2
11	THDI(RFD)	Data Input to RFD ↓ Hold Time-Interlocked Handshake					
12	TdRFDf(ACK)	RFD ↓ to $\overline{\text{ACKIN}}$ ↑ Delay Interlocked Handshake	0		0		
13	TdACKr(RFD)	$\overline{\text{ACKIN}}$ ↑ ($\overline{\text{DAV}}$ ↑) to RFD ↑ Delay-Interlocked and 3-Wire Handshake	0		0		
14	TdDAVr(ACK)	$\overline{\text{DAV}}$ ↑ to $\overline{\text{ACKIN}}$ ↑ (RFD ↑)-Interlocked and 3-Wire Handshake	0		0		
15	TdACK(DAV)	$\overline{\text{ACKIN}}$ ↑ (RFD ↑) to $\overline{\text{DAV}}$ ↓ Delay-Interlocked and 3-Wire Handshake	0		0		
16	TdDAVIf(DAC)	$\overline{\text{DAV}}$ ↑ to DAC ↑ Delay-Input 3-Wire Handshake	0		0		
17	ThDI (DAC)	Data Input to DAC ↑ Hold Time-3-Wire Handshake	0		0		
18	TdDACOr(DAV)	DAC ↑ to DAV ↑ Delay-Input 3-Wire Handshake	0		0		
19	TdDAVIr(DAC)	$\overline{\text{DAV}}$ ↑ to DAC ↓ Delay-Input 3-Wire Handshake	0		0		
20	TdDAVOf(DAC)	$\overline{\text{DAV}}$ ↑ to DAC ↓ Delay-Output 3-Wire Handshake	0		0		
21	ThDO(DAC)	Data Ouput to DAC ↑ Hold Time-3-Wire Handshake	2		2		2
22	TdDACIr(DAV)	DAC ↑ to $\overline{\text{DAV}}$ ↑ Delay-Output 3-Wire Handshake	2		2		2
23	TdDAVOr(DAC)	$\overline{\text{DAV}}$ ↑ to DAC ↓ Delay-Output 3-Wire Handshake	0		0		

Notes : 1. This time can be extended through the use of deskew timers.
2. Units equal to TcPC.
* Timing are preliminary and subject to change. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0".
† Units in nanoseconds (ns), except as noted.

COUNTER/TIMER TIMING

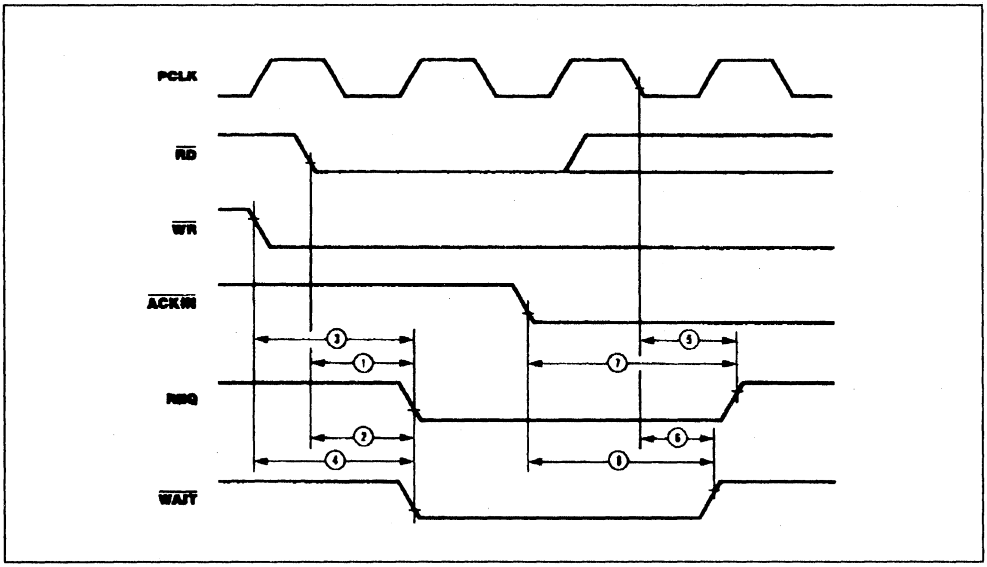


N°	Symbol	Parameter	4 MHz Min.	4 MHz Max.	6 MHz Min.	6 MHz Max.	Notes* †
1	TcCI	Counter Input Cycle Time	500		330		
2	TCIh	Counter Input High Width	230		150		
3	TWCI1	Counter Input Low Width	230		150		
4	TfCI	Counter Input Fall Time		20		15	
5	TrCI	Counter Input Rise Time		20		15	
6	TsTI(PC)	Trigger Input to PCLK ↓ Setup Time (timer mode)	150				1
7	TsTI(CI)	Trigger Input to Counter Input ↓ Setup Time (counter mode)	150				1
8	TwTI	Trigger Input Pulse Width (high or low)	200				
9	TsGI(PC)	Gate Input to PCLK ↓ Setup Time (timer mode)	100				1
10	TsGI(CI)	Gate Input to Counter Input ↓ Setup Time (counter mode)	100				1
11	ThGI(PC)	Gate Input to PCLK ↓ Hold Time (timer mode)	100				1
12	ThGI(CI)	Gate Input to Counter Input ↓ Hold Time (counter mode)	100				1
13	TdPC(CO)	PCLK to Counter Output Delay (timer mode)		475			
14	TdCI(CO)	Counter Input to Counter Output Delay (counter mode)		475			

Notes : 1. These parameters must be met to guarantee trigger or gate are valid for the next counter/timer cycle.
* Timing are preliminary and subject to change. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0".
† Units in nanoseconds (ns)

SGS-THOMSON MICROELECTRONICS

REQUEST/$\overline{\text{WAIT}}$ TIMING



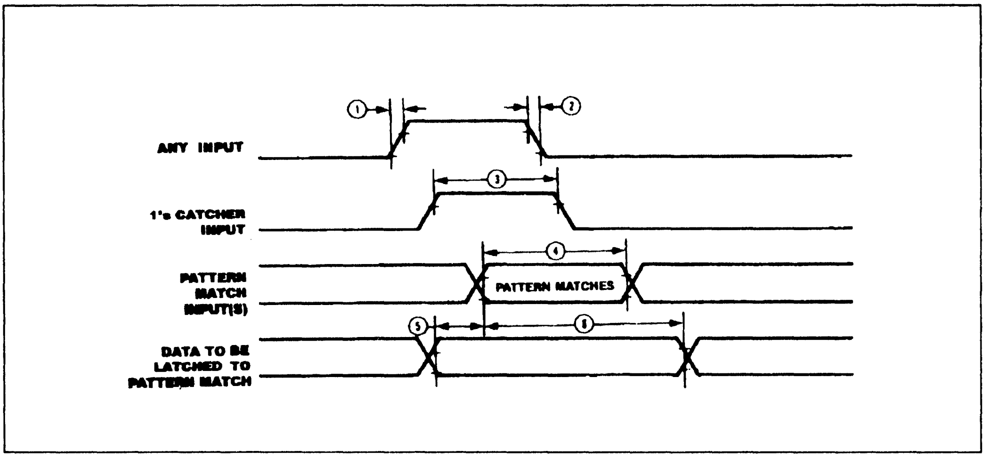
N°	Symbol	Parameter	4 MHz		6 MHz		Notes* †
			Min.	Max.	Min.	Max.	
1	TdRD(REQ)	$\overline{\text{RD}}$ ↓ to REQ ↓ Delay		500			
2	TdRD(WAIT)	$\overline{\text{RD}}$ ↓ to $\overline{\text{WAIT}}$ ↓ Delay		500			
3	TdWR(REQ)	$\overline{\text{WR}}$ ↓ to REQ ↓ Delay		500			
4	TdWR(WAIT)	$\overline{\text{WR}}$ ↓ to $\overline{\text{WAIT}}$ ↓ Delay		500			
5	TdPC(REQ)	PCLK ↓ to REQ ↑ Delay		300			
6	TdPC(WAIT)	PCLK ↓ to $\overline{\text{WAIT}}$ ↑ Delay		300			
7	TdACK(REQ)	$\overline{\text{ACKIN}}$ ↓ to REQ ↑ Delay		8 + 1000			1.2
8	TdACK(WAIT)	$\overline{\text{ACKIN}}$ ↓ to $\overline{\text{WAIT}}$ ↑ Delay		10 + 600			1.2

Notes : 1. The delays is from DAV ↑ for 3-wire input handshake. The delay is from DAC ↑ for 3-wire output handshake.
2. Units equal to TcPC + ns.
* Timing are preliminary and subject to change. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0".
† Units in nanoseconds (ns), except as noted.

N°	Symbol	Parameter	4 MHZ		6 MHZ		Notes* †
			Min.	Max.	Min.	Max.	
1	TdRD(WR)	Delay From $\overline{\text{RD}}$ ↑ to $\overline{\text{WR}}$ ↓ for No Rest	50		50		
2	TdWR(RD)	Delay From $\overline{\text{WR}}$ ↑ to $\overline{\text{RD}}$ ↓ for No Reset	50		50		
3	TwRES	Minimun Width of $\overline{\text{RD}}$ and $\overline{\text{WR}}$ both Low For Reset	250		250		

Notes : * Timing are preliminary and subject to change. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0".
† Units in nanoseconds (ns).

MISCELLANEOUS PORT TIMING



N°	Symbol	Parameter	4 MHz		6 MHz		Notes*†
			Min.	Max.	Min.	Max.	
1	Trl	Any Input Rise Time		100		100	
2	Tfl	Any Input Fall Time		100		100	
3	Tw1's	1's Catcher High Width	250		170		1
4	TwPM	Pattern Match Input Valid (Bit Port)	750		500		
5	TsPMD	Data latched on pattern match setup time (bit port)	0		0		
6	ThPMD	Data latched on pattern match hold time (bit port)	1000		65		

Notes : 1. If the input is programmed inverting, e low-going pulse of the same which will be detected.
* Timing are preliminary and subject to change. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0".
† Units in nanoseconds (ns).

ORDERING INFORMATION

Sales Type	Frequency	Temp. Range	Package
Z8536B1V	4MHz	0 to 70°C	DIL40-PLA
Z8536AB1V	6MHz	0 to 70°C	DIL40-PLA
Z8536B6V	4MHz	– 40 to 85°C	DIL40-PLA
Z8536AB6V	6MHz	– 40 to 85°C	DIL40-PLA
Z8536C1V	4MHz	0 to 70°C	PLCC44
Z8536AC1V	6MHz	0 to 70°C	PLCC44
Z8536C6V	4MHz	– 40 to 85°C	PLCC44
Z8536AC6V	6MHz	– 40 to 85°C	PLCC44
Z8536D1N	4MHz	0 to 70°C	DIL40-CER
Z8536AD1N	6MHz	0 to 70°C	DIL40-CER
Z8536D6N	4MHz	– 40 to 85°C	DIL40-CER
Z8536AD6N	6MHz	– 40 to 85°C	DIL40-CER
Z8536D2N	4MHz	– 55 to 125°C	DIL40-CER
Z8536AD2N	6MHz	– 55 to 125°C	DIL40-CER

APPLICATION NOTE

SGS-THOMSON
MICROELECTRONICS

APPLICATION NOTE

INTERFACING Z8500 UNIVERSAL PERIPHERALS TO THE TS68000

INTRODUCTION

This Application Note discusses interfacing the SGS-THOMSON Z8500 family of peripherals to the TS68000 microprocessor. The Z8500 peripheral family includes the Z8536 Counter/Timer and Parallel I/O Unit (CIO), The Z8038 FIFO Input/Output Interface Unit (FIO), and the Z8530 Serial Communications Controller (SCC). This document discusses the Z8500/68000 interfaces and presents hardware examples, and verification techniques. One of the three hardware examples given in this paper shows how to implement the Z8500/68000 interface using a single-chip programmable logic array (PAL).

This Application Note about interfacing supplements the following documents, which discuss the individual components of the interface.

- Z8038 FIO/FIFO Datasheet
- Z8530 SCC Datasheet
- Z8531 A-SCC Datasheet
- Z8536 CIO Datasheet
- TS68000 Datasheet

This Application Note is divided into four sections. The first section gives a general description of the Z8500 family and discusses pin functions, interrupt structures, and the programming of operating modes. The second section discusses the Z8500 interface itself. It shows how the different Z8500 control signals are generated from the 68000 signals and summarizes the critical timings for the three types of bus cycles. The third section shows three examples of implementing the TS68000 to Z8500 peripheral interface. The fourth section suggests methods of verifying the interface design by checking the three different types of bus cycle : Read, Write, and Interrupt Acknowledge.

GENERAL Z8500 FAMILY DESCRIPTION

The Z8500 family is made up of programmable peripherals that can interface easily to the bus of any nonmultiplexed CPU microprocessor, such as the TS68000. The three members of this family, the CIO, SCC, and FIO, can solve many design problems. The peripherals' operating modes can be programmed simply by writing to their internal registers.

PROGRAMMING THE OPERATING MODES

The CPU can access two types of register : Control and Data. Depending on the peripheral, registers are selected with either the A_0, A_1, $A/\overline{B}$, or $D/\overline{C}$ function pins.

Peripheral operating modes are initialized by programming internal registers. Since these registers are not directly addressable by the CPU, a two-step procedure using the Control register is required : first, the address of the internal register is written to the Control register, then the data is written to the Control register. A state machine determines whether an address or data is being written to the Control register. Reading an internal register follows a similar two-step procedure : first, the address is written then the data is read.

The Data registers that are most frequently accessed, for example, the SCC's transmit and receive buffer, can be addressed directly by the CPU with a single read or write operation. This reduces overhead in data transfers between the peripheral and CPU.

GENERATING Z8500 CONTROL SIGNALS

This section shows how to generate the Z8500 control signals. To simplify the discussion, the section is divided into two parts. The first part takes each individual Z8500 signal and shows how it is generated from the TS68000 signals. The second part discusses the Z8500 timing that must be met when generating the control signals.

Z8500 SIGNAL GENERATION

The right-hard side of table 1 lists the Z8500 signals that must be generated. Each of these signals is discussed in a separate paragraph.

A_0,A_1, $A/\overline{B}$, $D/\overline{C}$. These pins are used to select the peripheral's Control and Data registers that program the different operating modes. They can be connected to the TS68000 A_1 and A_2 Address bus lines.

$\overline{CE}$. Each peripheral has an active Low Chip Enable that can be derived by ANDing the selected address decode and the 68000's Address Strobe ($\overline{AS}$). The active Low $\overline{AS}$ guarantees that the TS68000 addresses are valid.

D_0-D_7. The Z8500 Data bus can be directly connected to the lowest byte (D_0-D_7) of the 68000 Data bus.

IEI, IEO. The peripherals use these pins to decide the interrupt priority. The highest priority device should have its IEI tied High. Its IEO should be connected to the IEI pin of the next highest priority device. This pattern continues with the next highest priority peripheral, until the peripherals are all connected, as shown in Figure 1.

$\overline{INT}$. The interrupt request pins for each peripheral in the daisy chain can be wire-ORed and connected to the 68000's $\overline{ILP}_n$ pins. The 68000 has seven interrupt levels that can be encoded into the $\overline{ILP}_0$, $\overline{ILP}_1$, and $\overline{ILP}_2$ pins. Multiple 68000 interrupt levels can be implemented by using a multiplexer like the 74LS148.

$\overline{INTACK}$. The $\overline{INTACK}$ pin signals the peripheral that an Interrupt Acknowledge cycle is occurring. The following equation describes how $\overline{INTACK}$ is generated :

$$\overline{INTACK} = \overline{(FC_0) . (FC_1) . (FC_2) . (\overline{AS})}$$

The 68000 FC_0-FC_2 are status pins that indicate an Interrupt Acknowledge when they are all High. They should be ANDed with inverted $\overline{AS}$ to guarantee their validity. The $\overline{INTACK}$ signals must be synchronized with PCLK to guarantee set-up and hold times. This can be accomplished by changing the state of $\overline{INTACK}$ on the falling edge of PCLK. If the $\overline{INTACK}$ pin is not used, it must be tied High.

PCLK. The SCC and CIO require a clock for internal synchronization. The clock can be generated by dividing down the 68000 CLK.

$\overline{RD}$. The Read strobe goes active Low under three conditions : hardware reset, normal Read cycle, and an Interrupt Acknowledge cycle. The following equation describes how RD is generated :

$$\overline{RD} = \overline{[(R/\overline{W}) . (AS) + RESET)]}$$

The Read strobe timing must meet both the Read timing and Interrupt Acknowledge timing discussed in the following section. In addition to enabling the Data bus drivers, the falling edge of $\overline{RD}$ sets the Interrupt Under Service (IUS) bits during an Interrupt Acknowledge cycle.

$\overline{WR}$. This signal strobes data into the peripheral. A data to-write setup time requires that data be valid before $\overline{WR}$ goes active Low. The $\overline{WR}$ equation for generating the $\overline{WR}$ strobe is made up of two components : an active reset and a normal Write cycle, as shown in the allowing equation :

$$\overline{WR} = \overline{[(\overline{R/\overline{W}}) . (AS) + RESET]}$$

Forcing $\overline{RD}$ and $\overline{WR}$ simultaneously Low resets the peripherals.

Table 1 : Z8500 and TS68000 Pin Functions.

TS68000 Signals		Z8500 Signals	
Mnemonic	**Function**	**Mnemonic**	**Function**
A_1-A_{23}	Address Bus	A_0, A_1, A/$\overline{B}$, D/$\overline{C}$*	Register Select
$\overline{AS}$	Address Strobe	$\overline{CE}$	Chip Enable
CLK	68000 Clock (8MHz)	D_0-D_7	Data Bus
D_0-D_{15}	Data Bus	IEI, IEO	Interrupt Daisy Chain Control
$\overline{DTACK}$	Data Transfer Acknowledge	$\overline{INT}$	Interrupt Request
FC_0-FC_2	Processor Status	$\overline{INTACK}$	Interrupt Acknowledge
$\overline{ILP_0}$-$\overline{ILP_2}$	Interrupt Request	PCLK	Peripheral Clock
R/$\overline{W}$	Read/write	$\overline{RD}$	Read Strobe
$\overline{VMA}$	Valid Memory Address	$\overline{WR}$	Write Strobe
$\overline{VPA}$	Valid Peripheral Address		

Note : * The register select pins on each peripheral have different names.

Z8500 TIMING CYCLES

This section discusses the timing parameters that must be met when generating the control signals. The Z8500 family uses the control signals to communicate with the CPU via three types of bus cycle : Read, Write, and Interrupt Acknowledge. The discussion that follows pertains to the 4MHz peripherals, but the 6MHz devices have similar timing considerations.

Although the peripherals have a standard CPU interface, some of their particular timing requirements vary. The worst-case parameters are shown below ; the timing can be optimized if only one or two of the Z8500 family devices are used.

READ CYCLE

The Read cycle transfers data from the peripheral to the CPU. It begins by selecting the peripheral and appropriate register (Data or Control). The data is gated onto the bus with the RD line. A setup time of 80ns from the time the register select input (A/B, C/D, A_0, A_1) are stable to the falling edge of RD guarantees that the proper register is accessed. The access time specification is usually measured from the falling edge of RD to valid data and varies between peripherals. The SCC specifies an additional register select to valid data time. The Read cycle timing is shown in figure 2.

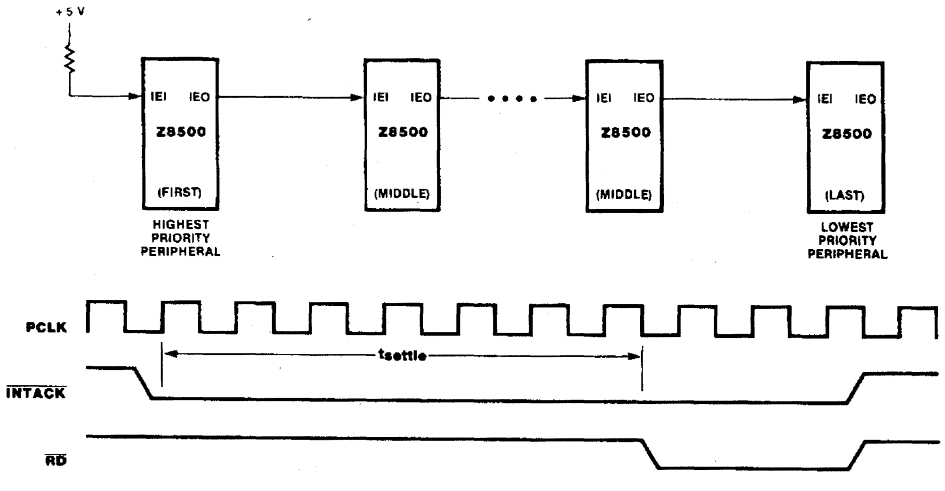
WRITE CYCLE

The Write cycle transfers data from the CPU to the peripheral. It begins by selecting the peripheral and addressing the desired register. A setup time of 80ns from register select stable to the falling edge of WR is required. The data must be valid prior to the falling edge of WR. The WR pulse width is specified at 400ns. Write cycle timing is shown in figure 2.

INTERRUPT ACKNOWLEDGE CYCLE

The Z8500 peripheral interrupt structure offers the designer many options. In the simplest case, the Z8500 peripherals can be polled with interrupts disabled. If using interrupts, the timing shown in figure 2 should be observed. (Detailed discussions of the interrupt processing can be found in the relevant datasheet).

Figure 1 : Peripheral Interrupt Daisy Chain.



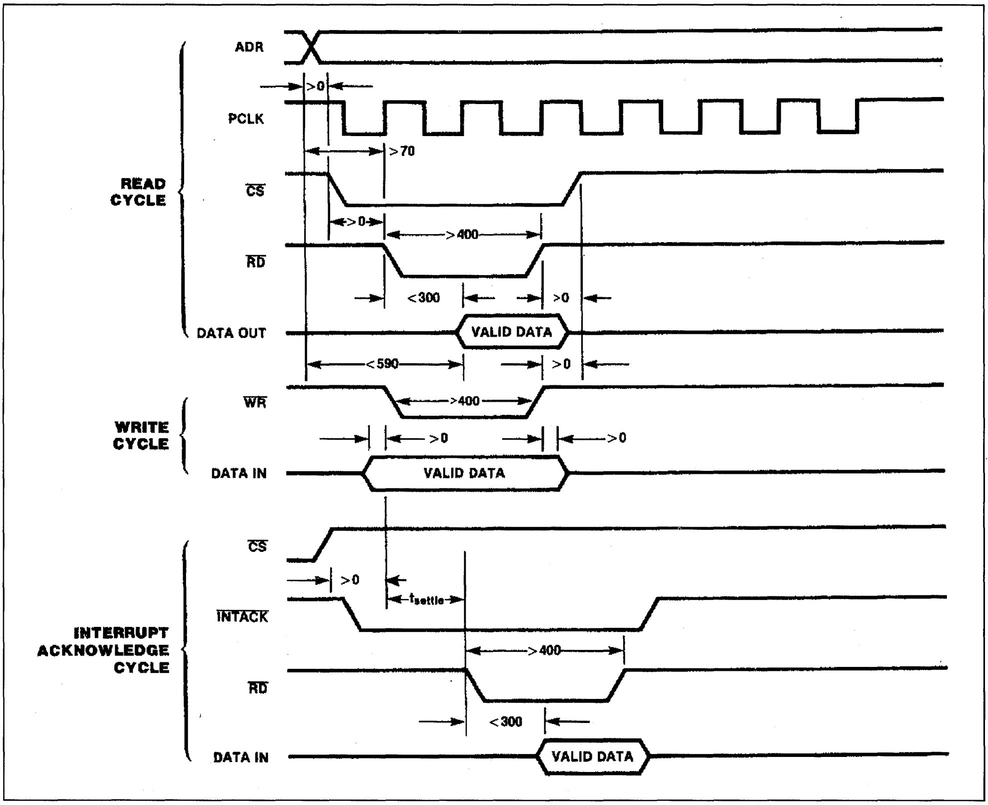
PERIPHERAL (4MHz)	t_{settle} (NS) FIRST	MIDDLE	LAST
CIO	350	150	100
FIO	350	150	100
SCC	250	120	120

An interrupt sequence begins with an $\overline{INT}$ going active because of an interrupt condition. The CPU acknowledges the interrupt with an INTACK signal.

A daisy-chain settle time (dependent upon the number of devices in the chain) ensures that the interrupts are prioritized. The falling edge of $\overline{RD}$ causes the IUS bit to be set and enables a vector to go out on the bus.

The table given figure 1 can be used to calculate the amount of settling time required by a daisy chain. Even if there is only one peripheral in the chain, a minimum settling time is still required because of the internal daisy chain. The first column specifies the amount of settling time for only one peripheral. If there are two peripherals, the time is computed by adding together the times shown in the first and the last columns. For each additional peripheral in the chain, the time specified in the middle column is added.

Figure 2 : Z8500 Interface Timing (4MHz).



RECOVERY TIME

The read/write recovery time specifies a minimum amount of time between Read or Write cycles to the same peripheral. The recovery time differs among peripherals and is summarized in figure 3. In most cases, this parameter is met because of the time required for instruction fetches. The recovery time specification does not have to be met if $\overline{CE}$ is deselected when Read or Write occurs.

68000 INTERFACE EXAMPLES

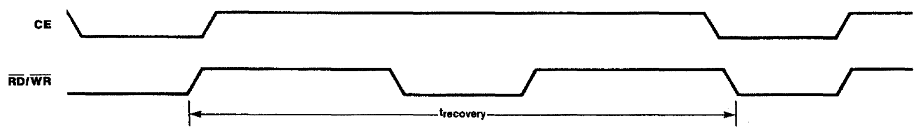
This section shows three examples, presented in increasing order of complexity, for interfacing the 4MHz Z8500 peripherals to an 8MHz TS68000. Faster CPUs or peripherals can be used by modifying some of the timing. These examples suggest possible ways of implementing the interface but may require some modifications to operate properly. They were chosen because they give the user a variety of interface design ideas. The first example uses a minimum amount of TTL logic to implement the interface because the Valid Peripheral Address (VPA) cycle meets the Z8500 timing requirements. In this mode the 68000 accepts only nonvectored interrupts. The second example uses the Data Transfer Acknowledge (DTACK) pin. This interface allows faster operation and makes use of the Z8500's 8-bit vectored interrupts. The third example also uses a DTACK cycle and is similar to the second, except the external logic is integrated into a single chip, the PAL20X10 programmable array logic.

EXAMPLE 1 : A TTL INTERFACE USING A VPA CYCLE

The 68000 has a special input pin, Valid Peripheral Address (VPA), that can be activated by the Z8500 chip select logic at the beginning of the cycle to indicate to the 68000 that a peripheral is being accessed. This generates a special Read/Write cycle that meets the peripheral timing requirements. This cycle allows the Z8500 control signals to be generated easily. The 68000 responds to interrupts using an autovector and the Z8500 can be programmed not to return a vector.

Figure 4 shows how the hardware can be implemented. PCLK is generated by dividing down the 68000 CLK. $\overline{RD}$, $\overline{WR}$, and $\overline{INTACK}$ are simply ANDed 68000 singnals. The worst-case daisy-chain settle time is 450ns. Connecting $\overline{INT}$ to $\overline{IPL}_0$ generates a level 1 interrupt. The internal registers are accessed by A_0, A_1, D/$\overline{C}$ and A/$\overline{B}$, which can be the 68000 lowest order addresses. The timing is shown in figure 5.

Figure 3 : Recovery Time.



Peripheral (4MHz)	Recovery Time
CIO	Greater than 3 PCLK cycles or 1000ns
FIO	Greater than 1000ns
SCC	Greater than 6 PCLK cycles + 200ns

Note : the diagram shows that the recovery time is measured between consecutive reads and writes only if the peripheral is selected.

Figure 4 : Interface Using the $\overline{VPA}$ Cycle.

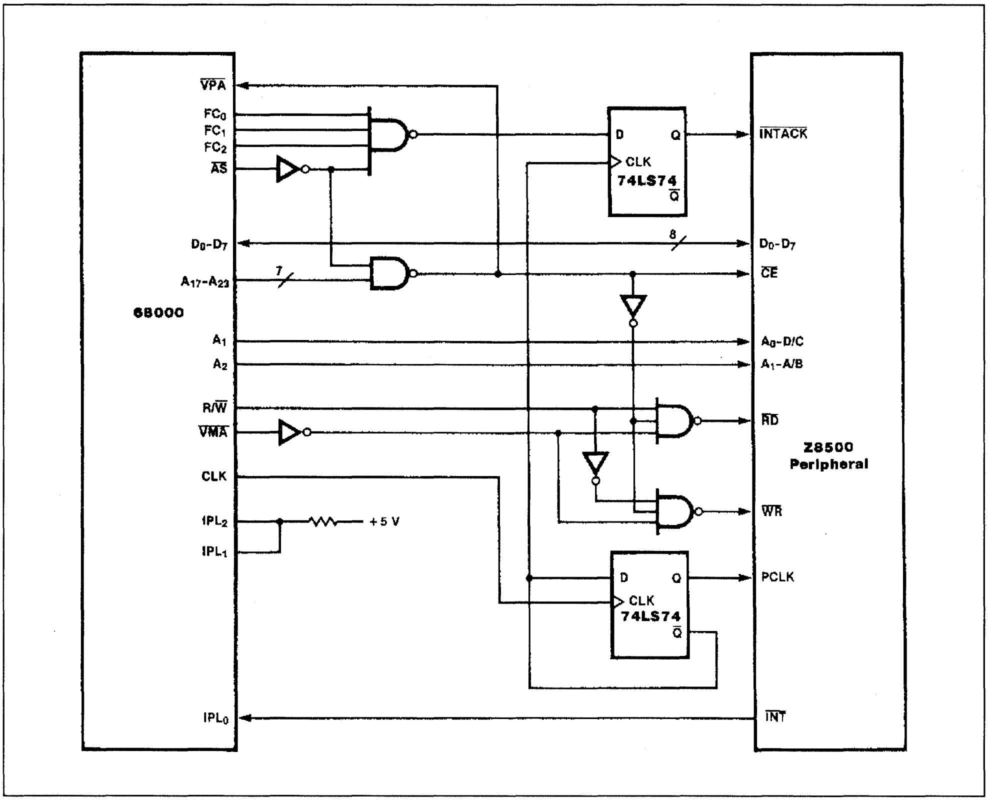
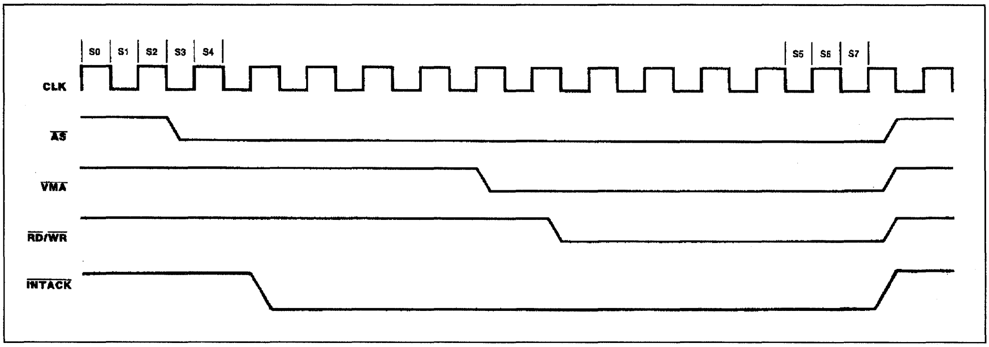


Figure 5 : $\overline{VPA}$ Cycle Timing.



FUNCTIONAL DESCRIPTION

$\overline{VPA}$ is pulled Low at the beginning of the cycle and the CPU automatically inserts Wait states until E is synchronized.

VPA = [(AS) . (CE)]

RD = [(CE) . (VMA) . (R/$\overline{W}$)]

WR = [(CE) . (VMA) . $\overline{(R/\overline{W})}$]

INTACK = [(FC0) . (FC1) . (FC2) . (AS)]

EXAMPLE 2 : A TTL INTERFACE USING DTACK CYCLES

Using at the 68000 Data Transfer Acknowledge (DTACK) cycle is a second way of interfacing to the Z8500 peripherals. The 68000 inserts Wait states until the $\overline{DTACK}$ input is strobed Low to complete the transfer. In addition to generating the control signals, the interface logic must also generate DTACK.

The timing shown in figure 6 can be generated by the hardware shown in figure 7. The 8-bit Shift register (74LS164) is used to generate the proper timing. At the beginning of each cycle, Q_A (figure 7) is set High for one PCLK cycle and then reset. This pulse is shifted through the Q_A-Q_H outputs and is used to generate $\overline{RD}$, $\overline{WR}$, and $\overline{DTACK}$ signals. Some of the extra Wait states can be eliminated by tapping the Shift register sooner (e.g., Q_C).

EXAMPLE 3 : SINGLE-CHIP PAL INTERFACE

This example illustrates how to interface the 4MHz Z8500 peripherals to the 8MHz 68000 using a PAL20X10 device to generate all the required control signals. The PAL reduces the required interface logic to a single chip, thus minimizing board space. This interface offers flexibility because the internal logic can be reprogrammed without changing the pin functions. The PAL uses 68000 signals to generate Read, Write, and Interrupt Acknowledge cycles. In addition to generating the Z8500 control signals, the PAL also generates a $\overline{DTACK}$ (Data Transfer Acknowledge) to inform the 68000 of a completed data transfer cycle. This allows the 68000 to use the peripheral's vectored interrupts.

Figure 6 : Timing for $\overline{DTACK}$ Interface.

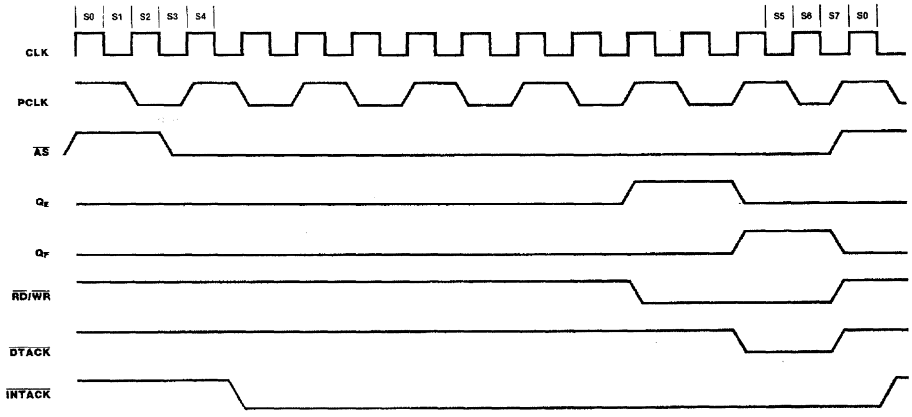
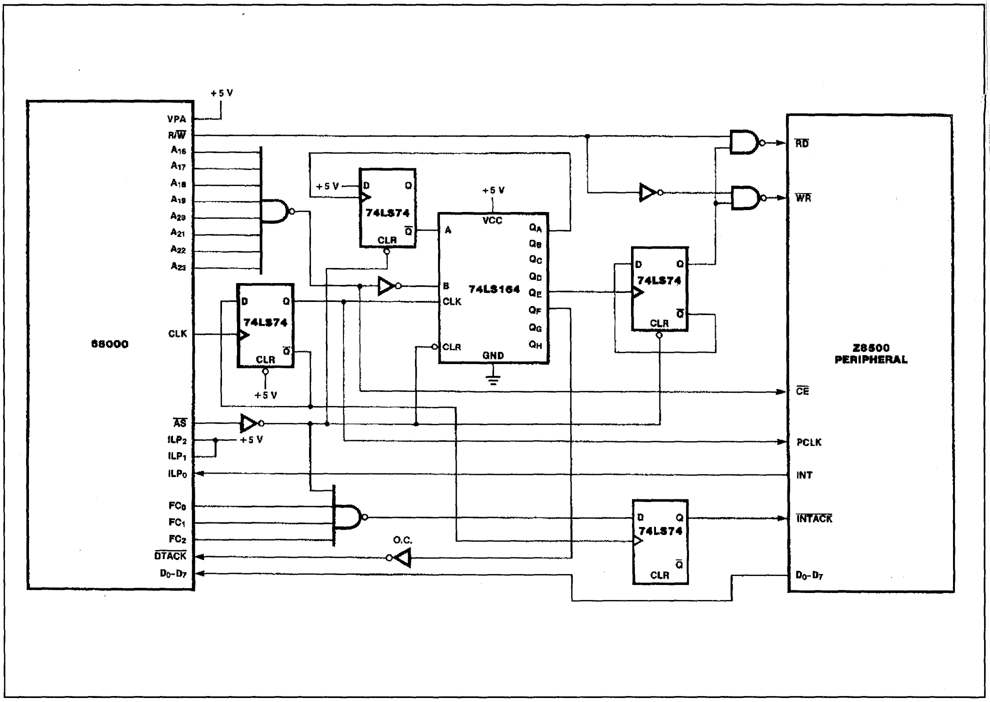


Figure 7 : Hardware Diagram for $\overline{\text{DTACK}}$ Interface.

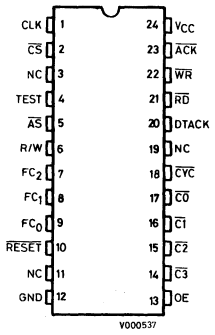


FUNCTIONAL DESCRIPTION

Figure 8 shows the PAL's pin functions. The PAL generates five control signals, of which four ($\overline{\text{WR}}$, $\overline{\text{RD}}$, $\overline{C_0}$, and $\overline{\text{INTACK}}$) go to the Z8500 and one ($\overline{\text{DTACK}}$) goes to the 68000. The remaining signals are used internally to generate these outputs. Timing diagrams for the Read, Write, and Interrupt Acknowledge cycles are shown in Figure 9.

The PAL uses a 4-bit downcounter to generate the proper placement of the control signals where C_0 is the least-significant bit and C_3 is the most-significant bit. All of the PAL is clocked with the rising edge of the 68000's CLK. The counter toggles between counts 14 and 15 and starts counting down when $\overline{\text{AS}}$ goes active. The counter goes back to toggling when $\overline{\text{AS}}$ goes inactive. CYC goes active Low at the same time the counter starts counting down. The equations in Table 2 can be entered into a development board to program the PAL.

Figure 8 : PAL Pinout.



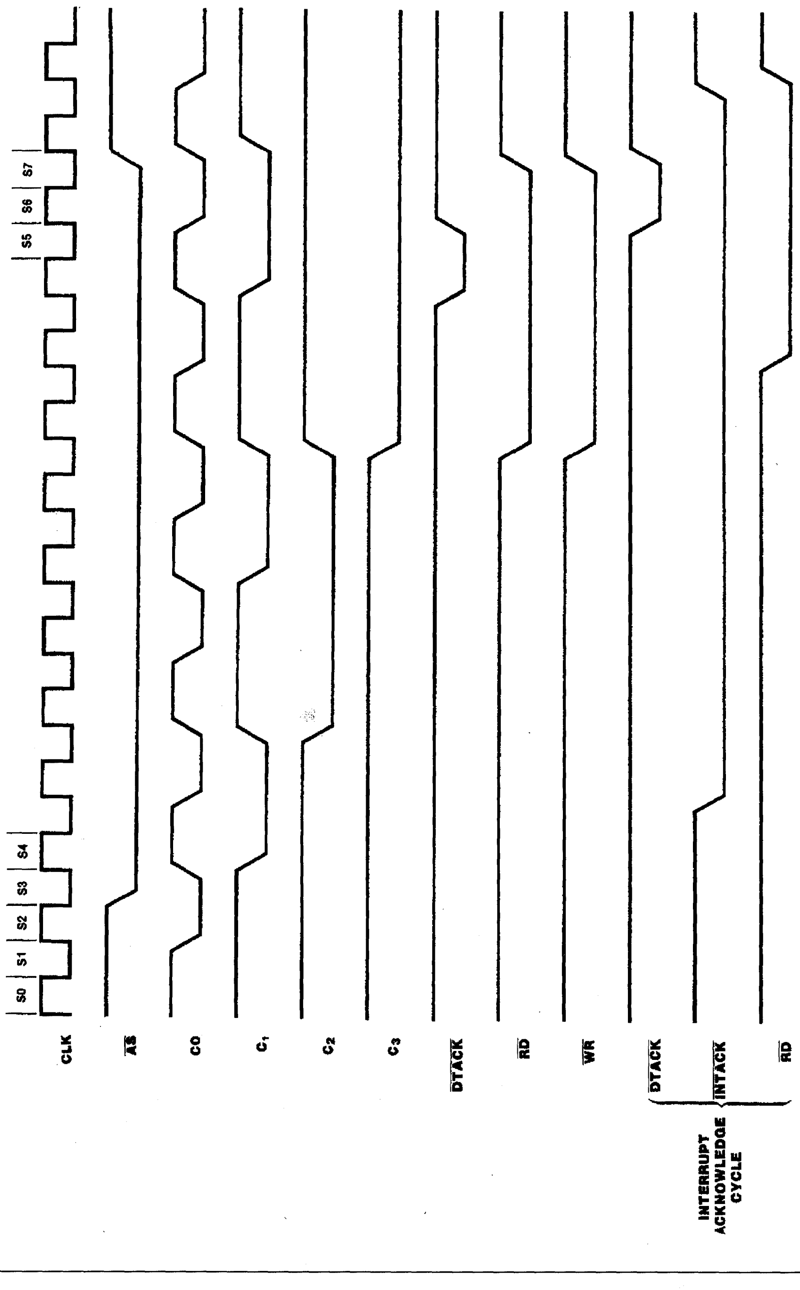
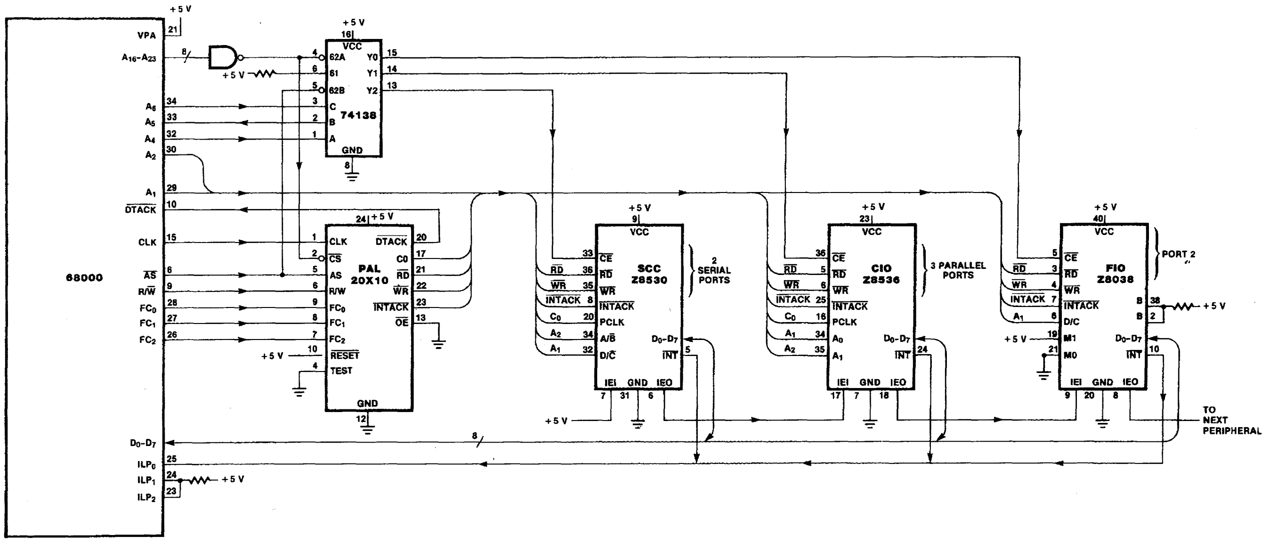


Figure 9 : PAL Interface Timing.

Table 2 : PAL Equations.

```
PAL20X10                                                    PAL DESIGN SPECIFICATION
P7089
MC68000 TO SGS-THOMSON PERIPHERAL INTERFACE
MMI, SUNNYVALE, CA
CLK / CS NC TEST / AS RW
FC2 FC1 FC0 / RESET NC GND
/ OE / C3 / C2 / C1 / C0 / CYC
NC / DTK / RD / WR / ACK VCC


CD     :=    /C0 */TEST                                     ; COUNT/HOLD (LSB)


C1     :=    /RESET*AS*C1                                   ; HOLD
       :+:   /RESET*AS*C0                                   ; DECREMENT


C2     :=    /RESET*AS*C2                                   ; HOLD
       :+:   /RESET*AS*C0*C1                                ; DECREMENT


C3     :=    /RESET*AS*C3                                   ; HOLD
       :+:   /RESET*AS*C0*C1*C2                             ; DECREMENT


DTK    :=    /RESET*/ACK*CYC*C3*/C2*/C1*C0*CS               ; DTACK FOR RD/WR CYCLE
        +    /RESET*ACK*CYC*C3*/C2*C1*/C0                   ; DTACK FOR INTERRUPT
                                                            ; OPERATION


CYC    :=    /RESET*AS*/CYC*C0                              ; NEW CYCLE STARTED
        +    /RESET*AS*CYC                                  ; PROCESSING OF CYCLE
       :+:   /RESET*CYC*DTK                                 ; END OF CYCLE


RD     :=    /RESET*CYC*/ACK*RW*C3*/C2*CS                   ; NORMAL READ OPERATION
        +    /RESET*CYC*/ACK*RW*/C3*C2*C1*C0*CS             ; NORMAL READ OPERATION
       :+:   /RESET*CYC*ACK*RW*C3                           ; READ DURING OPERATION


WR     :=    /RESET*CYC*/ACK*/RW*C3*/C2*CS                  ; WRITE
        +    /RESET*CYC*/ACK*/RW*/C3*C2*C1*C0*CS            ; WRITE
       :+:   RESET


ACK    :=    /RESET*FC0*FC1*FC2*AS*CYC*/CD                  ; INTERRUPT ACKNOWLEDGE
        +    /RESET*FC0*FC1*FC2*CYC                         ; INTERRUPT ACKNOWLEDGE
```

Figure 10 : PAL Hardware Diagram.



HARDWARE DIAGRAM

The hardware diagram of the PAL interface is shown in figure 10. The 68000 signals CLK, $\overline{CS}$, $\overline{AS}$, R/$\overline{W}$, FC_0, FC_1 and FC_2 are used to generate the Z8500 control signals. The control signals are synchronous with the rising edge of the 68000's CLK. TEST and $\overline{OE}$ must be grounded. $\overline{CS}$ is used to enable $\overline{DTACK}$, $\overline{RD}$, and $\overline{WR}$ as shown in the equations. The Z8500 $\overline{INT}$ is connected to $\overline{ILP}_0$, which generates a 68000 level 1 interrupt. The peripherals are memory-mapped into the highest 64K byte block of memory, where A_{17} - A_{23} equals "FF_H". Addresses A_4 - A_6 are used to select the peripheral ; A_1 - A_3 select the internal registers. Table 3 shows the peripheral's memory map.

Table 3 : Peripheral Memory Map.

Peripheral	Register	Hex Address
SCC (Z8530)	Channel B Control	FF0020
	Channel B Data	FF0022
	Channel A Control	FF0024
	Channel B Data	FF0026
CIO (Z8536)	Port C's Data Register	FF0010
	Port B's Data Register	FF0012
	Port A's Data Register	FF0014
	Control Register	FF0016
FIO (Z8038)	Data Register	FF0000
	Control Register	FF0002

INTERFACE VERIFICATION TECHNIQUES

This section suggests possible ways of verifying the Read, Write, and Interrupt Acknowledge cycles.

READ CYCLE VERIFICATION

The Read cycle should be checked first because it is the simplest operation. The Z8500 should be hardware reset by simultaneously pulling $\overline{RD}$ and $\overline{WR}$ Low. When the peripheral is in the reset state, the Control register containing the reset bit can be read without writing the pointer. Reading back the FIO and CIO Control register should yield a 01_H.

The SCC's Read cycle can be verified by reading the bits in RR0. Bits D_2 and D_6 are set to 1 and bits D_0, D_1, and D_7 are 0. Bits D_3 - D_5 reflect the input pins DCD, SYNC, and CTS, respectively.

WRITE CYCLE VERIFICATION

This Write cycle can be checked by writing to a register and reading back the results. Both the CIO and FIO must have their reset bits cleared by writing 00_H to their Control registers and reading back the result. The SCC can be checked by writing and reading to an arbitrary read/write register, for example, the Time Constant register (WR12 or WR13).

INTERRUPT ACKNOWLEDGE CYCLE VERIFICATION

Verifying an Interrupt Acknowledge ($\overline{INTACK}$) cycle consists of several steps. First, the peripheral makes an Interrupt Request ($\overline{INT}$) to the CPU. When the processor is ready to service the interrupt, it initiate an Interrupt Acknowledge (INTACK) cycle. This peripheral then puts an 8-bit vector on the bus, and the 68000 uses that vector to get to the correct service routine. This test checks the simplest case.

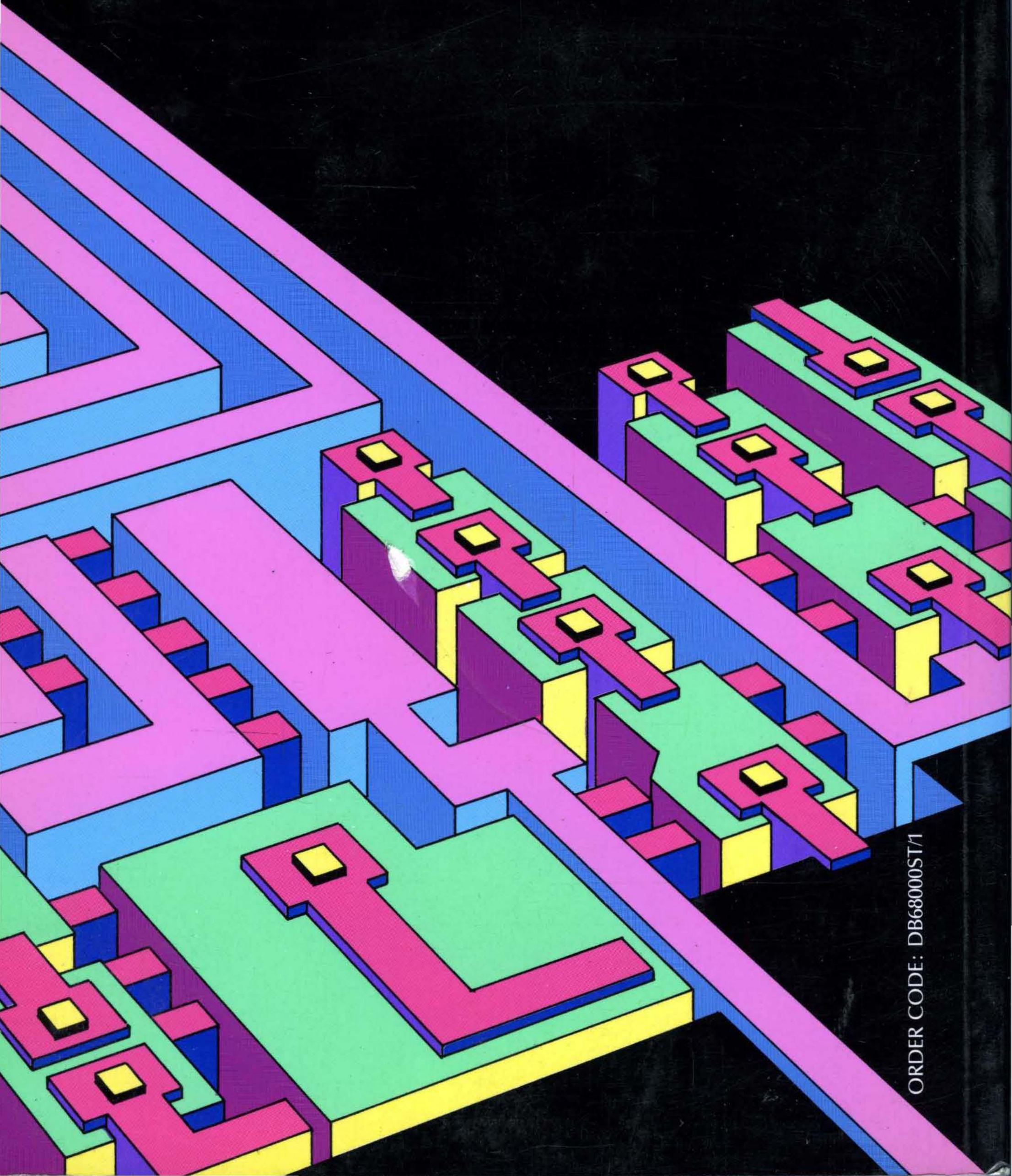
First, load the Interrupt Vector register with a vector, disable the Vector Includes status (VIS), and enable interrupts (IE = 1, MIE = 1, IEI = 1). Disabling VIS guarantees that only one vector is put on the bus. The address of the service routine corresponding to the 8-bit vector number must be loaded into the 68000's vector table.

Initiating an interrupt sequence in the FIO and CIO can be accomplished by setting one of the interrupt pending (IP) bits and seeing if the 68000 jumps to the service routine (setting a breakpoint at the beginning of the service routine is an easy way to check if this has happened).

Initiating an interrupt sequence in the SCC is not quite as simple because the IP bits are not as accessible to the user. An interrupt can be generated indirectly via the CTS pin by enabling the following : CTS IE (WR15 20), EXT INT EN (WR1 01), and MIE (WR9 08). Any transition on the CTS pin can initiate the interrupt sequence. The interrupt can be re-enabled by RESET EXT/STATUS INT (WR0 10) and RESET HIGHEST IUS (WR0 38).

CONCLUSION

SGS-THOMSON Z8500 family of nonmultiplexed Address/Data bus peripherals can interface easily with the SGS-THOMSON TS68000 and provide all the support required in a high-performance microprocessor system. The many features offered by the SCC, FIO and CIO solve many system design problems by making interfacing to the external world easy. These intelligent peripherals also greatly enhance the system performance by relieving the CPU of many burdensome overhead tasks. Additionally, the powerful interrupt structure allows the TS68000 to use vectors and reduce interrupt response time.



ORDER CODE: DB68000ST/1