TEXAS INSTRUMENTS

TMS320C14/TMS320E14

User's Guide

1988

Digital Signal Processor Products

TEXAS INSTRUMENTS

TMS320C14/TMS320E14

User's Guide

1988

Digital Signal Processor Products

TMS320C14/TMS320E14
1988

TMS320C14/TMS320E14 User's Guide

TEXAS INSTRUMENTS

Contents

Section *Page*

Illustrations

Figure *Page*

Tables

Section 1

Introduction

The TMS320C14 and TMS320E14 are members of Texas Intruments first-generation TMS320 digital signal processor (DSP) family. The TMS320C14/E14 has been specifically designed for control system applications and are the first devices that combine the high performance of a DSP with the on-chip peripherals of a microcontroller. The TMS320C14/E14, at 25.6 MHz, offer 10 to 20 times the speed of traditional 16-bit microcontrollers and microprocessors.

The DSP engine of the TMS320C14/E14 provides analog designers, for the first time, a digital solution without sacrificing the precision and performance of their systems. In fact, system performance can be enhanced through the use of advanced control algorithms. These include adaptive control, Kalman filtering, and state controllers. The TMS320C14/E14 offer the reliability and programmability of a digital solution. Analog control systems, on the other hand are hard-wired solutions, and can experience performance degradation due to aging and other environmental factors.

The high speed central processing unit (CPU) of the TMS320C14/E14 allows the digital designer to process algorithms in real time as opposed to approximation results via look-up tables. System performance is thus dramatically increased. The general purpose instruction set of the TMS320C14/E14 coupled with the extensive development support for the TMS320 DSP family reduces development time, and provides the same ease of use as traditional 8- and 16-bit microcontrollers.

The TMS320 family architecture has been available for more than six years providing users with the security of a standard architecture. The TMS320 family has now expanded into three generations of processors: TMS320C1x, TMS320C2x, and TMS320C3x (see Figure 1-1). Many features are common among these generations. Some specific features are added in each processor to provide different cost/performance tradeoffs. Software compatibility is maintained throughout the family to protect the user's investment in architecture. Each processor has software and hardware tools to facilitate rapid design.

This section includes the following information listed below:

- Control System Design Considerations (Section 1.1 page 1-3)
- TMS320C14/E14 Description (Section 1.2 on page 1-5)
- Key Features (Section 1.3 on page 1-6)

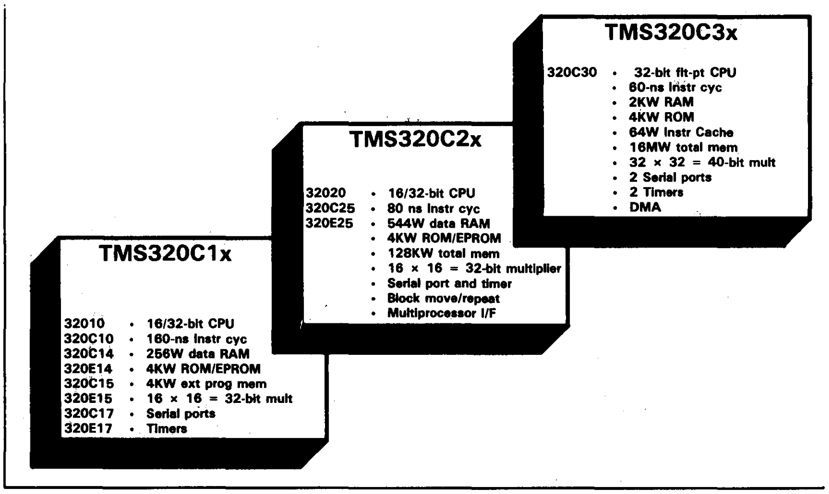


Figure 1-1. TMS320 Family Evolution

1.1 Control System Design Considerations

Traditional control systems have been implemented with analog components, and designers have had to learn to live with these systems. Analog components exhibit the disadvantages of temperature drift and component aging. An analog solution for an application requires that hardware be designed to perform a specific function. Modifications and upgrades for such hardware mean labor-intensive projects. These problems have prompted a growing trend toward implementing control systems in digital form.

In a digital control system, the actual processing of the intput signal is done in digital form by a microprocessor. However, the input signal has to be converted into digital form by an analog-to-digital (A/D) converter. The output of the microprocessor has to be converted again (D/A) to analog form to provide a control function.

Although 8/16-bit microcontrollers have the necessary peripherals that would seem to provide a single chip solution for a digital controller, they lack both the performance and the architecture needed to process control signals. For years control designers have worked around this problem by using look-up tables. In this method, the processing of control signals is done in non-realtime on large computers and the results are programmed in the microcontroller's memory. When an input signal is received, the corresponding result is looked up from memory and routed from the processor. The algorithms that can be used with this kind of approach are usually simpler and limited to single input, single output systems.

In some cases realtime processing of control signals is implemented with traditional microcontrollers/microprocessors. These controllers are restricted to low bandwidth systems. In spite of these limitations, most digital control systems implemented today use 8/16-bit general purpose microprocessors/microcontrollers. Designers of high precision and high bandwidth systems find solutions provided by these CPUs far from optimum. They still have to rely upon analog solutions to provide the performance needed for their systems.

To provide an optimum solution for a digital controller, a processor must have the architecture, performance, and peripherals necessary for digital control systems. The input signal is not processed continuously in digital control systems, but rather sampled at discrete intervals. The discrete samples are then processed by the controller. Selection of a sampling interval is critical and is usually chosen to be six to ten times the bandwidth of the system. For realtime performance the processor should be able to process the sample before the arrival of the next sample. Most digital control and DSP algorithms are made up of many multiply and accumulate terms and the processor should be able to perform these operations very rapidly. For this reason, the TMS320 family of processors incorporate a hardware multiplier to perform a multiply in only one clock cycle.

In addition to having realtime capability, the controller should have the appropriate architecture. Conversion of a continuous signal into discrete form results in loss of resolution. This is often referred to as quantization error. Therefore, the controller should have a large wordlength to reduce quantization error. Additional errors are also introduced due to processing of signals. As an example, if a 16 x 16 multiply is done, the result is 32 bits. However, if only 16 bits are stored, an error is induced because of lack of precision. This is referred to as truncation error. The TMS320 family minimizes the effects of truncation by employing 32-bit precision for storing intermediate results.

Additional truncation and quantization errors can be introduced if the numbers are not scaled properly (i.e., all significant bits are not maintained, and there are too many leading or trailing zeros). To maintain the correct precision, shifters should be available that do not require CPU overhead. Finally, numeric processing of control signals can create overflows, causing the accumulator to wrap-around and suddenly go from the most positive value to the most negative with disastrous consequences for the control system. In the TMS320 family, the accumulator can be prevented from wrapping around, and the value is kept at the most positive or most negative value. This simulates the saturation of an analog system.

The final requirement for an optimum controller is to provide all the necessary peripherals on a single chip. Different members of the TMS320 family have been optimized for specific signal processing applications. The TMS320C14/E14 integrates peripherals on-chip that have been optimized for implementing digital controllers.

1.2 TMS320C14/E14 Description

The TMS320C14/E14 is a DSP that meets the requirements for an optimum digital controller. Using a 25.6 MHz clock input, the TMS320C14/E14 can execute 6.4 million instructions per second. Almost all instructions are executed in a single cycle, including multiplication. This high performance allows execution of very complex control algorithms such as adaptive control and Kalman filters in realtime. Very high sampling rates can also be implemented to minimize loop delays.

The TMS320C14/E14 has been optimized for digital control system applications and has all the architectural features necessary for high-speed signal processing. The device possesses all the peripherals needed to provide a single-chip solution in control system applications. Based on the TMS320 family's first-generation CPU (the industry standard TMS320C10), the device includes 256 words of RAM and 4K words of ROM or EPROM. Also integrated on-chip are additional peripheral functions for control systems and other applications requiring a single-chip stand-alone DSP controller. These peripherals include bit-selectable I/O ports, a serial port with USART and codec-compatible modes. six high-precision pulse width modulation (PWM) outputs, four capture inputs, and four independent timers. The TMS320C14/E14 devices are manufactured using CMOS technology, achieving a power dissipation of less than one sixth that of a comparable NMOS device.

The instruction set of the TMS320C14/E14 is source and object code compatible with the other members of the TMS320C1x family, allowing users to protect their investment in TMS320C1x software.

The TMS320C14/E14 architecture is also optimized for processing control signals. A 16-bit wordlength is used along with 32-bit registers for storing intermediate results, and two hardware shifters are available to scale numbers independent of the CPU. This combination minimizes quantization and truncation errors, and increases processing power for additional functions. Such functions might include: A notch filter that could cancel mechanical resonances in the system, or an estimation technique that could eliminate state sensors in a system.

The VLSI of the TMS320C14/E14 allows extra on-chip peripherals that provide additional functions. An event manager, with its own capture inputs/PWM compare outputs, simplifies system design. Up to four timers are available for sequencing of control signal processing. The on-chip peripherals of the TMS320C14/E14 make it the ideal solution for digital control.

1.3 Key Features

Some of the key features of the TMS320C14/E14 devices are listed below:

- 160-ns instruction cycle
- 256-word on-chip data RAM
- 4K-word on-chip program ROM (TMS320C14)
- 4K-word on-chip program EPROM (TMS320E14)
- EPROM code protection for copyright security
- 4K-word total external memory at full speed (microprocessor mode)
- 32-bit ALU/accumulator
- 16 x 16-bit multiplier with a 32-bit product
- 0 to 16-bit barrel shifter
- Seven input and seven output channels
- 16-bit bidirectional data bus with 50-Mbps transfer rate
- Bit-selectable I/O port (16pins)
- Serial port with programmable protocols
- Event manager with capture inputs and compare outputs
- Four independent timers (watchdog, general purpose (2), serial port)
- 15 external/internal interrupts

Section 2

Pinout and Signal Descriptions

The TMS320C14/E14 digital signal processor devices are available in plastic-leaded chip carrier (PLCC) and ceramic-leaded chip carrier (CLCC) packages.

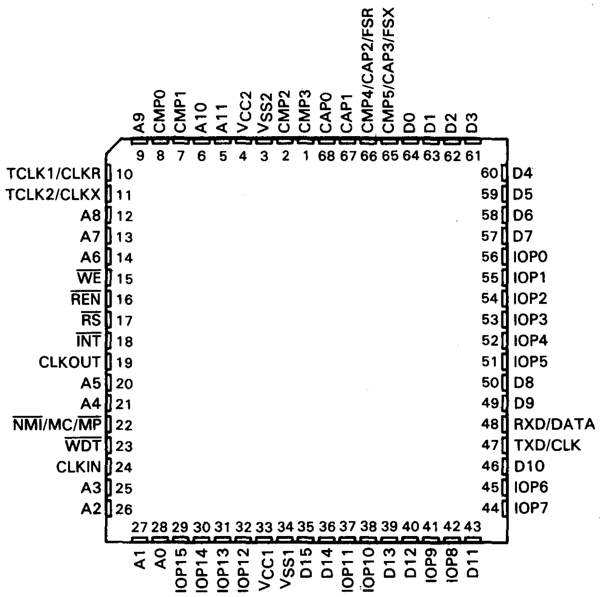
This section provides the pinouts and signal descriptions in the following sections:

- Pinouts (Section 2.1 on page 2-2)
- Pin Descriptions (Section 2.2 on page 2-3)

Electrical specifications and mechanical data are given in the data sheet in Appendix A. Refer to Appendix F for the pinouts used for EPROM programming.

2.1 Pinouts

Figure 2-1 shows the pinouts of the TMS320C14/E14 devices.



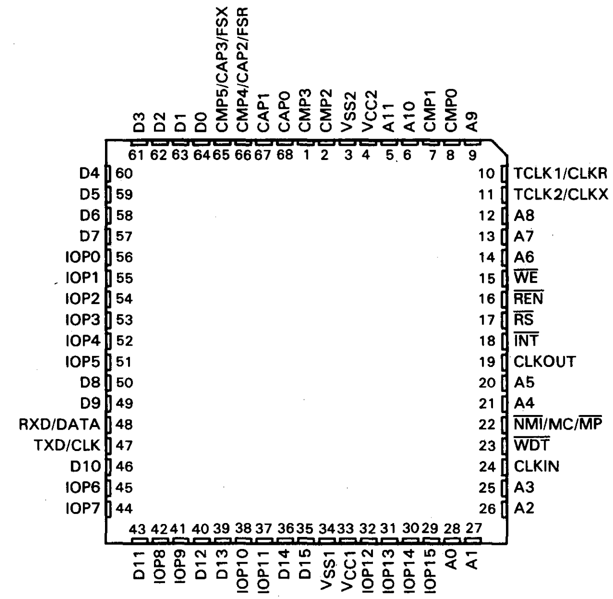


Figure 2-1. TMS320C14/E14 Pin Assignments

2.2 Signal Descriptions

This section provides the signal descriptions for the TMS320C14/E14 devices. Table 2-1 lists each signal, its pin location, operating mode (i.e., input, output, high impedance state), and description. The signals are grouped according to function and alphabetized within that grouping.

Table 2-1. TMS320C14/E14 Signal Descriptions

SIGNAL	PIN	I/O/Z†	DESCRIPTION
			ADDRESS/DATA BUSES
A11	5	O/Z	Program memory address bus A11 (MSB) through A0 (LSB) and port addresses PA2 (MSB) through PA0 (LSB). Addresses A11 through A0 are always active and never go to high impedance except during reset. During execution of the IN and OUT instructions, pins 26, 27, and 28 carry the port addresses. Pins A3 through A11 are held high when port accesses are made on pins PA0 through PA2.
A10	6		
A9	9		
A8	12		
A7	13		
A6	14		
A5	20		
A4	21		
A3	25		
A2/PA2	26		
A1/PA1	27		
A0/PA0	28		
D15 MSB	35	I/O/Z	Parallel data bus D15 (MSB) through D0 (LSB). The data bus is always in the high-impedance state except when $\overline{\text{WE}}$ is active (low). The data bus is also active when internal peripherals are written to.
D14	36		
D13	39		
D12	40		
D11	43		
D10	46		
D9	49		
D8	50		
D7	57		
D5	58		
D5	59		
D4	60		
D3	61		
D2	62		
D1	63		
D0 LSB	64		

† Input/Output/High-impedance state

Table 2-1. TMS320C14/E14 Signal Descriptions (Continued)

SIGNAL	PIN	I/O	DESCRIPTION
			INTERRUPT AND MISCELLANEOUS SIGNALS
$\overline{\text{INT}}$	18	I	External interrupt input. The interrupt signal is generated by a low signal on this pin.
$\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$	22	I	Non-maskable interrupt. When this pin is brought low, device is interrupted irrespective of the state of INTM (status register ST) bit. Microcomputer/Microprocessor select. This pin is also sampled when $\overline{\text{RS}}$ is low. If high during reset, internal program memory is selected. If low during reset, external program memory will be selected.
$\overline{\text{WE}}$	15	O	Write enable. When active low, $\overline{\text{WE}}$ indicates that device will output data on the bus.
$\overline{\text{REN}}$	16	O	Read enable. When active low, $\overline{\text{REN}}$ indicates that device will accept data from the bus.
$\overline{\text{RS}}$	17	I	Reset. When this Schmidt trigger input is low, the device is reset and PC is set to zero.
			SUPPLY/OSCILLATOR SIGNALS
CLKOUT	19	O	System clock output (one fourth CLKIN frequency).
V_{CC}	4,33	I	5-V supply pins.
V_{SS}	3,34	I	Ground pins.
CLKIN	24	I	Master clock input (from external clock source).
			SERIAL PORT AND TIMER SIGNALS
RXD/DATA	48	I/O	In the asynchronous and codec modes, this pin is the receive input. In the synchronous mode, this pin is data in while receiving data, and data out while transmitting data.
TXD/CLK	47	I/O	In the asynchronous and codec modes, this pin is the transmit output. In the synchronous mode, this pin is clock input with external clock, and clock output with internal clock.
TCLK1/CLKR	10	I	Timer 1 clock. If external clock selected, it serves as clock input to Timer 1. Can also be configured as serial port receive clock in codec mode.
TCLK2/CLKX	11	I	Timer 2 clock. If external clock selected, it serves as clock input to Timer 2. Can also be configured as serial port transmit clock in codec mode.
$\overline{\text{WDT}}$	23	O	Watchdog timer output. An active low is generated on this pin when the watchdog timer times out.

Table 2-1. TMS320C14/E14 Signal Descriptions (Concluded)

SIGNAL	PIN	I/O	DESCRIPTION
			BIT I/O PINS
IOP15 MSB IOP14 IOP13 IOP12 IOP11 IOP10 IOP9 IOP8 IOP7 IOP6 IOP5 IOP4 IOP3 IOP2 IOP1 IOP0 LSB	29 30 31 32 37 38 41 42 44 45 51 52 53 54 55 56	I/O	16 bit I/O lines that can be individually configured as inputs or outputs and also be individually set or reset when configured as outputs.
			COMPARE AND CAPTURE SIGNALS
CMP0 CMP1 CMP2 CMP3	8 7 2 1	O	Compare outputs. The states of these pins are determined by the combination of compare and action registers.
CAP0 CAP1	68 67	I	Capture inputs. A transition on these Schmidt trigger inputs causes the timer register value to be loaded into the corresponding FIFO.
CMP4/CAP2/ FSR	66	I/O	This pin can be configured as a compare output, capture input, or as an external framing input/output for the receiver of the serial port in codec mode.
CMP5/CAP3/ FSX	65	I/O	This pin can be configured as a Schmidt trigger input or as an output. That is, as a compare output, capture input, or as external framing input/output for transmit sectionof the serial port while in codec mode.

Section 3

Architecture

This section describes the architecture of the TMS320C14/E14, which is based on the TMS320C1x architecture. The term TMS320C1x architecture is used to describe the features that are generic to all members of the TMS320C1x family, e.g., TMS320C10, TMS320C14/E14, TMS320C15/E15, TMS320C17/E17, and others. For more information regarding the TMS320C1x family, refer to the TMS320C1x User's Guide. Major topics discussed in this section are listed below.

3.1 Architectural Overview

The TMS320C14/E14 architecture is based on the TMS320C1x, which utilizes a modified Harvard architecture for speed and flexibility. In a strict Harvard architecture, program and data memory lie in two separate spaces, permitting a full overlap of instruction fetch and execution. The TMS320C1x modification of the Harvard architecture allows transfers between program and data spaces, thereby increasing the flexibility of the device. This permits gain constants (or coefficients) stored in program memory to be read into RAM, allowing expansion beyond the 256 word data memory space. This also makes available immediate instructions and subroutines based on computed values. The functional block diagram shown in Figure 3-1 outlines the principal hardware structure of the TMS320C14/E14 devices. Both devices are the same except for the respective ROM/EPROM difference in program memory.

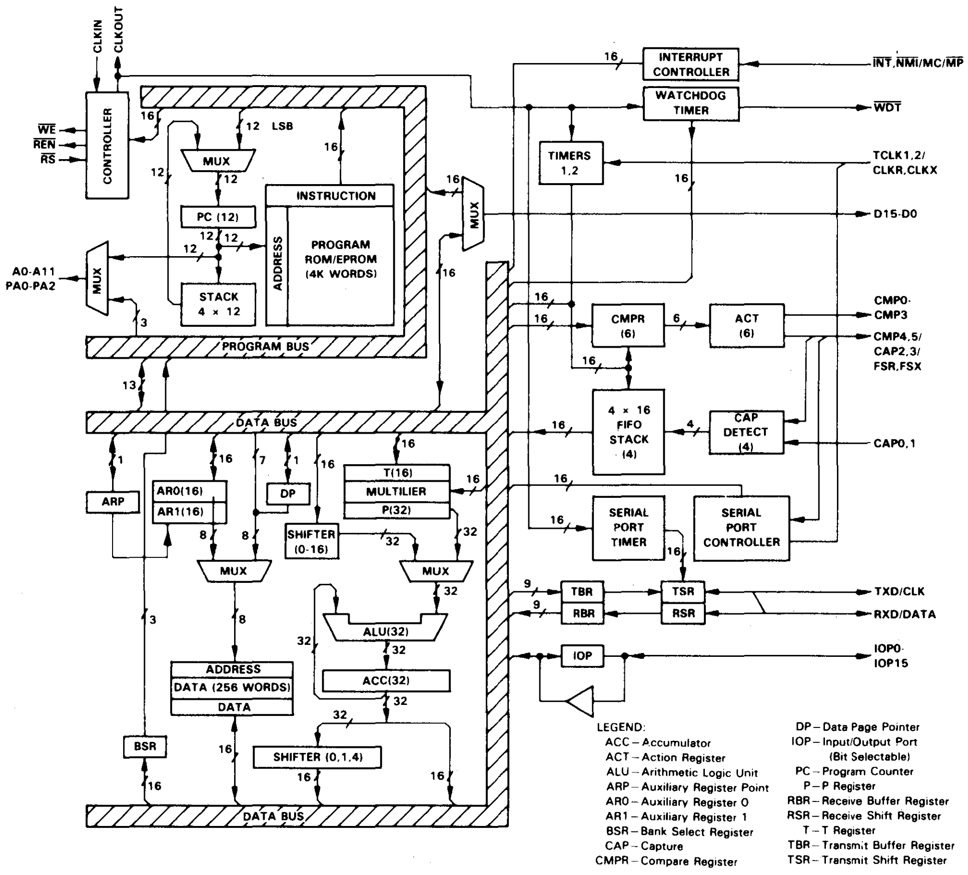


Figure 3-1. TMS320C14/E14 Functional Block Diagram

3.1.1 Processing Hardware

The TMS320C14/E14 devices contain a 32-bit ALU and accumulator for support of double-precision, two's complement arithmetic. The ALU is a general-purpose arithmetic unit that uses 16-bit words taken from data RAM or derived from immediate instructions, or uses the 32-bit result of the multiplier's product register. In addition to the usual arithmetic instructions, the ALU can perform Boolean operations, providing the bit manipulation ability required of a high-speed controller. The accumulator stores the output from the ALU and is often an input to the ALU. The accumulator is 32-bits in length and is divided into a high-order word (bits 16 through 31) and a low-order word (bits 0 through 15). Instructions are provided for storing the high-and low-order accumulator words in memory.

The multiplier performs a 16 x 16-bit two's complement multiplication with a 32-bit result in a single instruction cycle. The multiplier consists of three elements: the T register, P register, and multiplier array. The 16-bit T register temporarily stores the multiplicand; the P Register stores the 32-bit product. Multiplier values either come from the data memory, or are derived immediately from the MPYK (multiply immediate) instruction word. The fast on-chip multiplier allows the device to efficiently perform mathematically intensive algorithms such as Kalman filtering, PID loops, and lead/lag compensation.

Two shifters are available for manipulating data. The ALU barrel shifter performs a left-shift of 0 to 16 places on data memory words loaded into the ALU. This shifter extends the high-order bit of the data word and zero-fills the low-order bits for two's complement arithmetic. The accumulator parallel shifter performs a left-shift of 0, 1, or 4 places on the entire accumulator, and stores the resulting high-order accumulator bits into data RAM. Both shifters are useful for scaling and bit extraction.

The TMS320C1x devices contain a four-level hardware stack for saving the contents of the program counter during interrupts and subroutine calls. Instructions are available for saving the device's complete context. PUSH and POP instructions permit a level of nesting restricted only by the amount of available RAM.

Table 3-1 provides a summary of the processing hardware contained in the TMS320C14/E14.

Table 3-1. TMS320C14/E14 Processing Hardware Summary

UNIT	SYMBOL	FUNCTION
Accumulator	ACC	A 32-bit accumulator divided into a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). Used for storage of ALU output.
Arithmetic Logic Unit	ALU	A 32-bit two's-complement arithmetic logic unit having two 32-bit input ports and one 32-bit output port feeding the accumulator.
Auxiliary Registers	AR0,AR1	Two 16-bit registers used for data memory addressing and loop count control. Nine LSBs of each register are configured as up/down counters.
Auxiliary Register Pointer	ARP	A status bit that indicates the currently active auxiliary register.
Central Arithmetic Logic Unit	CALU	The grouping of the ALU, multiplier, accumulator, and shifters.
Data Bus	D(15-0)	A 16-bit bus used to route data to and from RAM.
Data Memory Page Pointer	DP	A status bit that points to the data RAM address of the current page. A data page contains 128 words.
Data RAM	-	256 words of on-chip random access memory containing data.
External Address Bus	A(11-0)/ PA(2-0)	A 12-bit bus used to address external program memory. The three LSBs are port addresses in the I/O mode.
Interrupt Flag	INTF	A single-bit flag that indicates an interrupt request has occurred (is pending).
Interrupt Mode	INTM	A status bit that masks the interrupt flag.
Multiplier	MULT	A 16 x 16-bit parallel hardware multiplier.
Overflow Flag	OV	A status bit flag that indicates an overflow in arithmetic operations.
Overflow Mode	OVM	A status bit that defines a saturated or unsaturated mode in arithmetic operations.
P Register	P	A 32-bit register containing the product of multiply operations.
Program Bus	P(15-0)	A 16-bit bus used to route instructions from program memory.
Program Counter	PC (11-0)	A 12-bit register used to address program memory. The PC always contains the address of the next instruction to be executed. The PC contents are updated following each instruction decode operation.
Program ROM/EPROM	-	4K words of on-chip read only memory (ROM or EPROM) containing the program code.
Shifters	-	Two shifters: the ALU barrel shifter that performs a left-shift of 0 to 16 bits on data memory words loaded into the ALU, and the accumulator parallel shifter that performs a left-shift of 0, 1, or 4 places on the entire accumulator and places the resulting high-order bits into data RAM.
Stack	-	A 4 x 12-bit hardware stack used to store the PC during interrupts or calls.

Table 3-1. TMS320C14/E14 Processing Hardware Summary (Concluded)

UNIT	SYMBOL	FUNCTION
Status Register	ST	A 16-bit status register that contains status and control bits.
T Register	T	A 16-bit register containing the multiplicand during multiply operations.

3.1.2 I/O Structure

The TMS320C1x architecture implements a variety of I/O functions that can be used for communicating with internal/external peripherals. The 16-bit parallel data bus can be utilized to perform I/O functions in two cycles using IN and OUT instructions. The I/O ports are addressed by A0 through A2 of the address bus, with A0 as the LSB of the I/O port address. The upper address bits of A3 through A11 are driven high during I/O port accesses. In the TMS320C14/E14, the I/O ports addressed can be on-chip or off-chip.

I/O design is simplified by having I/O treated the same way as memory. I/O peripherals, whether on-chip or off-chip, are mapped into the I/O space using the processor's internal/external address and data buses in the same manner as memory mapped devices.

Input/output of data, to/from an on-chip or off-chip peripheral, is accomplished by IN and OUT instructions. If external peripherals are addressed, data is transferred over the external 16-bit data bus to and from data memory by two independent strobes: read enable ($\overline{\text{REN}}$) and write enable ($\overline{\text{WE}}$). If on-chip peripherals are addressed, data is transferred over the internal data bus and the $\overline{\text{REN}}$ and $\overline{\text{WE}}$ strobes are not active.

Note:

Unlike other TMS320C1x devices, strobe $\overline{\text{REN}}$ is active for *ALL* external accesses, whether for an I/O port or program memory.

The bidirectional external data bus (D15 - D0) is always in the high-impedance state, except when $\overline{\text{WE}}$ is active (low). $\overline{\text{WE}}$ goes low during the first cycle of the OUT instruction, if external peripherals are addressed. $\overline{\text{WE}}$ also goes low during the second cycle of the TBLW instruction if external program memory is addressed. If internal peripherals are addressed, then $\overline{\text{WE}}$ remains inactive (high).

3.1.3 I/O Peripherals

The TMS320C14/E14 includes all the features of the TMS320C1x achitecture as well as some additional I/O functions. The 16-bit parallel data bus can be utilized to access external program memory and I/O functions. These external bus cycles are controlled by the write enable ($\overline{\text{WE}}$) and read enable ($\overline{\text{REN}}$) pins.

The TMS320C14/E14 has 16-pins of bit I/O that can be individually selected as inputs or outputs. There are provisions to allow setting and clearing of each pin without affecting the others. The capability to detect and match patterns on the input pins is also included. Refer to section 3.5 for more information on the bit I/O pins.

Also included in the TMS320C14/E14 are two 16-bit timers that can be used as event counters with internal or external clocks, and a Watchdog timer that is available for time-out functions. A fourth timer, the serial port baud rate generator, is intended for serial port operation, but may also be used as a general-purpose timer if sychronous/asychronous communication is not used. Associated with each timer is a 16-bit period register. Refer to Section 3.6 for more information on the timers.

The TMS320C14/E14 has an event manager that consists of a compare subsystem and a capture subsystem. The compare subsystem has six compare registers that constantly compare their outputs with one of the timers. Associated with each compare register is an action register that controls the compare output pins. The action registers determine actions that take place on output pins in case of a match between the timer and a compare register. In addition, the compare subsystem can be configured to generate six channels of high-precision PWM. The event manager also contains four capture inputs. This subsystem captures the value of a timer in a corresponding four-deep FIFO stack when a certain transition is detected on a capture input pin. Section 3.8 contains more information on the event manager.

The serial port of the TMS320C14/E14 provides three modes of operation: synchronous, asynchronous, and codec-compatible. Two protocols for interprocessor communication are supported, and a dedicated timer generates the baud rates. Refer to Section 3.8 for more information on the serial port.

The TMS320C14/E14 has a total of 15 internal/external interrupts that can be individually masked. An external non-maskable interrupt ($\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$) is also available. Each of the interrupts triggers a master interrupt. The master interrupt is controlled by the INTM bit in the status register. For more information, refer to Section 3.2.5 regarding interrupts.

A maximum of eight I/O addresses are available on the TMS320C1x and TMS320C14/E14 for interfacing to peripheral devices: eight 16-bit multiplexed input ports and eight 16-bit multiplexed output ports. To maintain compatibility with the TMS320C1x architecture, all peripherals on the TMS320C14/E14 are implemented in the I/O address space. However, the number of on-chip peripheral devices that can be accessed is greater than 16. To implement the peripherals within the address space of eight I/O ports, a dual address scheme has been adopted that includes a bank address, and a port address within that bank. This allows a user to preserve his investment in TMS320C1x software and development tools.

3.1.4 On-Chip Peripheral Register Mapping

The on-chip peripherals on the TMS320C14/E14 are controlled by peripheral registers mapped in the I/O space. A 16-bit bank select register (BSR) is used to map all these internal peripherals in the I/O space. Each peripheral register has a bank address and a port address. The bank address is selected by writing the bank address into the bank select register (BSR). The port address is specified in the IN or OUT instruction.

Table 3-2 shows the location of each peripheral register.

Table 3-2. I/O Register Map

PORT	BANK0	BANK1	BANK2	BANK3	BANK4	BANK5	BANK6	BANK7	BANKFFFF
0	IOP	WDT	TMR1	CMPR0	ACT0	SCON	FIFO0		EXT. I/O
1	DDR	WPER	TPR1	CMPR1	ACT1	SSET	FIFO1		EXT. I/O
2	BSET	WTPL4	TMR2	CMPR2	ACT2	SCLR	FIFO2	SMAT	EXT. I/O
3	BCLR	SYSCON	TPR2	CMPR3	ACT3	TBR	FIFO3	TSR	EXT. I/O
4	IF		TCON	CMPR4	ACT4	RBR	CCON	RSR	EXT. I/O
5	IM			CMPR5	ACT5	SBRG	CCLR	STMR	EXT. I/O
6	FCLR								EXT. I/O
7	BSR	BSR	BSR	BSR	BSR	BSR	BSR	BSR	BSR

3.1.5 On-Chip/Off-Chip Peripheral Selection

To select a particular register, the bank address is first stored in the BSR, enabling the selected bank. Then an access to that port is made with an IN or OUT instruction. If another register in the same bank needs to be accessed, it is not necessary to write to the bank register again. For an OUT instruction, data always appears on the data bus. The $\overline{\text{WE}}$ strobe, however, is not activated.

Port 7 is reserved in all banks for the BSR. Any I/O read/write to port 7 using IN or OUT instructions automatically access the bank select register (BSR). As shown in Figure 3-2, only the lower three bits of the BSR are used to select the bank address for the on-chip peripherals. The upper 13 bits must be zeroes.

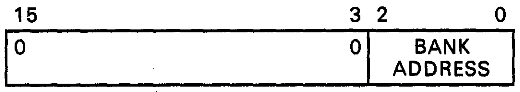


Figure 3-2. Bank Select Register

To select off-chip peripherals, bank address FFFFh must be used. Table 3-3 lists the peripheral registers available on the TMS320C14/E14 and their port and bank addresses. A more detailed description of their operation is given in the detailed descriptions of the on-chip peripherals.

Table 3-3. Peripheral Registers

REGISTER	PORT	BANK	DESCRIPTION
SYSCON	3	1	System configuration register. Configures the device as microcomputer or microprocessor.
BSR	7	All	Bank select register. Stores bank address for a register. All read/writes for port 7 go automatically to BSR. Used in selection of on-chip/off-chip peripherals.
			BIT I/O PORTS
IOP	0	0	I/O port latch. Stores data for IOP pins configured as output. Also stores data for pattern match on input IOP pins.
DDR	1	0	Data direction register. Configures IOP pins as inputs or outputs.
BSET	2	0	Bit set register. Allows setting of individual bits in IOP latch without affecting others.
BCLR	3	0	Bit clear register. Allows clearing of individual bits in IOP latch without affecting others.
			INTERRUPTS
IF	4	0	Interrupt flag register. Indicates interrupts that have been received by the device.
IM	5	0	Interrupt mask register. Directs the CPU to acknowledge or ignore an interrupt source.
FCLR	6	0	Flag register clear. Allows individual clearing of bits in IF register without affecting other bits.
			TIMERS
WDT/TMR4	0	1	Watchdog timer/Timer 4. Free-running 16-bit timer that acts as a watchdog timer. Can also be used as a fourth timer. Timer is reset to 0000h when it equals the period register.
WPER/TPR4	1	1	Watchdog timer period register. Causes the timer to reset to 0000h when its value equals the period register.
WTPL	2	1	Watchdog timer period register latch that prevents the watchdog period from being changed on the fly. This register stores the old value of WPER, used by WDT to determine its period.
TMR1	0	2	Timer 1. 16-bit timer that can be used as an event counter. TMR1 is reset to 0000h when its value value equals TPR1.
TPR1	1	2	Timer 1 period register. Causes TMR1 to reset to 0000h when its value equals TPR1.
TMR2	2	2	Timer 2. 16-bit timer that can be used as an event counter. TMR2 is reset to 0000h when its value equals TPR2.
TPR2	3	2	Timer 2 period register. Causes TMR2 to reset to 0000h when its value equals TPR2.
TCON	4	2	Timer control register. Controls operation and configuration of Timers 1 and 2, and the compare and capture subsystems.

Table 3-3. Peripheral Registers (Concluded)

REGISTER	PORT	BANK	DESCRIPTION
			EVENT MANAGER
CMPR0 CMPR1 CMPR2 CMPR3 CMPR4 CMPR5	0 1 2 3 4 5	3 3 3 3 3 3	Compare registers. Contents of compare registers are constantly being compared with Timer 1 or Timer 2. When any one of the compare registers matches the timer, it generates an action specified by an action register. In the PWM mode, a match with the timer resets the corresponding CMPx pin to a low level.
ACT0 ACT1 ACT2 ACT3 ACT4 ACT5	0 1 2 3 4 5	4 4 4 4 4 4	Action registers. Contents of action registers determine what action should take place on the CMPx pin when the compare registers match the timer, including generation of an interrupt. In the PWM mode, the action registers act as double buffers for the corresponding CMPRx register.
FIFO0 FIFO1 FIFO2 FIFO3	0 1 2 3	6 6 6 6	Four-deep FIFO stacks that capture timer values when a transition is detected on the corresponding CAPx pin.
CCON	4	6	Capture control register. Controls configuration and operation of capture inputs. It also holds the status of the FIFOs.
CCLR	5	6	CCON bit clear register. Allows clearing of individual bits in CCON without affecting other bits.
			SERIAL PORT
SCON	0	5	Serial port control register. Controls configuration and operation of serial port.
SSET	1	5	SCON bit set register. Allows setting of individual bits in SCON without affecting other bits.
SCLR	2	5	SCON bit clear register. Allows clearing of individual bits in SCON without affecting other bits.
TBR	3	5	Transmit buffer register. Stores temporary data while old data is being shifted out from transmit register.
TSR	3	7	Transmit shift register. Stores outging data currently being transmitted.
RBR	4	5	Receive buffer register. Stores temporary data while new data is being shifted into receive register.
RSR	4	7	Receive shift register. Stores incoming data currently being received.
SMAT	2	7	Serial port match word register. Serial port stays in sleep mode until match of received value with contents of SMAT register is detected.
SBRG/TPR3	5	5	Serial port baud rate generator. Sets divide ratios for serial port timer to generate baud rates for asynchronous and synchronous modes. Can also be used as period register when not used by serial port.
STMR/TMR3	5	7	Timer 3 Free-running 16-bit timer used for baud-rate generation for serial port. Can be used as general purpose timer when not used by serial port.

3.2 System Control

System control on the TMS320C14/E14 processors is provided by the program counter and stack, the SYSCON register (mode control), the external reset signal, the status register, and the interrupts. This section explains the function of these components in system control.

Note:

The TMS320C14/E14 does not have the $\overline{\text{BIO}}$ pin present on other TMS320C1x devices. An attempt to execute the BIOZ (Branch on $\overline{\text{BIO}}$ low) instruction will result in a two cycle NOP action.

3.2.1 Program Counter and Stack

The program counter and stack enable the execution of branches, subroutine calls, interrupts, and table read/table write instructions. The program counter (PC) is a 12-bit register that contains the program memory address of the next instruction to be executed. The TMS320C14/E14 reads the instruction from the program memory location addressed by the PC and increments the PC in preparation for the next instruction prefetch. The PC is initialized to zero by activating the reset ($\overline{\text{RS}}$) line.

The TMS320C14/E14 devices utilize a modified Harvard architecture in which data memory and program memory lie in two separate spaces, thus permitting a full overlap of instruction fetch and execution. Figure 3-3 outlines the overlap of the instruction prefetch and execution. On the falling edge of CLKOUT, the program counter (PC) is loaded with the address of the instruction (load PC 2) to be prefetched while the current instruction (execute 1) is decoded and begins execution. The next instruction is then fetched (fetch 2) while the current instruction continues to execute (execute 1). Even as another prefetch occurs (fetch 3), both the current instruction (execute 2) and the previous instruction are still executing. This is possible because of a highly pipelined internal structure.

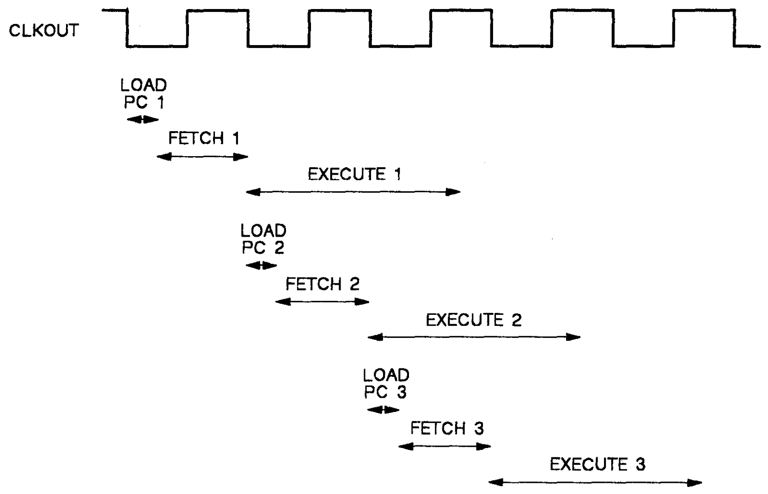


Figure 3-3. Instruction Pipeline Operation

To permit the use of external program memory, the PC outputs are buffered and sent to the external address bus pins, A11 through A0. The PC outputs appear on the address bus during all modes of operation. The nine MSBs of the PC (A11 through A3) have unique outputs assigned to them, while the three LSBs are multiplexed with the port address lines, PA2 through PA0. The port address field is used by the I/O instructions, IN and OUT.

Program memory is always addressed by the contents of the PC. The contents of the PC can be changed by a branch instruction if the particular branch condition being tested is true. Otherwise, the branch instruction simply increments the PC. All branches are absolute, rather than relative, i.e., a 12-bit value derived from the branch instruction word is loaded directly into the PC in order to accomplish the branch. When interrupts or subroutine call instructions occur, the contents of the PC are pushed onto the stack to preserve return linkage to the previous program context.

The stack is 12 bits wide and four levels deep. The PC stack is accessible through the use of the PUSH and POP instructions. The PUSH instruction pushes the twelve LSBs of the accumulator onto the top of the stack (TOS). Whenever the contents of the PC are pushed onto the TOS, the previous contents of each level are pushed down, and the fourth location of the stack is lost. Therefore, data will be lost if more than four successive pushes (stack overflow) occur before a pop. The reverse happens on pop operations. The POP instruction pops the TOS into the twelve LSBs of the accumulator. Any pop after three sequential pops yields the value at the fourth stack level. All four stack levels then contain the same value. Following the POP instruction, the TOS can be moved into data memory by storing the low-order accumulator word (SACL instruction). This allows expansion of the stack into data RAM. From data RAM, it can easily be copied into program RAM off-chip by using the TBLW (table write) instruction. In this way, the stack can be expanded to very large levels.

Note that the TBLR and TBLW instructions utilize one level of the stack; therefore, only three nested subroutines or interrupts can be accommodated without stack overflow occurring.

To handle subroutines and interrupts of much higher nesting levels, part of the data RAM or external RAM can be allocated to stack management. In this case, the TOS is popped immediately at the start of a subroutine or interrupt routine and stored in RAM. At the end of the subroutine or interrupt routine, the stack value stored in RAM is pushed back onto the TOS before returning to the main routine.

3.2.2 Microprocessor/Microcomputer Modes

The TMS320C14/E14 has two modes of operation: microprocessor and microcomputer. These modes are controlled by the MC/MP (bit 0) of the SYSCON register (see Figure 3-4). Bits 15 - 1 in this register must be set to all ones.

15	1	0
1 →	← 1	MC/ MP

Figure 3-4. Syscon Register

Writing a 1 to the MC/MP bit configures the device to microcomputer mode. In this mode, all program memory accesses are performed from internal 4K-word program memory. The 16-bit external data bus can be used for accessing off-chip peripherals. However, only the lower three address lines (PA0-PA2) are driven, while the upper 9 address lines will always be 1. Writing a 0 to the MC/MP bit (bit 0) of the SYSCON register configures the device into microprocessor mode. In this mode, all program accesses are made from external memory, and the internal ROM/EPROM is disabled.

In addition to the software control of the MC/MP bit in the SYSCON register, the TMS320C14/E14 has an external hardware option that controls this bit and configures the device in microprocessor or microcomputer mode. The $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin is sensed while the $\overline{\text{RS}}$ pin is low. If the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin is low at that time,internal program memory is disabled and all program accesses are from external memory. If the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin is high at the time, the device is placed in the microcomputer mode and all program memory accesses are from internal memory. Once $\overline{\text{RS}}$ goes high and this pin is brought high, it behaves as a normal $\overline{\text{NMI}}$ pin. The $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ (non-maskable interrupt) is edge triggered, and the pin can be brought high anytime after a reset without generating an interrupt. For more timing information, see Section 6.1.3..

A device may be initially configured by hardware to be in microcomputer or microprocessor mode. However, the user can still modify the MC/MP bit to change the mode on the device. Using a software bit, in addition to the hardware option, provides much greater flexibility. Changing the microprocessor/microcomputer modes with software allows the user to switch from internal to external memory, thus doubling the memory size to 8K words.

To provide an orderly switch of program memory, a delay of two cycles is needed. The context switches two cycles after writing to the MC/MP bit (using an OUT instruction). The two cycle delay allows a B (branch), CALL, or RET instruction to be executed and reach the desired memory location. If no location change is desired, two NOPs should be introduced to track the two-cycle delay. The SYSCON register can be accessed at bank 1h and port address 3h.

3.2.3 Reset

Reset is an external interrupt that cannot be masked. It can be used at any time to put the TMS320C14/E14 into a known state. Reset is typically applied after power-up when the machine is in a random state. The $\overline{\text{RS}}$ Schmidt trigger input pin must be held low for a minimum of five clock cycles to be effective.

Driving the $\overline{\text{RS}}$ signal low causes the TMS320C14/E14 to terminate execution, and forces the program counter to zero. $\overline{\text{RS}}$ affects various registers and status bits. At power-up the state of the processor is undefined. For correct system operation after power-up, a reset signal must be asserted low to guarantee a reset of the device. Processor execution begins at location 0, which normally contains a branch statement. This statement directs program execution of the system initialization routine.

Upon receiving an $\overline{\text{RS}}$ signal, the following actions take place in the TMS320C14/E14.

- The program counter (PC) is set to 0, and the address bus A11- A0 is placed in a high-impedance state.
- The control lines $\overline{\text{WE}}$ and $\overline{\text{REN}}$ are forced high.
- The data bus D15-DO is placed in a high-impedance state.
- Bit I/O pins IOP15-IOP0 are configured as inputs and placed in a high-impedance state.
- Pins CMP4/CAP2 and CMP5/CAP3 are configured as capture inputs and placed in a high-impedance state.
- Output pins CMP3-CMP0 are reset to 0.
- $\overline{\text{WDT}}$ pin is set to 1.
- Serial port pins TXD/CLK and RXD/DATA are placed in a high- impedance state.
- The $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin is sampled to determine whether internal or external program memory is enabled.
- $\overline{\text{RS}}$ is brought high, and the address bus A11 - A0 is cleared to all zeros in the next clock cycle.

In addition, Table 3-4 shows those registers that are also configured to the known state during reset. The status of all other registers is unknown at reset.

Table 3-4. Registers Configuration on Reset

REGISTER	STATUS	DESCRIPTION
BSR	FFFFh	Bank select register. Points to off-chip peripherals.
DDR	0000h	Data direction. All bit I/O pins configured as inputs.
IF	0000h	Interrupt flag register. All interrupt flags cleared.
IM	0FFFFh	Interrupt mask register. All interrupts masked or disabled.
WDT	0000h	Watchdog timer. Set to 0.
WPER	0FFFFh	Watchdog period. Set to maximum count.
WTPL	0FFFFh	Watchdog timer period latch. Set to maximum count.
TCON	0000h	Timer control register. Timers 1 and 2 disabled. Compare pins held at 0. Compare disabled. Capture system enabled.
SCON	0A841h	Serial port control register. Synchronous slave mode, continuous reception, no protocol, parity disabled, 8 data bits.
CCON	0000h	Capture control register. Capture on all individual pins disabled.

3.2.4 Status Register

The status register consists of five status bits. These status bits can be individually altered through dedicated instructions. In addition, the SST instruction provides for storing the status register in data memory. The LST instruction loads the status register from data memory, with the exception of the INTM bit. This bit can be changed only by the EINT/DINT (enable/disable interrupt) instructions. In this manner, the current status of the device may be saved on interrupts and subroutine calls.

Table 3-5 describes the status register bits and shows instructions that affect the status register contents. Note that several bits in the status registers are reserved and read from the status register as logic ones by the SST instruction.

Table 3-5. Status Register Field Definitions

FIELD	FUNCTION
ARP	Auxiliary Register Pointer. This single-bit field selects the AR to be used in indirect addressing. ARP = 0 selects AR0; ARP = 1 selects AR1. ARP may be modified by executing instructions that permit the indirect addressing option, and by the LARP, MAR, and LST instructions.
DP	Data Memory Page Pointer. The single-bit DP register is concatenated with the 7 LSBs of an instruction word to form a direct memory address of 8 bits. DP = 0 selects the first 128 words of data memory, i.e., page 0. DP = 1 selects page 1, the next 128 words in data memory. DP may be modified by the LST, LDP, and LDPK instructions.
INTM	Interrupt Mode Bit. When an interrupt is serviced, the INTM bit is automatically set to one before the interrupt service routine begins. INTM = 0 enables all maskable interrupts; INTM = 1 disables all maskable interrupts. INTM is set and reset by the DINT and EINT instructions, respectively. $\overline{RS}$ also sets INTM. INTM has no effect on the unmaskable $\overline{RS}$ or $\overline{NMI}$ interrupts. Note that INTM is unaffected by the LST instruction.
OV	Overflow Flag. OV = 0 indicates that the accumulator has not overflowed. OV = 1 indicates that an overflow has occurred. Once an overflow occurs, the OV remains set until a reset, BV, or LST instruction clears the OV.
OVM	Overflow Mode Bit. OVM = 0 disables the overflow mode, causing overflowed results to remain in the accumulator. OVM = 1 enables the overflow mode, causing the accumulator to be set to either its most positive or negative value upon encountering an overflow. The SOVM and ROVM instructions set and reset this bit. LST may also be used to modify the OVM.

The contents of the status register can be stored in data memory by executing the SST instruction.

Note:

If the SST instruction is executed using the direct addressing mode, the device automatically stores this information on page 1 of data memory at the location specified by the instruction, regardless of the value of the data page pointer.

If the indirect addressing mode is selected, the contents of the status register may be stored in any RAM location selected by the auxiliary register.

The SST instruction does not modify the contents of the status register. Figure 3-5 shows the position of the status bits as they appear in the appropriate data RAM location after execution of the SST instruction.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OV	OVM	INTM	1	1	1	1	ARP	1	1	1	1	1	1	0	DP

Figure 3-5. Status Register Organization

The LST instruction may be executed to load the status register. LST does not assume the status register has been stored on page one. When direct memory addressing has been used, the DP must be set to one for the LST instruction to access status bits stored on page one. The interrupt mode (INTM) bit cannot be changed by the LST instruction. However, all other status bits can be modified by this instruction.

3.2.5 Interrupts

The TMS320C14/E14 provides a total of 15 external and internal interrupts for communication with time-critical internal and external operations. Two interrupts are dedicated for external sources, and are triggered by a negative edge on pins $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ and $\overline{\text{INT}}$. The remainder of the interrupts are used to service the on-chip peripherals. All the interrupts, internal or external, are mapped into a 16-bit register called the interrupt flag register (IF). In addition, a 16-bit register called the interrupt mask register (IM), is also available to mask individual interrupts. Figure 3-6 shows the architecture of the interrupt subsystem.

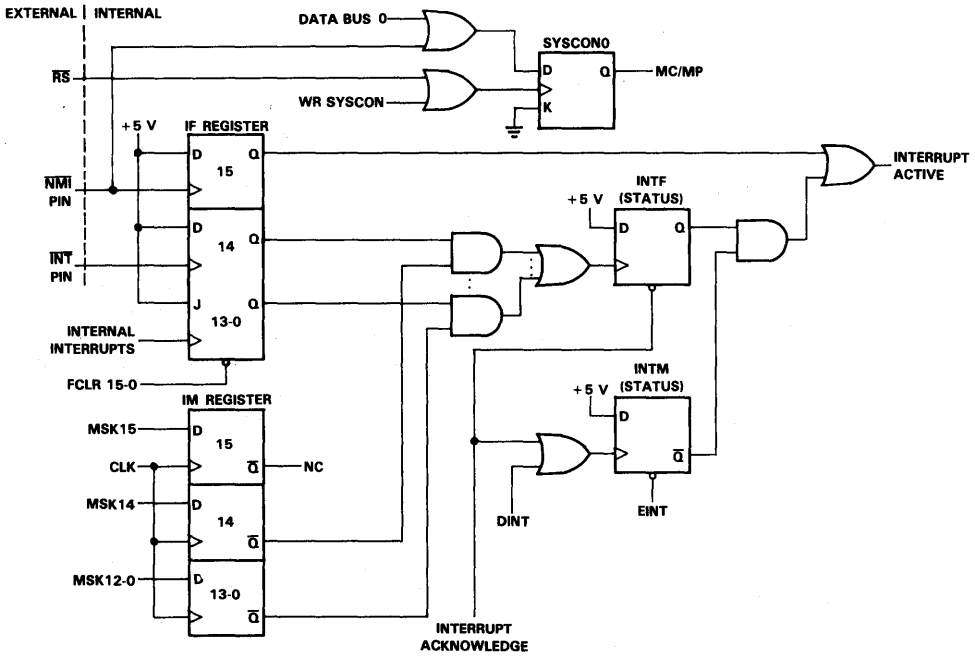


Figure 3-6. Interrupt Subsystem

When an interrupt is generated either by a peripheral or an external source, the following sequence of events occurs:

1) A bit is set to 1 in the IF register corresponding to that interrupt. This indicates an interrupt pending.

2) A check is made to determine if the corresponding bit in the IM register is 0. This indicates an interrupt is unmasked.

3) If the interrupt is unmasked, an interrupt is generated to the CPU by setting the interrupt flag (INTF). If the interrupt is masked, it continues to be latched in the IF register, and will interrupt the CPU only when unmasked.

4) If the interrupt mode (INTM) bit in the status register is 0 (CPU interrupt is enabled), the CPU responds by saving the present program counter value (PC) on the hardware stack and branching to location 2 in program memory. If the INTM bit in the status register is 1, the CPU interrupt continues to be latched in the INTF bit.

5) Sequence of events 1 through 4 is true for all interrupts except the non-maskable interrupt (NMI). In the case of the NMI, the interrupt is passed straight through to the CPU without checking the IM register or the INTM bit. The CPU responds immediately by saving the present PC value on the hardware stack and branching to location 2 in program memory.

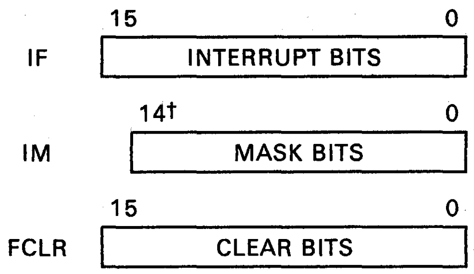
6) The INTM bit is set to 1, disabling further interrupts (except NMI).

7) The interrupt service routine, starting at address 2h, polls the IF flag register to determine which peripheral is the source of the interrupt. In addition, the corresponding flag in the IF flag register must be cleared and the CPU interrupt enabled by executing an enable interrupt (EINT) instruction.

To facilitate clearing individual flags in the IF register, an additional register called the Flag Clear, (FCLR) is also provided. When a 1 is written to a bit in the FCLR register, it clears the corresponding bit in the IF register. Writing a 0 to a bit in the FCLR register leaves the corresponding bit in the IF register unaffected. The IF, IM, and FCLR registers have a bank address of 0h, and port addresses of 4h, 5h, and 6h, respectively. Figure 3-7 shows the relationship of the IF, IM and FCLR registers.

Note:

The IF register should not be written to directly. To reduce the risk of affecting the wrong bits, all writes should be through the FCLR register.



NOTE: † Most significant **usable** bit. Writing a 1 to IM bit 15 does not mask corresponding interrupt (NMI).

Figure 3-7. IF/IM/FCLR Register Relationship

Figure 3-8 and Table 3-6 describe the individual interrupt bits of the IF register. The bits of the IM and FCLR registers correspond directly to those of the IF register. Detailed descriptions of how these interrupts are generated are found in the sections describing peripheral operation in detail. When an interrupt is received, the corresponding bit in the IF register is set to 1. This IF register bit remains set until cleared by the user. To mask an interrupt, the corresponding bit in the IM register is set to 1. The mask will remain in effect until the IM bit is cleared by the user.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMI	INT	†	TIMINT3/ STMRINT	CAP- INT3	CAP- INT2	CAP- INT1	CAP- INT0	CMP- INT1	CMP- INT0	TIM- INT2	TIM- INT1	RXINT	TXINT	WDT INT	IOPINT

NOTE: † Reserved bit

Figure 3-8. IF Register

Table 3-6. IF Register Description

BIT#	INTERRUPT	DESCRIPTION
15	NMI	Nonmaskable interrupt.
14	INT	External interrupt
13	Reserved	IF 13 should be set to 0, IM 13 should be set to 1.
12	TIMINT3/ STMRINT	Timer 3 interrupt (Timer 3 is normally used by serial port).
11	CAPINT3	Capture interrupt 3.
10	CAPINT2	Capture interrupt 2.
9	CAPINT1	Capture interrupt 1.
8	CAPINT0	Capture interrupt 0.
7	CMPINT1	Compare interrupt 1.
6	CMPINT0	Compare interrupt 0.
5	TIMINT2	Timer 2 interrupt.
4	TIMINT1	Timer 1 interrupt.
3	RXINT	Serial port receive interrupt.
2	TXINT	Serial port transmit interrupt.
1	WDTINT	Watchdog timer interrupt.
0	IOPINT	I/O port IOP interrupt.

3.3 Central Arithmetic Logic Unit (CALU)

The Central Arithmetic Logic Unit (CALU) contains a 16 x 16-bit parallel multiplier, a 32-bit Arithmetic Logic Unit (ALU), a 32-bit accumulator (ACC), and two shifters. This section describes the CALU components and their functions. Figure 3-9 is a block diagram showing the components of the CALU.

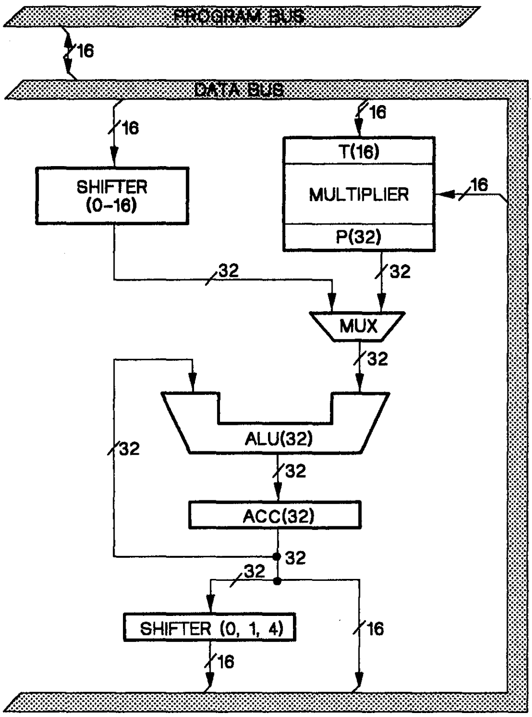


Figure 3-9. Central Arithmetic Logic Unit (CALU)

The following steps occur in the implementation of a typical ALU operation:

1) Data is fetched from the RAM on the data bus.

2) Data is passed through the barrel shifter where it can be left-shifted 0 to 16 bits, depending on the value specified by the instruction.

3) Data enters the ALU where it is operated upon and loaded into the accumulator.

4) The result obtained in the accumulator is passed through a parallel left-shifter present at the accumulator output to aid in scaling results.

5) The result is stored in the data RAM. Since the accumulator is 32 bits wide, both halves must be stored separately.

One input to the ALU is always provided from the accumulator, and the other input may be provided from the P Register of the multiplier or the barrel shifter that is loaded from data memory.

3.3.1 Shifters

Two shifters are available for manipulating data: a barrel shifter for shifting data from the data RAM into the ALU and a parallel shifter for shifting the accumulator into the data RAM (see Figure 3-9).

The barrel shifter has a 16-bit input connected to the data bus and a 32-bit output connected to the ALU The barrel shifter produces a left shift of 0 to 16 bits on all data memory words that are loaded into, subtracted from, or added to the accumulator by the LAC, SUB, and ADD instructions. The shifter zero-fills the LSBs and sign-extends the 16-bit data memory word to 32 bits by an arithmetic left-shift (i.e., the bits to the left of the MSB of the data word are filled with ones if the MSB is a one or with zeros if the MSB is a zero). This differs from a logical left-shift where the bits to the left of the MSB are always filled with zeros. A small amount of code is required to perform an arithmetic right-shift or a logical right-shift.

The following examples illustrate the barrel shifter's function:

- Data memory location 20 holds the two's-complement number: 7EBCh.

  The LAC (load accumulator) instruction is executed, specifying a left-shift of 4:

  ```
  LAC   20,4
  ```

  The accumulator then holds the following 32-bit signed two's-complement number:

31			16	15			0
0	0	0	7	E	B	C	0

  Since the MSB of 7EBCh is a zero, the upper accumulator was zero-filled.

- Data memory location 30 holds the two's-complement number: 8EBCh.

  The LAC (load accumulator) instruction is executed, specifying a left-shift of 8:

  ```
  LAC   30,8
  ```

  The accumulator then holds the following 32-bit signed two's-complement number:

31			16	15			0
F	F	8	E	B	C	0	0

Since the MSB of 8EBCh is a one, the upper accumulator was filled with ones.

Instructions are provided that perform operations with the lower half of the accumulator and a data word without first sign-extending the data word (i.e., treating it as a 16-bit rather than a 32-bit word). The mnemonics of these instructions typically end with an 'S,' indicating that sign-extension is suppressed (e.g., ADDS, SUBS). Along with the instructions that operate on the upper half of the accumulator, these instructions allow the manipulation of 32-bit precision numbers.

The parallel shifter is activated only by the SACH (store high-order accumulator word) instruction. This instruction causes the shifter to be loaded with the 32-bit contents of the accumulator. The data is then left-shifted. The most-significant 16 bits from the shifter are stored in RAM, resulting in a loss of the high-order bits of data. The contents of the accumulator remain unchanged. The parallel shifter can execute a shift of only 0, 1, or 4. Shifts of 1 and 4 are used with multiplication operations. No right-shift is directly implemented. The following example illustrates the accumulator shifter's function:

- The accumulator holds the following 32-bit signed two's-complement number:

31			16	15			0
A	3	4	B	7	8	C	D

  The SACH instruction is executed, specifying that a left-shift of four be performed on the high-order accumulator word before it is stored in data memory location 40:

  ```
  SACH 40,4
  ```

  Data memory location 40 then contains the two's-complement number: 34B7h. The accumulator still retains A34B78CDh.

3.3.2 ALU and Accumulator

The 32-bit ALU and accumulator (see Figure 3-9) implement a wide range of arithmetic and logical functions, the majority of which execute in a single clock cycle. Once an operation is performed in the ALU, the result is transferred to the accumulator where additional operations such as shifting may occur. Data that is input to the ALU may be scaled by the barrel shifter.

The ALU is a general-purpose arithmetic logic unit that operates on 16-bit data words, producing a 32-bit result. The ALU can add, subtract, and perform logical operations. The accumulator is always the destination and the primary operand. The result of logical operations is shown in Table 3-7. A data memory value (dma) is the operand for the lower half of the accumulator (bits 15 through 0). Zero is the operand for the upper half of the accumulator.

Table 3-7. Accumulator Results of a Logical Operation

FUNCTION	ACC BITS 31-16	ACC BITS 15-0
XOR	(0).XOR.(ACC (31-16))	(dma).XOR.(ACC (15-0))
AND	(0).AND.(ACC (31-16))	(dma).AND.(ACC (15-0))
OR	(0).OR.(ACC (31-16))	(dma).OR.(ACC (15-0))

The 32-bit accumulator stores the output from the ALU and is also often an input to the ALU. The accumulator is divided into two 16-bit words for storage in data memory: a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). The SACH and SACL instructions are used to store the high- and low-order accumulator words in data memory. These instructions can be used in the implementation of double-precision arithmetic.

A shifter at the output of the accumulator provides a left-shift of 0, 1, or 4 places. This shift is performed while the data is being transferred to the data bus for storage. The contents of the accumulator remain unchanged. When the high-order word is shifted left, the LSBs are transferred from the low-order word, and the MSBs are lost.

The accumulator also has the ability to simulate the effect of saturation in analog systems. This capability is implemented using the accumulator overflow saturation mode, which is controlled by the OVM (overflow mode) status register bit. The accumulator saturation mode is enabled or disabled by setting or resetting the OVM bit, respectively, through the use of the SOVM and ROVM (set and reset OVM bit) instructions. If OVM is set and accumulator operation results in an overflow, the accumulator is loaded with either the largest positive or negative number, depending on the sign of the operands and the actual result. The value of the accumulator upon saturation is 7FFFFFFFh (positive) or 80000000h (negative). If OVM is reset and an overflow occurs, the overflowed results are loaded into the accumulator without modification. (Note that logical operations cannot result in overflow.)

It is particularly desirable to enable the saturation mode when the accumulator contents represent a signal value, since without saturation mode enabled, overflows cause undesirable discontinuities in the represented waveform. When saturation mode is enabled, behavior of the accumulator more closely resembles the tendency of an analog system to limit or saturate at a maximum level when subjected to excessively large size signals.

When an overflow occurs, the OV (overflow) bit in the status register is set, regardless of whether or not the OVM bit is set. The BV (branch on overflow) instruction, which branches only if OV is set, can be used to allow programs to make decisions based on whether or not an overflow has occurred and act accordingly. Once set, OV is reset only by the BV instruction, or by directly loading the status register. Since OV is part of the status register, its state can be stored in data memory using the SST (store status register) instruction or loaded using the LST (load status register) instruction. This allows the state of OV from different program contexts to be saved independently, if desired, and examined outside of time-critical code segments.

The TMS320C14/E14 also has the capability of executing branch instructions that depend on the status of the ALU and accumulator. These instructions (BLZ, BLEZ, BGEZ, BGZ, BNZ, and BZ) cause a branch to be executed if a specific condition is met (see Section 4 for a complete list of TMS320C14/E14 instructions).

3.3.3 Multiplier, T and P Registers

The TMS320C14/E14 utilizes 16 x 16-bit hardware multiplier (see Figure 3-9), which is capable of computing a 32-bit product in a single machine cycle. The following two registers are associated with the multiplier:

- A 16-bit Temporary Register (T) that holds one of the operands for the multiplier, and
- A 32-bit Product Register (P) that holds the product.

In order to use the multiplier, an operand must first be loaded into the T register from the data bus using an LT, LTA, or LTD instruction. Then, the MPY (multiply) or MPYK (multiply immediate) instruction provides the second operand (also from the data bus). If the MPY instruction is used, the multiplier value is a 16-bit number. If the MPYK instruction is used, the value is a 13-bit immediate constant contained in the MPYK instruction word. This 13-bit constant is right-justified and sign-extended. After execution of the multiply instruction, the product will be placed in the P register. The product can then be added to, subtracted from, or loaded into the accumulator by executing a PAC, APAC, SPAC, LTA, or LTD instruction. Pipelined multiply and accumulate operations can be accomplished with the LTA/LTD and MPY/MPYK instructions. Note that no special provisions are made for the condition of 8000h × 8000h. If this condition arises, the product will be C0000000h.

Note that the contents of the P register cannot be restored without altering other registers. Interrupts are prevented from occurring until the instruction following the MPY/MPYK instruction has been executed. Therefore, the multiply instruction should always be followed by an instruction that combines the P register with the accumulator.

3.4 Memory Organization

The TMS320C14/E14 devices utilize a modified Harvard architecture with two separate spaces for data and program storage. The TMS320C14 contains 256 words of RAM for data and 4K words of ROM program space. The TMS320E14 contains 256 words of RAM for data and 4K words of EPROM program space. As mentioned in section 3.2.2, internal or external program memory space may be accessed by the CPU, depending on the mode the CPU is operating in. Mode changing, which is both hardware and software controllable, effectively doubles the amount of program memory, allowing the processor to perform multiple tasks.

3.4.1 Data Memory

The TMS320C14/E14 devices contain a 256-word x 16-bit RAM area for data storage. Figure 3-10 shows the memory map for the data RAM. The data RAM may be considered as 256 registers for temporary storage/fast access for data.

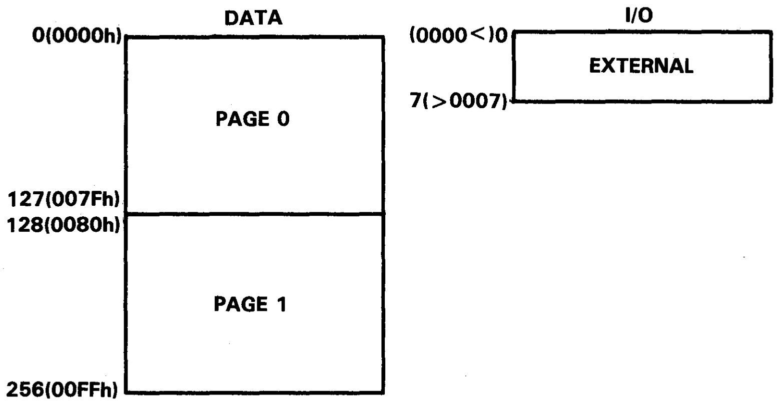


Figure 3-10. Data Memory Map

3.4.2 Program Memory

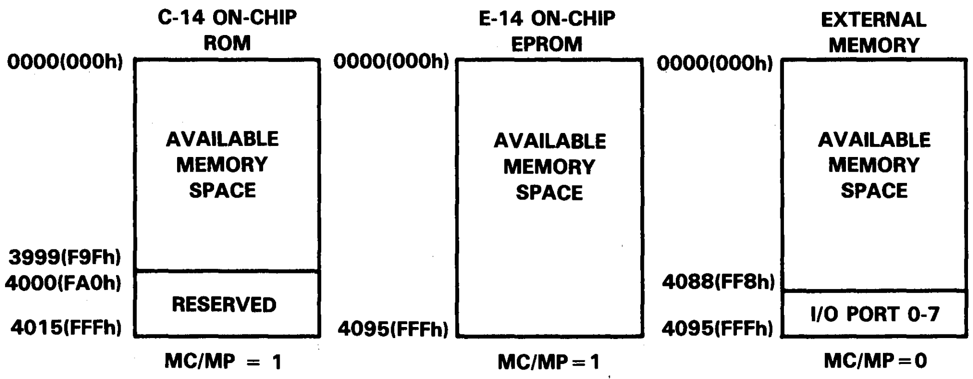


Figure 3-11. Program Memory Map

Figure 3-11 shows options for program memory storage. Aside from an on-chip program 4K-word x 16-bit ROM or EPROM, the TMS320C14/E14 can access external memory as well. The selection of memory resource is done with the MC/MP bit found as part of the SYSCON register. As stated in Section 3.2.2, setting this bit high configures the device to select the on-chip resource (ROM or EPROM). Setting this bit low configures the device to select the off-chip memory device. This device is initialized at reset when the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin is sampled, and may be subsequently altered through writes to the SYSCON register. This ability to alter the MC/MP bit at run time yields a mechanism to easily expand program memory beyond 4K-words up to a maximum of 8K-words. Note that the upper eight words of external memory are reserved for external I/O ports 0 through 7.

It should also be noted, that the TMS320E14 includes a security bit which, when set, denies read access to the EPROM. Finally, the last 96 words in the ROM on the TMS320C14 are reserved for Texas Instruments internal use.

3.4.3 Auxiliary Registers

The TMS320C14/E14 devices provide two 16-bit auxiliary registers (AR0 and AR1). This section discusses each register's function and how an auxiliary register is selected, loaded, and stored.

The auxiliary registers may be used for indirect addressing of data memory, temporary data storage, and loop control. Indirect addressing allows placement of the data memory address of an instruction operand into the least-significant eight bits of an auxiliary register. The registers are selected by a single-bit Auxiliary Register Pointer (ARP) that is loaded with a value of 0 or 1, designating AR0 or AR1, respectively. The ARP is part of the status register, and can be stored in memory.

When the auxiliary registers are autoincremented/decremented by an indirect addressing instruction or by the BANZ (branch on auxiliary register not zero) instruction, the lowest nine bits are affected (see Figure 3-12). The auxiliary registers are useful as counters when the BANZ instruction is used. This counter portion of an auxiliary register is a 9-bit counter, as shown in Figure 3-13 and Figure 3-14.

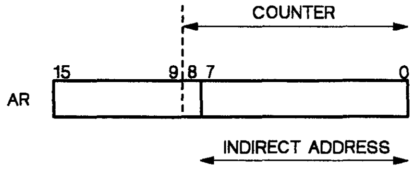


Figure 3-12. Auxiliary Register Counter

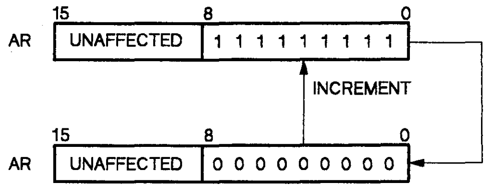


Figure 3-13. Indirect Addressing Autoincrement

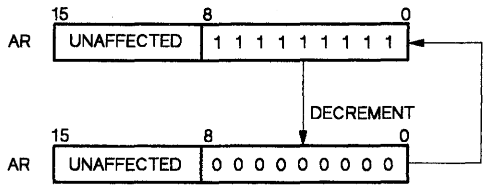


Figure 3-14. Indirect Addressing Autodecrement

The upper seven bits of an auxiliary register (i.e., bits 9 through 15) are unaffected by any autoincrement/decrement operation. This includes autoincrement of 111111111 (the lowest nine bits go to 0) and autodecrement of 000000000 (the lowest nine bits go to 111111111); in each case, bits 9 through 15 are unaffected.

The auxiliary registers can be saved in and loaded from data memory with the SAR (store auxiliary register) and LAR (load auxiliary register) instructions. This is useful for performing context saves. SAR and LAR transfer entire 16-bit values to and from the auxiliary registers even though indirect addressing and loop counting utilize only a portion of the auxiliary register. See Section 4 for programming of the indirect addressing mode.

The BANZ instruction permits the auxiliary registers to also be used as loop counters. BANZ checks if an auxiliary register is zero. If not, it decrements and branches. See Section 5.3.3 for loop code using the auxiliary registers.

3.4.4 Memory Addressing Modes

The TMS320C14/E14 can address up to 4K words of program memory and up to 256 words of data memory. Three forms of instruction operand addressing can be used: direct, indirect, and immediate addressing. Figure 3-15 illustrates operand addressing in the three modes. The addressing modes are described in detail in Section 4.1.

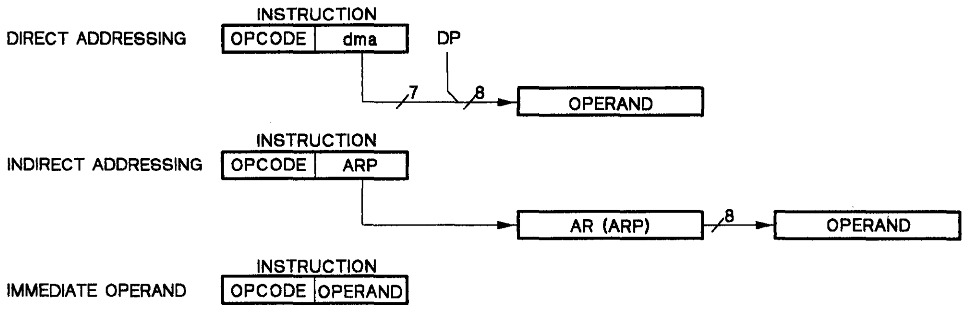


Figure 3-15. Methods of Instruction Operand Addressing

In the direct addressing mode, the 1-bit data memory page pointer (DP) selects either page 0 consisting of memory locations 0-127 or page 1 consisting of locations 128-255. The data memory address (dma), specified by the seven LSBs of the instruction concatenated with the DP, addresses the desired word within the page. Note that DP is part of the status register and thus can be stored in data memory.

Indirect addressing uses the lower eight bits of the auxiliary registers as the data memory address. This is sufficient to address all 256 data words; no paging is necessary with indirect addressing. The current auxiliary register is selected by the auxiliary register pointer (ARP). In addition, the auxiliary registers can be made to autoincrement/decrement during any given indirect instruction. Note that the increment/decrement occurs after the current instruction is finished executing.

When an immediate operand is used, it is contained within the instruction word itself.

3.5 Bit Selectable I/O Port

The TMS320C14/E14 incorporates a 16-bit I/O Port (IOP) consisting of 16 individually bit-selectable I/O pins IOP0 (LSB) through IOP15 (MSB). Key features of the IOP are listed below:

- 16 bit-selectable I/O pins
- Independant input/output pin configuration
- Independant set/clear control
- Dedicated register for output data storage
- Specific pattern detection
- Maskable interrupt

The IOP is controlled by registers configured by user software. Table 3-8 provides a summary of the IOP registers.

Table 3-8. I/O Port Register Summary

REGISTER	PORT	BANK	DESCRIPTION
IOP	0	0	I/O port latch. Stores data for IOP pins that are configured as output. Also stores data for pattern match on input IOP pins.
DDR	1	0	Data direction register.Configures IOP pins as inputs or outputs. DDR=1 configures the IOP pin as output. DDR=0 configures the pin as an input.
BSET	2	0	Bit set register. Allows setting of individual bits in IOP latch without affecting others. BSET=1 sets the IOP bit to a 1. BSET=0 leaves the IOP bit unaffected.
BCLR	3	0	Bit clear register. Allows clearing of individual bits in IOP latch without affecting others. BCLR=1 clears the IOP bit to a 0. BCLR=0 leaves the IOP unaffected.

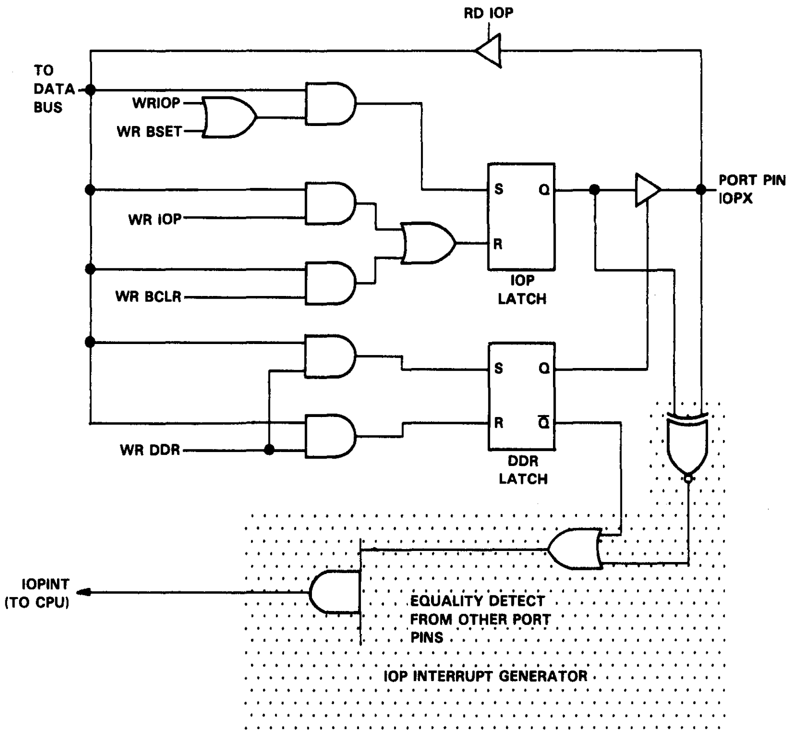


Figure 3-16. Bit Selectable I/O Port

3.5.1 Configuring Data Direction

The direction of data at the IOP pins is determined by the 16-bit data direction register (DDR). Each bit of the DDR controls a corresponding pin in the IOP port. Each pin of the IOP can be individually configured as an input or output pin by specifying the corresponding bit in the DDR (see Figure 3-16). Clearing the bit to 0 configures the corresponding pin to be an input. Setting the bit to 1 configures the pin to be an output. Bit DDR0 configures pin IOP0, while bit DDR15 configures pin IOP15. Upon reset, the DDR is configured to be 0000h, thus making the I/O pins inputs. The DDR has a bank address of 0h and a port address of 1h.

3.5.2 I/O Port Status and Control

Associated with the IOP port is a 16-bit latch called the IOP register. The IOP register provides the following functions:

1) Stores data to be transmitted on IOP pins configured as outputs

2) Stores the status of IOP pins configured as inputs when read, and a pattern to be matched with the input signals when written.

Two additional 16-bit registers, the bit set (BSET) and bit clear (BCLR) facilitate the use of the IOP register. Each bit of these registers corresponds to a bit in the IOP register. Writing a one to bit 15 of the BSET register sets bit 15 of the IOP register. Writing a one to bit 15 of the BCLR register clears bit 15 of the IOP register (see Figure 3-17)

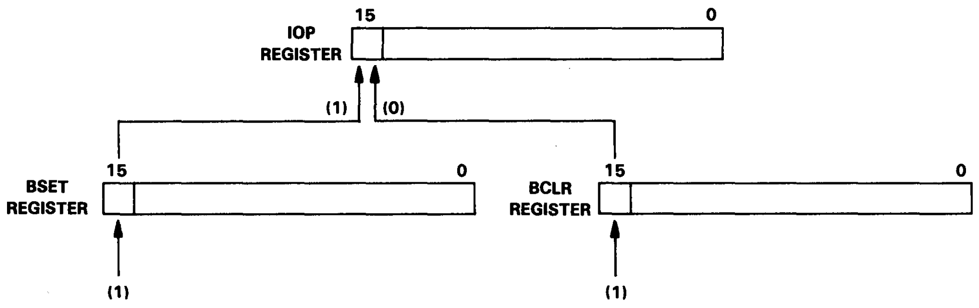


Figure 3-17. Configuring the IOP Register

Writing a zero to the BSET or BCLR has no effect on the IOP register. Reading the IOP register gives the status of the IOP pins configured as inputs, and the data to be transmitted on IOP pins configured as outputs. BSET register has a bank address of 0h and a port address of 2h. BCLR register uses bank address 0h and port address 3h. BSET and BCLR are write-only registers.

3.5.3 Input Pattern Match

The TMS320C14/E14 provides an automatic compare feature. This feature uses IOP register bits that correspond to the IOP pins configured as inputs (see Figure 3-16). Since these bits would normally provide only the status of the input pins, these bits are available for an automatic compare function. A specific pattern can be stored in the IOP register of the bits corresponding to IOP inputs. This pattern is then constantly compared to the data received on the input pins. When a match occurs, an interrupt (IOPINT) is sent to the CPU. The pattern can be written to the IOP register directly, or using the BSET and BCLR register. Writing to the IOP register directly should be avoided, since this will also affect other bits. Once a pattern is written into the IOP register, it cannot be read back. A read will give the data on the I/O port input pins, instead of the data in the IOP register bits.

The pattern in the IOP register must match all the pins configured as inputs. No subset of the input pins is possible. This feature helps the CPU detect a comparison on the input ports without polling. The IOPINT interrupt appears only the first time a match is detected, after which it appears only if the comparison becomes false and then valid again. Interrupt IOPINT sets bit 0 in the interrupt flag register IF. IOPINT can be disabled by setting bit 0 in the interrupt mask register IM.

Note:

To reduce the risk of affecting the wrong bits, all writes to the IOP register should be through the BSET and BCLR registers.

3.6 Timers

The TMS320C14/E14 is equipped with four independent timers; a watchdog timer, two general-purpose timers, plus the internal clock/baud-rate generator. The first three 16-bit timers can be used as software development aids as well as integrated into the DSP function. The fourth timer, the internal clock/baud rate generator, is intended for use with the serial port, but may also be used as a general purpose timer if not required for serial communication.

Each of the four timers contain the following features:

- A 16-bit free-running timer
- A 16-bit period register
- A dedicated maskable interrupt

Table 3-9. Timer Module Register Summary

REGISTER	ADDR	BANK	DESCRIPTION
WDT/TMR4	0	1	Watchdog timer/Timer 4. Free-running 16-bit timer that acts as a watchdog timer. Can also be used as a fourth timer. The timer is reset when it equals the period register.
WPER/TPR4	1	1	Watchdog timer period register. It causes the timer to be reset to 0000h when it equals the period register.
WTPL	2	1	Watchdog timer period register that prevents the watchdog period from being changed on the fly. This register stores the old value of WPER, used by WDT to determine its period.
TMR1	0	2	Timer 1. Free running 16-bit timer that can be used as an event counter. It is reset to 0000h when its value equals TPR1.
TPR1	1	2	Timer 1 period register. It causes TMR1 to be reset to 0000h when its value equals TPR1.
TMR2	2	2	Timer 2. Free running 16-bit timer that can be used as an event counter. It is reset to 0000h when its value equals TPR2.
TPR2	3	2	Timer 2 period register. It causes TMR2 to be reset to 0000h when its value equals TPR2.
TCON	4	2	Timer control register. Controls operation and configuration of Timers 1, 2, and event manager.
STMR/TMR3	5	7	Timer 3. Free-running 16-bit timer used for baud-rate generation for serial port. Can be used as general purpose timer when not used by serial port.
SBRG/TPR3	5	5	Baud rate generator. Sets divide ratios for serial port timer to generate baud rates for asynchronous and synchronous mode. Can also be used as period register when not used by the serial port.

3.6.1 Watchdog Timer

The TMS320C14/E14 is equipped with a free running 16-bit Watchdog Timer (WDT). This timer (Figure 3-18) is designed to prevent software hang-ups. If the WDT is allowed to time out, a pulse is generated on pin $\overline{\text{WDT}}$ that can be used to reset the TMS320C14/E14 and/or external hardware. In addition to the external pulse, an interrupt (WDTINT) is also routed to the CPU. In this way, a mechanism for recovering from software faults is provided.

To reset the watchdog timer (WDT), a pattern ABCDh followed by 2345h should be written to the WDT register with consecutive OUT instructions.

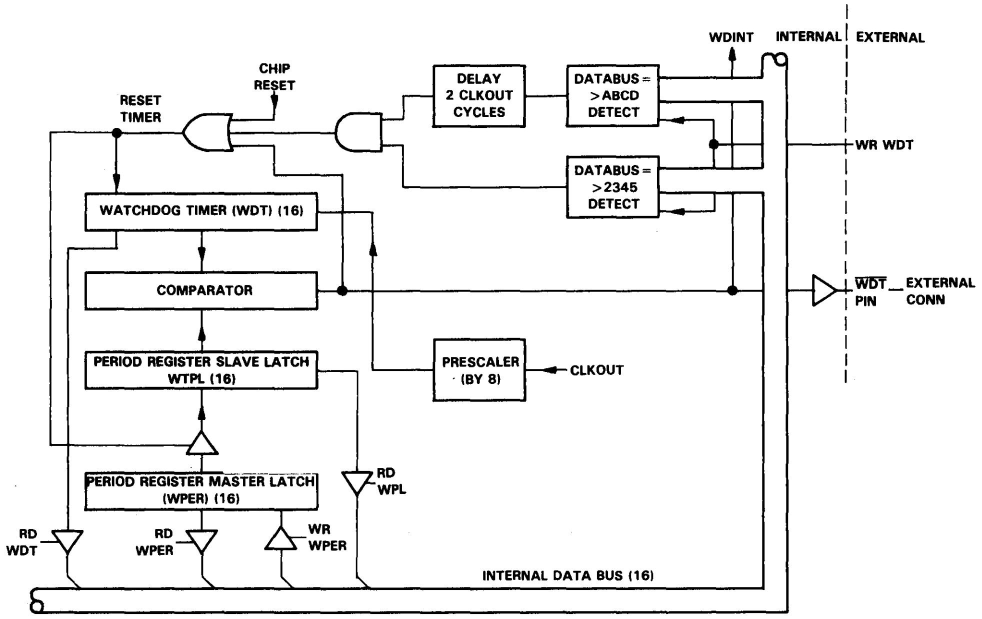


Figure 3-18. Watchdog Timer Module

The WDT is a read-only register and cannot be updated. However, WDT is reset to 0000h under the following conditions:

1) Pattern 0ABCDh followed by 02345h is written to the WDT by two consecutive OUT instructions. (Writing anything else has no affect.)

2) WDT times out when its contents match that of the period register latch WTPL.

3) Device is reset ($\overline{\text{RS}}$ driven low).

In the case of a timeout, interrupt WDTINT is generated along with a pulse on pin $\overline{\text{WDT}}$. If needed, the contents of the WDT can be read. WDT has a bank address of 1h and a port address of 0h.

3.6.1.1 WDT Period Register

Associated with the WDT are two 16-bit registers, WPER and WTPL. These registers operate in a master/slave relationship to store the maximum time that could occur between updates to the WDT. Timeout on the WDT occurs if it is not reset by consecutive writes, and its value matches with the contents of the WTPL.

The WPER is directly accessible from the data bus. It is double-buffered to prevent accidental changes to the watchdog period register. Associated with the WPER is a 16-bit buffer latch (WTPL) that stores the old value of the WPER if the WPER is changed. The WDT timer compares its value with the WTPL instead of the WPER. If the WPER is updated in the meantime, it has no affect on WTPL or on the period of the WDT. The WTPL is not directly accessible for writing and is only updated when the WDT is reset to 0000h. At that time the new period value is transferred from the WPER to the WTPL. However, the WTPL can be read if needed. The WTPL uses bank address 1h and port address 2h. The WPER uses bank address 1h and port address 1h. At reset, the WPER is set to 0FFFFh (maximum timeout value).

3.6.1.2 WDT Input

Input to the WDT is CLKOUT (CLKIN/4) divided by 8. At 25.6 MHz, this gives a clock rate of 800 KHz. The external pulse generated on pin $\overline{\text{WDT}}$ when a timeout occurs is one WDT clock cycle or 8 CLKOUT periods. The maximum value that can be entered in the WPER is FFFFh, which gives a maximum period of 81.9 ms at 25.6 MHz. The interrupt routed to the CPU, WDTINT, sets bit 1 in the interrupt flag (IF) register. WDTINT can be disabled by setting bit 1 in the interrupt mask (IM) register. Although the WDT is not writable (except for the reset pattern), it can still be read and used as a regular timer in the system. The WDT can be used as a general purpose timer if not being used for the watchdog function. Table 3-9 provides a summary of the registers used by the WDT.

3.6.2 General Purpose Timers

The TMS320C14/E14 is equipped with two 16-bit general-purpose incrementing timers (TMR1 and TMR2) that can be used as event counters (see Figure 3-19). Input to the timers can be from internal or external sources. Associated with the timers are two 16-bit period registers. When a timer value matches the value in the corresponding period register, the timer is reset in the next count cycle. A corresponding interrupt to the CPU is then generated.

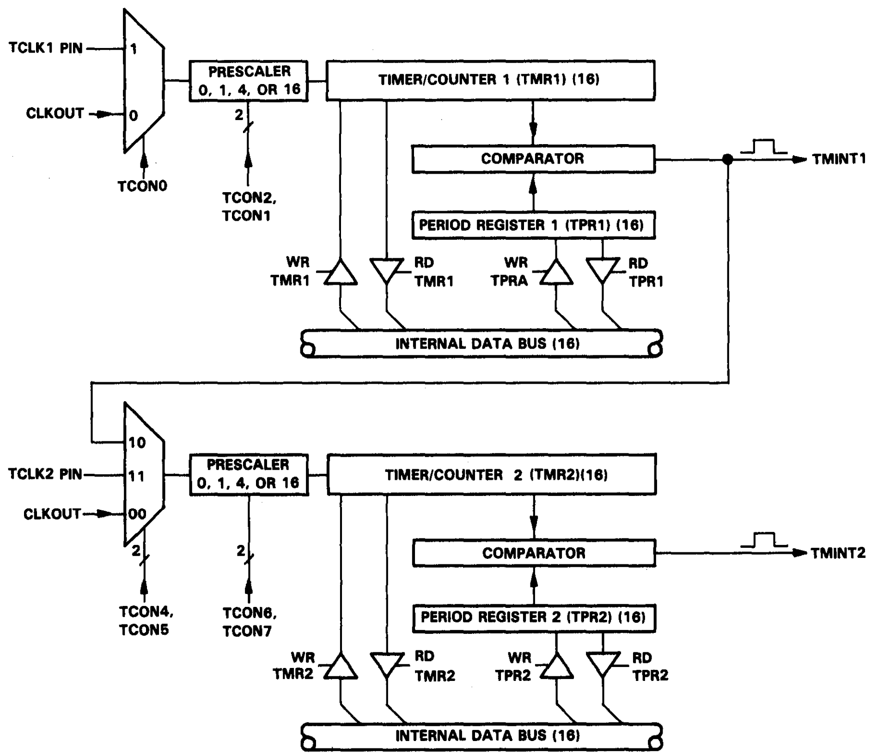
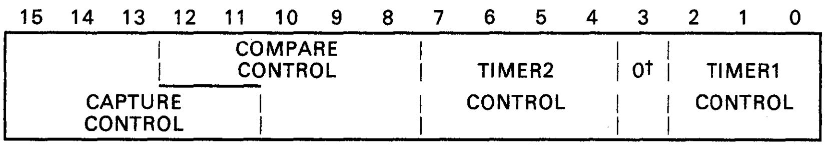


Figure 3-19. Timer1 and Timer2 Block Diagram

The operation of the timers is controlled by the timer control (TCON) register.Figure 3-20 shows the usage of the TCON register bits. The TCON register also controls the compare and capture subsystems. The TCON register uses bank register 2h and port address 4h. Table 3-10 shows a summary of the TCON register bit functions.

Bit 0 of the TCON selects the clock source for TMR1. If bit 0 is 0, TMR1 input is from internal clock (CLKOUT). If bit 0 is 1, TMR1 input is from external source on pin TCLK1. The maximum clock rate from an external source is CLKOUT/2, and TMR1 is incremented on the rising edge of TCLK1. If desired, the input to TMR1 can be prescaled. whether internal or external clock is selected. Bits 1 and 2 select the prescale value. The clock input can have a prescale factor of 1 (no prescaling), 4, or 16. In addition, prescale bits 1 and 2 can also be used to stop or disable TMR1 at any time.



NOTE: † Reserved bit, should be cleared to 0.

Figure 3-20. TCON Register Timer Bit Configuration

Table 3-10. TCON Register Timer Description

BIT	FUNCTION
7, 6	Prescale select for TMR2. 00 = Timer stop. Stops timer as soon as set to 00 01 = Prescale of 4. Divides input by 4. 10 = Prescale of 16. 11 = Prescale of 1. No prescaling. Timer disabled on reset.
5, 4	Selects clock source for TMR2. 00 = internal clock. 10 = TMR1 output is input clock. 11 = external clock.
3	Reserved. Should always be cleared (0).
2, 1	Prescale select for TMR1. 00 = Timer stop. Stops timer as soon as set to 00 01 = Prescale of 4. Divides input clock by 4. 10 = Prescale of 16. Divides input clock by 16. 11 = Prescale of 1. No prescaling. Timer disabled on reset.
0	Clock select for TMR1. 0 = Internal clock (CLKOUT). 1 = External clock. Selects clock input on TCLK1.

NOTE: All bits cleared to 0 on reset.

Bits 4 and 5 specify the clock source for timer 2 (TMR2). In addition to internal (CLKOUT) and external clock, output of TMR1 can be specified as input to TMR2. It is thus possible to have a 32-bit timer with 8-bits of prescale. If external clock is specified, input to TMR2 is from pin TCLK2. Bits 6 and 7 select the prescale values for TMR2. Bits 6 and 7 can also be used to hold or disable TMR2.

Both TMR1 and TMR2 can be written to and read from. TMR1 has a 16-bit period register (TPR1) associated with it. When the value of TMR1 matches TPR1, TMR1 is reset to 0000h, and an interrupt (TIMINT1) to the CPU is generated. TIMINT1 sets bit 4 in the interrupt flag (IF) register. TIMINT1 can be disabled by setting bit 4 in the interrupt mask (IM) register. TMR1 register has a bank address of 2h and a port address of 0h. TPR1 has a bank address of 2h and a port address of 1h. At reset, TPR1 is set to FFFFh.

The TMR2 has a 16-bit period register (TPR2) associated with it. When the value of TMR2 matches TPR2, TMR2 is reset to 0000h and an interrupt (TIMINT2) to the CPU is generated. TIMINT2 sets bit 5 in the IF register. It can be disabled by setting bit 5 in the IM register. TMR2 has a bank address of 2h and a port address of 2h. The TPR2 has a bank address of 2h and a port address of 3h. At reset, TPR2 is set to FFFFh.

3.6.3 Serial Port Baud Rate Generator (as a timer)

The serial port baud rate generator (STMR) is primarily intended for use with the serial port during synchronous/asynchronous communication. It can, however, be used as a 16-bit general purpose timer if such communications are not used. The input to the STMR is always CLKOUT. Like the other timers, the STMR has a 16-bit period register (SBRG) associated with it, and an interrupt (STMRINT) can be generated to the CPU that sets bit 12 (which is maskable) in the interrupt flag (IF) register.

3.7 Event Manager

The TMS320C14/E14 comes with an event manager that consists of compare and capture subsystems. The event manager uses TMR1 and TMR2 as associated clock sources, and has eight pins divided between the compare (CMP) and capture subsystems.

The compare subsystem consists of compare and action registers and output pins. The compare registers compare their values with those of the timers. When a match is detected, the action registers specify an action that takes place on the output pins. The action registers can specify that an interrupt to the CPU should be generated. The compare subsystem also has a high precision PWM mode. In the PWM mode, six channels of high precision PWM outputs are available.

The capture subsystem consists of four FIFO's and input pins. When a change is detected on the input pins, the current values of TMR1 or TMR2 are captured in the FIFOs. An interrupt to the CPU can also be generated at this time.

3.7.1 Compare Subsystem

The compare subsystem includes the following features:

- Six 16-bit compare registers
- Six 16-bit action registers
- Six possible compare output pins
- High precision PWM mode with double-buffered PWM registers and six PWM channels
- Two maskable interrupts

The 16-bit compare (CMPRx) registers are used for comparison with a selected timer, while the 16-bit action (ACTx) registers control the action of the output pins (see Figure 3-21). The action registers can specify generation of two interrupts (CMPINT0 and CMPINT1). The compare subsystem has four dedicated output pins (CMP0 through CMP3) and two pins that can be specified as compare outputs or capture inputs. Table 3-11 summarizes the registers associated with the compare subsystem.

Table 3-11. Compare Subsystem Register Summary

REGISTER	ADDR	BANK	DESCRIPTION
TCON	4	2	Timer control register. Controls operation and configuration of Timers 1 and 2 and compare and capture subsystems.
CMPR0 CMPR1 CMPR2 CMPR3 CMPR4 CMPR5	0 1 2 3 4 5	3 3 3 3 3 3	Compare registers. Contents of compare registers are constantly being compared with Timer 1 or Timer 2. When any one of the compare registers matches the timer, it generates an action specified by an action register. In the PWM mode, a match with timer resets the corresponding CMPx pin to a low level.
ACT0 ACT1 ACT2 ACT3 ACT4 ACT5	0 1 2 3 4 5	4 4 4 4 4 4	Action registers. Contents of action registers determine what action should take place on the CMPx pin when the compare registers match the timer, including generation of an interrupt. In the PWM mode, the action registers act as double buffers for the corresponding CMPRx register.

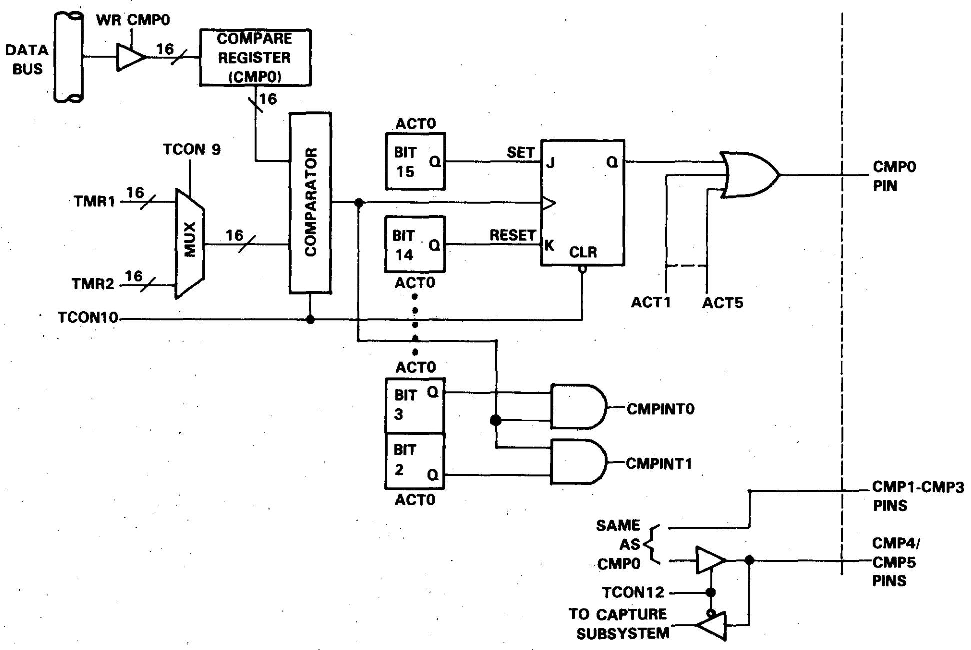


Figure 3-21. Compare Module

The compare registers are used to store certain values or periods. These are compared with timer values. Bits 8 through 12 of the timer control (TCON) register control the operation of the compare registers. Figure 3-22 and Table 3-12 show the functions of the TCON register bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			COMPARE CONTROL					TIMER2 CONTROL				0†	TIMER1 CONTROL		
CAPTURE CONTROL															

NOTE: † Reserved bit. Should be cleared to 0.

Figure 3-22. TCON Register Compare Bit Configuration

Table 3-12. TCON Compare Register Description

BIT #	DESCRIPTION
12	CMP5/CAP3 Configure. Set to zero on reset. 0 = Configures pin CMP5/CAP3 as capture input. 1 = Configures pin CMP5/CAP3 as compare output.
11	CMP4/CAP2 Configure. Set to zero on reset. 0 = Configures pin CMP4/CAP2 as capture input. 1 = Configures pin CMP4/CAP2 as compare output.
10	Compare Enable. Set to zero on reset. 0 = Disables the compare subsystem. Pins CMP0 - CMP5 are held at 0. Compare registers CMPR0 through CMPR5 are reset and held at 0000h, compare interrupts are disabled. 1 = Enables the compare subsystem. Allows normal operation of compare subsystems.
9	Timer select for Compare subsystem. Set to zero on reset. 0 = Timer 1. Select Timer 1 for use in comparison 1 = Timer 2. Select Timer 2 for use in comparison
8	Compare mode selection. Set to Zero on reset. 0 = Normal compare subsystem operation. 1 = High precision PWM mode.

3.7.1.1 Compare Registers

Once a value is stored in each of the six compare registers, the compare system works independently without any intervention from the CPU. Each of the compare registers (CMPR0 through CMPR5) is constantly comparing itself with either TMR1 or TMR2.

When a match is detected between the value stored in the compare register and the value in the timer, a match signal enables the corresponding action register to carry out an action on a compare output pin. All compare registers have the same bank address of 3h, and port addresses of 0h through 5h. They can be read or written to using these addresses.

3.7.1.2 Action Registers

Each of the action registers is dedicated to a compare register, and is capable of controlling all six compare output pins (CMP0 through CMP5). When a match signal is received from the corresponding compare register (i.e., value of timer matches compare register value), each of the compare output pins can either be set high, reset low, or toggled. In addition, each action register can specify generation of one or both interrupts (CMPINT0 AND CMPINT1). Figure 3-23 shows the configuration of the bits in each action register.

15 14	13 12	11 10	9 8	7 6	5 4	3	2	1 0
CMP0	CMP1	CMP2	CMP3	CMP4	CMP5	CMPINT0	CMPINT1	Reserved

Figure 3-23. ACTx Register Configuration

Bits 4 through 15 of the action register control (in pairs) the action of CMP0 through CMP5. Bits 2 and 3 control generation of CMPINT0 and CMPINT1. Bits 0 and 1 are reserved. ACT0 through ACT5 have the same bank address of 4h and port addresses 0h through 5h. They can be read from or written to using these addresses.

When the value in the specified timer equals the value in a compare register, the corresponding action register will specify an action depending upon the configuration as shown in Table 3-13.

Table 3-13. Action Register Description

BIT #	FUNCTION	DESCRIPTION
15,14	Set/Reset CMP0	00 = No action taken on pin CMP0. 01 = Resets pin CMP0 to a low level. 10 = Sets pin CMP0 to a high level. 11 = Toggle pin CMP0.
13,12	Set/Reset CMP1	00 = No action taken on pin CMP1. 01 = Resets pin CMP1 to a low level. 10 = Sets pin CMP1 to a high level. 11 = Toggle pin CMP1.
11,10	Set/Reset CMP2	00 = No action taken on pin CMP2. 01 = Resets pin CMP2 to a low level. 10 = Sets pin CMP2 to a high level. 11 = Toggle pin CMP2.
9,8	Set/Reset CMP3	00 = No action taken on pin CMP03. 01 = Resets pin CMP3 to a low level. 10 = Sets pin CMP3 to a high level. 11 = Toggle pin CMP3.
7,6	Set/Reset CMP4	00 = No action taken on pin CMP4. 01 = Resets pin CMP4 to a low level. 10 = Sets pin CMP4 to a high level. 11 = Toggle pin CMP4.
5,4	Set/Reset CMP5	00 = No action taken on pin CMP5. 01 = Resets pin CMP5 to a low level. 10 = Sets pin CMP5 to a high level. 11 = Toggle pin CMP5.
3	Set CMPINT0	0 = No interrupt generated. 1 = Generates interrupt and sets bit 6 in IF register.
2	Set CMPINT1	0 = No interrupt generated. 1 = Generates interrupt and sets bit 7 in IF register
1	Reserved	Should be set to 1.
0	Reserved	Should be set to 1.

3.7.1.3 Compare pins

There are four output pins (CMP0 through CMP3), dedicated to the compare subsystem. Two additional pins are also available for the compare subsystem. These pins (CMP4/CAP2 and CMP5/CAP3) are shared with the capture (input) subsystem. CMP4/CAP2 and CMP5/CAP3 are configured by bits 11 and 12 of the TCON register. Each of the compare output pins can be controlled by all six action registers to create specific waveforms. However, if two action registers specify simultaneous action (i.e., set is specified by one action register, while toggle is specified by other action register), unpredictable action can occur. Pins CMP0 through CMP3 are set low on reset. Pins CMP4/CAP2 and CMP5/CAP3 are configured as inputs and put in a high impedance at reset.

3.7.1.4 Compare Interrupts

Bits 2 and 3 of each action register can generate interrupts (CMPINT0 and CMPINT1) to the CPU, when the corresponding compare register generates a match or EQ signal. CMPINT0 sets bit 6 in the interrupt flag register (IF). CMPINT1 sets bit 7 of IF. Both interrupts can be masked by using the interrupt mask register (IM).

3.7.1.5 High Precision PWM Mode

The compare subsystem has a mode for generating high precision pulse width modulation (PWM) outputs (refer to Figure 3-24). In the high precision mode, the pulse width on pins CMPx has two extra bits of resolution. Thus, the pulse width in this mode can be specified with a minimum resolution of 40 ns @ 25.6 Mhz (vs 160 ns in the normal compare mode). Table 3-14 gives a comparison between the high precision PWM mode and the normal compare mode.

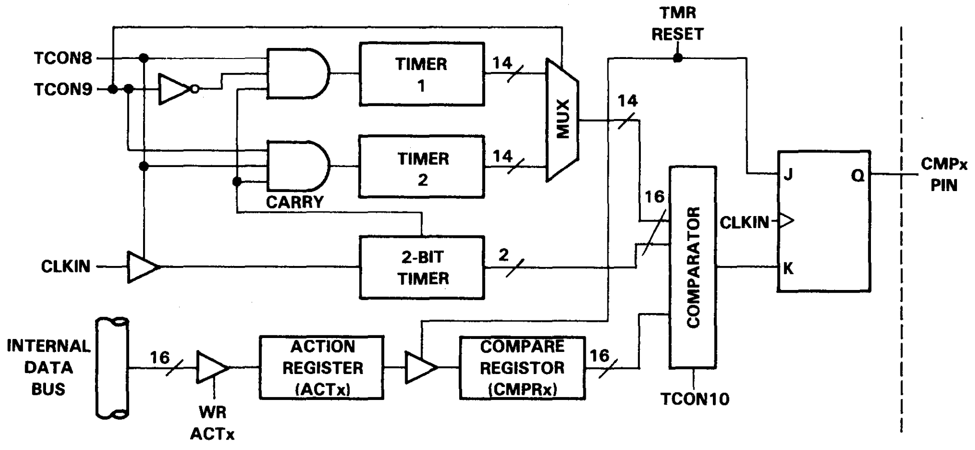


Figure 3-24. Compare Subsystem in PWM Mode

Table 3-14. PWM Resolution Bits Comparison

PWM Frequency (in KHz)	Bits of Resolution	
	Normal Compare Mode	High Precision PWM Mode
100	6	8
25	8	10
6.26	10	12
1.506	12	14

The PWM mode is enabled by setting bit 8 of TCON register to 1. In this mode, each of the six output pins is uniquely associated with one compare and action register, and each work independently. The pulse width of each compare output pin is determined by the associated compare register, while the overall period is determined by the selected timer period register.

To begin using the PWM mode, TMR1 or TMR2 is selected (prescale of 1, internal source), and its period register is loaded with the period value. The selected timer, clocked by CLKOUT, begins to count (refer to Figure 3-25) until the value of the 14 LSBs of the timer match the 14 MSBs of the compare register. The two LSBs of the specified compare register are then used to count the number of CLKIN pulses before the associated compare pin is reset (refer to Figure 3-25). If the two LSBs are 00, a transition occurs immediately on the compare pin. If 01 appears in the two LSBs, a transition occurs after one CLKIN cycle. If 10 appears in the two LSBs, a transition occurs after two CLKIN cycles. If 11 appears in the two LSBs, a transition occurs after three CLKIN cycles. In this way, the resolution of the PWM mode is increased by a factor of four. The timer continues to count until its 14 LSBs match the 14 LSBs of its associated period register. When this occurs, all compare pins are set (refer to Figure 3-25), the timer is reset to 0000h, and the compare registers are loaded with the values in their associated action registers. Since only the 14 LSBs are used in comparing the timer and its period register, the two MSBs in the period register **MUST BE ZEROS**. The timer begins counting from 0000h, and a new PWM cycle begins.

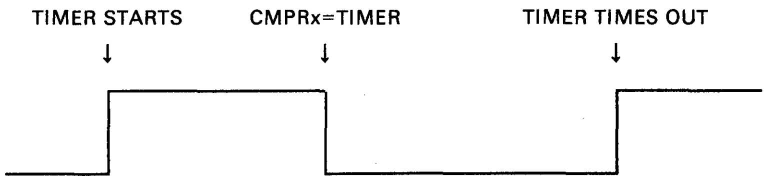


Figure 3-25. CMPx Pin Level

Note that the action of the CMPx pins are no longer controlled by the action registers. The action registers also do not generate any interrupts. Interrupt CMPINT0 is dedicated to register CMPR4, and is generated when contents of CMPR4 match contents of TMRx. Interrupt CMPINT1 is dedicated to CMPR5, and is generated when contents of CMPR5 match contents of TMRx. Either TMR1 or TMR2 can be used for comparison in this mode. Note that the selected timer has to be clocked at the CLKOUT rate with a prescale of one.

In summary:

1) TMR1 or TMR2 is selected, and the associated timer period register is loaded with the pulse period value.

2) TMRx counts until its 14 LSBs match the 14 LSBs of the compare register.

3) The two LSBs of the compare register decide which quarter phase will cause the compare pin to reset.

4) When contents of register CMPR4 matches value of TMRx, pin CMP4 is reset to 0, and interrupt CMPINT0 is generated.

5) When contents of register CMPR5 matches value of TMRx, pin CMP5 is reset to 0, and interrupt CMPINT1 is generated.

6) TMRx continues to count until its 14 LSBs match the 14 LSBs of its associated period register. Recall that the two MSBs in the period register must be zeros.

7) **ALL** compare pins are set.

8) TMRx is reset to 0000h and begins counting.

9) The value in the action register is automatically written into its associated compare register.

10) New PWM cycle starts.

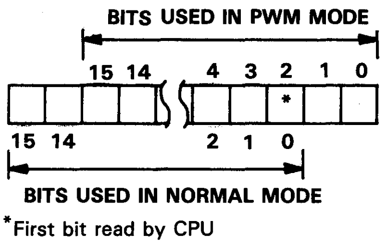


Figure 3-26. TMR Bit Configuration

3.7.2 Capture Subsystem

The capture subsystem provides a logging function for up to four different events (transitions). The capture subsystem includes the following features:

- Four 16 x 4 FIFO stacks
- Four possible Schmidt trigger input capture pins
- User specified edge detection
- FIFO status indicator bits
- Optional timer selection

The operation of the capture subsystem is controlled by the timer control (TCON) and capture control (CCON) registers. Figure 3-27 shows the structure of the capture subsystem. Table 3-15 summarizes the registers associated with the capture subsystem.

Table 3-15. Capture Subsystem Register Summary

REGISTER	ADDR	BANK	DESCRIPTION
TCON	4	2	Timer control register. Controls operation and configuration of TMR1, TMR2, compare and capture subsystems.
FIFO0 FIFO1 FIFO2 FIFO3	0 1 2 3	6 6 6 6	Four deep FIFO stacks that capture the timer values when a transition is detected on the associated CAPx pin.
CCON	4	6	Capture control register. Controls configuration and operation of capture inputs. It also holds the status of the FIFOs.
CCLR	5	6	CCON bit clear register. Allows clearing of individual bits in CCON without affecting other bits.

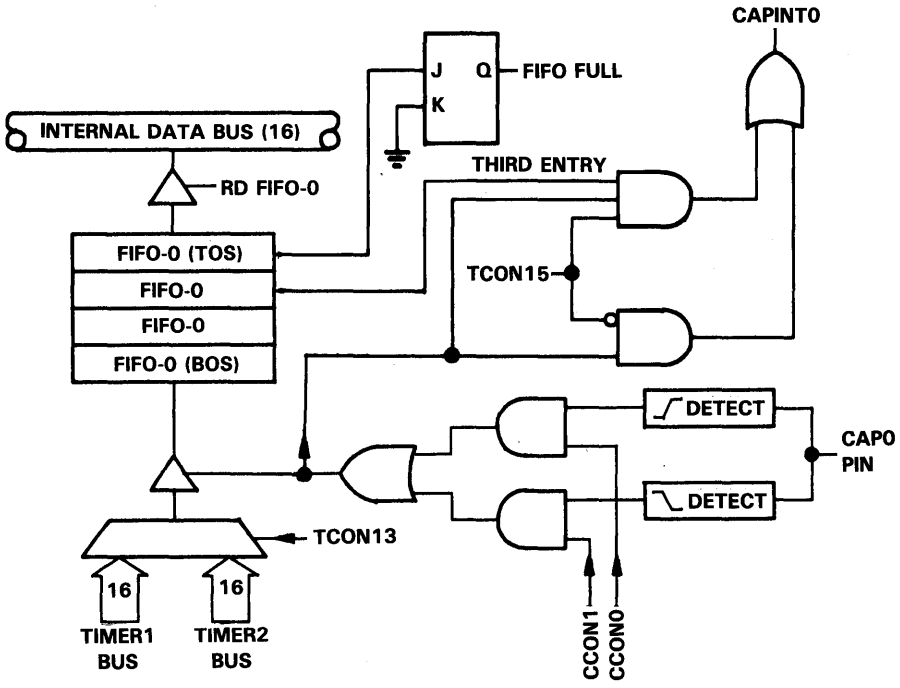


Figure 3-27. Capture Module

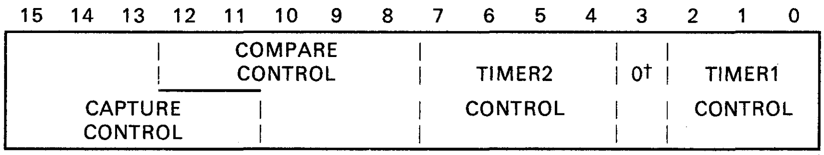
Any time a positive or negative transition is detected on a capture input pin, the current value of a timer is saved in a FIFO. In addition, when a capture is made, each FIFO can generate an interrupt to the CPU. There are four interrupts (CAPINT0 through CAPINT3) dedicated to the capture subsystem.

3.7.2.1 *FIFO Stacks*

Each of the FIFOs is four deep and dedicated to one of the capture input pins. When a transition is detected on a capture input pin, the current value of the timer is saved in the corresponding FIFO. Either TMR1 or TMR2 may be specified. Each FIFO has a dedicated interrupt (CAPINTx). The FIFO can generate an interrupt when the first capture is made, or when the third capture entry is received. The CPU has direct access only to the top of each FIFO. To read the rest of the FIFO entries, successive read must be made. When a FIFO is empty (i.e., all entries are read), a status bit (FIFOx NOT EMPTY) is cleared in CCON. When a FIFO is full and an additional capture is made, the overrun condition is indicated by setting a status bit (FIFOx OVERRUN) in CCON. At the same time, the FIFO preserves existing entries and ignores all additional entries thus losing them. The FIFO registers have a bank address of 6h, and a port address of 0h through 3h.

3.7.2.2 *TCON Register*

Bits 11 through 15 of the timer control (TCON) register control the operation of the capture subsystem. Figure 3-28 and Table 3-16 show the functions of the TCON register bits associated with the capture subsystem.



NOTE: † Reserved bit. Should be cleared to 0.

Figure 3-28. TCON Register Capture Bit Configuration

Table 3-16. TCON Capture Register Description

BIT #	DESCRIPTION
15	Interrupt on first or third entry in FIFOx. 0 = An interrupt is generated to the CPU on the first capture entry received in an empty FIFO. 1 = An interrupt is generated when the third capture entry is received in the FIFO, and the FIFO already has two entries.
14	Capture Enable. 0 = Enables Capture subsystem. Starts normal operation of capture subsystem. 1 = Disables the capture subsystem. No capture possible on any pin. All FIFO registers (FIFOx) are reset to 0000h.All FIFOx NOT EMPTY bits reset to 0. All FIFOx OVERRUN bits reset to 0.
13	Timer Select for Capture Subsystem. 0 = Timer 1. Selects Timer 1 for use in capture. 1 = Timer 2. Selects Timer 2 for use in capture.
12	CMP5/CAP3 Configure. 0 = Configures pin CMP5/CAP3 as capture input. 1 = Configures pin CMP5/CAP3 as compare output.
11	CMP4/CAP2 Configure. 0 = Configures pin CMP4/CAP2 as capture input. 1 = Configures pin CMP4/CAP2 as compare output.

NOTE: All bits cleared to zero on reset.

3.7.2.3 Capture Control Register

The Capture Control register (CCON), refer to Figure 3-29 and Table 3-17, is used to configure and control the operation of individual capture inputs. Each capture input has four bits in CCON that controls its operation. Capture can be enabled or disabled on each individual capture pin. In addition, each pin can be programmed to detect: a transition, a falling edge, or a rising edge. If either CMP4/CAP2 or CMP5/CAP3 pins are configured as output compare pins, it is still possible to enable a capture function on these pins. In this case the transition detected will be due to the compare output function, not an external input.

The CCON gives the status of each of the FIFO's, whether they are empty or have an overflow. When an overflow is detected, and bit FIFOx OVERRUN is set to 1, CCON has to be cleared to 0. The user does this in software after a read from FIFOx is done. Some bits of the CCON i.e. FIFO status bits, can only be read while the rest of the bits can be written to and read from.

Another 16-bit register - Capture Clear (CCLR) is also used by the capture subsystem. The CCLR allows clearing of individual bits in CCON without affecting other bits. The operation of the CCLR is similar to the BCLR register. There is a direct correlation between bits of CCLR and CCON,i.e. Setting bit 3 of CCLR to 1 will clear bit 3 of CCON. Figure 3-29 illustrates bit 15 of CCLR clearing bit 15 of CCON. The CCLR makes it easier to configure the CCON since it is not possible to write to the CCON directly. The CCON has a bank address of 6h and port address 4h. The CCLR has bank address 6h and port address 5h, and is described in Table 3-17.

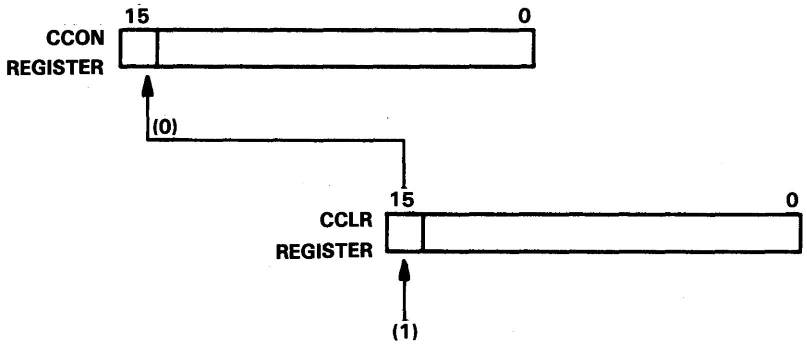


Figure 3-29. CCON and CCLR Registers

Table 3-17. CCON Register Description

BIT #	FUNCTION	DESCRIPTION
15	FIFO-3 overrun error.	0 = FIFO-3 not full. 1 = FIFO-3 is full, and another transition or capture entry is detected. This bit must be cleared by the user, if a read is done on FIFO-3. Status bit. Read and clear access.
14	FIFO-3 not empty.	0 = No entries available in FIFO-3. 1 = FIFO-3 has one or more unread entries. Cleared as soon as CPU has read all entries and FIFO-3 is empty. Status bit. Read access only.
13,12	Edge transition detection for CAP3	00 = Capture disabled on pin CAP3. 01 = Detect positive (rising) edge on pin CAP3. 10 = Detect negative (falling) edge on pin CAP3 11 = Detect any transition on pin CAP3. Control bits. Read and write access.
11	FIFO-2 overrun error.	0 = FIFO-2 not full. 1 = FIFO-2 is full, and another transition or capture entry is detected. This bit must be cleared by the user if a read is done on FIFO-2. Status bit. Read and clear access.
10	FIFO-2 not empty.	0 = No entries available in FIFO-2. 1 = FIFO-0 has one or more unread entries. Cleared as soon as CPU has read all entries and FIFO-2 is empty. Status bit. Read access only.
9,8	Edge transition detection for CAP2	00 = Capture disabled on pin CAP2. 01 = Detect positive (rising) edge on pin CAP2 10 = Detect negative (falling) edge on pin CAP2. 11 = Detect any transition on pin CAP2. Control bits. Read and write access.
7	FIFO-1 overrun error.	0 = FIFO-1 not full. 1 = FIFO-1 is full, and another transition or capture entry is detected. This bit must be cleared by the user if a read is done on FIFO-1. Status bit. Read and clear access.
6	FIFO-1 not empty.	0 = No entries available in FIFO-1. 1 = FIFO-1 has one or more unread entries. Cleared as soon as CPU has read all entries, and FIFO-1 is empty. Status bit. Read access only.
5,4	Edge transition detection for CAP1	00 = Capture disabled on pin CAP1. 01 = Detect positive (rising) edge on pin CAP1. 10 = Detect negative (falling) edge on pin CAP1 11 = Detect any transition on pin CAP1. Control bits. Read and write access.

NOTE: All bits cleared to zero on reset.

Table 3-17. CCON Register Description (Concluded)

BIT #	FUNCTION	DESCRIPTION
3	FIFO-0 overrun error	0 = FIFO-0 not full. 1 = FIFO-0 is full, and another transition or capture entry is detected. This bit must be cleared by the user if a read is done on FIFO-0. Status bit. Read and clear access.
2	FIFO-0 not empty.	0 = No entries available in FIFO-0. 1 = FIFO-0 has one or more unread entries. Cleared as soon as CPU has read all entries and FIFO-0 is empty. Status bit. Read access only.
1,0	Edge transition detection for CAPO	00 = Capture disabled on pin CAPO. 01 = Detect positive (rising) edge on pin CAPO. 10 = Detect negative (falling) edge on pin CAPO 11 = Detect any transition on pin CAPO. Control bits. Read and write access

NOTE: All bits cleared to zero on reset.

3.7.2.4 Capture Interrupts

Each capture input has a dedicated interrupt (CAPINTx), that can be generated either when the corresponding FIFO receives its first or third entry. This selection is done by means of bit 15 of TCON. CAPINT0 through CAPINT3 set bits 8 through bit 11 of IF. These interrupts can be masked by setting the appropriate bit in IM.

3.8 Serial Port

The TMS320C14/E14 has a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) type serial port that supports industry-standard communications protocols. Key features of the serial port include the following:

- Double-buffered receiver/transmitter
- Full-duplex asynchronous mode with 400 Kbps maximum transceiving rate (25.6 MHz clock)
- Half-duplex synchronous mode with master/slave option at maximum transceiving rate of 6.4 Mbps(25.6 MHz clock)
- Full-duplex codec-compatible mode with maximum transceiving rate of 1.6 MHZ (25.6 Mhz clock)
- Supports address detect and address match protocols
- Separate interrupts for receiver/transmitter
- Separate 16-bit timer for baud rate generation
- Programmable operation through 16-bit control/status register

The operational architecture of the serial port is dependent on the mode it is operating in. The asynchronous (async) mode is full-duplex with a maximum transmission/reception rate of CLKOUT/16. The async mode uses start and stop bits for synchronization at the byte level. The synchronous (sync) mode is half-duplex, with a maximum transceiving rate of CLKOUT. In the synchronous mode, the serial port can operate in either master or slave configuration, accepting an internal or external clock. The codec mode is full-duplex and compatible with industry standard codecs such as TCM29C13, etc. Maximum transceiving rate in the codec mode is CLKOUT/4. Either internal or external frame sync pulses can be used in this mode. Double-buffering of receive/transmit data is used in all modes.

The serial port supports two communication protocols. The first communication protocol supports 8051 type 9-bit communication, where the ninth bit indicates address or data reception/transmission. The second communication protocol supports a sleep/wakeup mode, in which all nine bits have to match the device address stored in a register (SMAT) in order to start data reception. These communication protocols are available in all three modes: asynchronous, synchronous, and codec. Table 3-18 summarizes the registers associated with the serial port.

Table 3-18. Serial Port Register Summary

REGISTER	ADDR	BANK	DESCRIPTION
SCON	0	5	Serial port control register. controls configuration and operation of serial port.
SSET	1	5	SCON bit set register. allows setting of individual bits in SCON without affecting other bits.
SCLR	2	5	SCON bit clear register. Allows clearing of individual bits without affecting other bits.
TBR	3	5	Transmit buffer register. Stores temporary data while old data is being shifted out from transmit register.
RBR	4	5	Receive buffer register. Stores temporary data while new data is being shifted into receive register.
SBRG	5	5	Serial Port Baud rate generator. Sets the divide ratios for the serial port timer to generate baud rates for synchronous mode.
SMAT	2	7	Serial Port Match word register. The serial port stays in the sleep mode until an address match with value in SMAT register is detected.
TSR	3	7	Transmit shift register. Stores outgoing data that is currently being transmitted.
RSR	4	7	Receive shift register. Stores incoming data that is currently being received.
STMR/TMR3	5	7	Timer 3. Free running timer that is used for baud rate generation for the serial port

3.8.1 Serial Control Register

The Serial Control (SCON) register (see Table 3-19) is a 16-bit register that controls operation of the serial port. At the same time, the SCON also acts as a status register to indicate the status of the serial port. Async, sync, or codec mode is selected by configuring bits 0 and 14 of the SCON. Reception on the serial port can be disabled by bit 11 of the SCON. Bits 5 and 6 select the number of data bits. Bits 9 and 10 select the communication protocol. Bits 3, 4, 7, and 8 indicate errors or overflows. The SCON should not be modified during reception or transmission, or unpredictable results may occur. The complete configuration of the SCON is described in the following section. The SCON has a a bank address of 5h and a port address of 0h.

Table 3-19. SCON Register Description

BIT #	FUNCTION	DESCRIPTION
0	Sync mode select (SYNC)	0 = Disables Synchronous mode and selects asynchronous mode. Note that in addition to configuring this bit, bit 14 also has to be cleared to 0 to select async/sync mode. 1 = Enables synchronous mode. Control bit. Write and read access. Set to 1 on reset.
1	Parity enable enable (PEN)	0 = Disable parity. 1= Enables parity. Parity is added during transmission, and detected during reception. Control bit. Write and read access. Cleared to 0 on reset.
2	Even/Odd parity select (EOP)	0 = Selects even parity. 1 = Selects odd parity. Control bit. Write and read access. Cleared to 0 on reset.
3	Parity error detect	0 = Normal operation. No parity error detected. 1 = Parity error has been detected during reception. Must be cleared in software by user. Status bit. Read and clear access only. Cleared to 0 on reset.
4	Framing error detect (FERR)	0 = Normal operation. No framing error detected. 1 = Framing error has been detected, i.e. invalid stop bit received. Must be cleared in software by user. Status bit. Read or clear access. Cleared to 0 on reset.
6,5	Data bits select (DN0,DN1)	00 = Selects 6 data bits. 01 = Selects 7 data bits. 10 = Selects 8 data bits. 11 = Selects 9 data bits. Control bits. Write and read access. Set to 10 on reset.
7	Receiver overflow error (ROV)	0 = Normal operation. No overflow detected. 1 = Receiver buffer overflow detected. Both receive registers (RSR and RBR) are full, and third data reception detected. Must be cleared in software by user. Status bit. Read and clear access. Cleared to 0 on reset.
8	Receive buffers full (BBF)	0 = No framing error detected. 1 = Receive buffers full detected. Both receive registers/buffers RSR, RBR have just filled. New data word is fully received in RSR. Status bit. Read and clear access. Cleared to 0 on reset.

Table 3-19. SCON Register Description (Continued)

BIT #	FUNCTION	DESCRIPTION
9	Receive qualifier1 (RINTQ1)	0 = Protocol 1 disabled. Normal reception mode. Receive interrupt, RXINT, generated to CPU every time new data frame is clocked into RSR register. 1 = Interrupt CPU if MSB of receive word is 1. Protocol 1 is enabled and normal reception mode is disabled. Serial port is put in sleep mode. Receive interrupt (RXTINT) is generated if MSB or last bit of received word is 1 (indicating address reception). Control bit. Write and read access. Cleared to 0 on reset.
10	Receive interrupt qualifier2 (RINTQ2)	0 = Protocol 2 disabled. Normal reception mode. Receive interrupt (RXINT) generated to CPU every time new data frame is clocked into RSR. 1 = Interrupt CPU if receive word matches SMAT. Protocol 2 is enabled and normal reception mode is disabled. Serial port is put in sleep mode. Receive interrupt (RXTINT) is generated if received word matches value in SMAT (indicates serial port has been addressed). Control bit. Write and read access. Cleared to 0 on reset.
11	Continuous receive enable. (CREN)	0 = Disables reception in asynchronous, synchronous slave and codec modes. Disables continous reception in synchronous "master" mode. 1 = Enables reception/continuous reception. In the async, codec, and synch "slave" modes this enables reception. In the sync "master" mode, it enables continuous reception. Control bit. Write and read access. Set to 1 on reset.
12	Single receive enable (SREN)/ Frame Synch pulse width (FSPW)	0 = Disables reception in sync "master" mode. Can be over-ridden by CREN bit. In codec mode, sets frame sync pulse generated by TMS320C14/E14 at one clock pulse wide. 1 = Allows single word reception in sync "master" mode. In the sync "master" mode,the receiver will receive exactly one word, then reset this bit to 0 to disable further reception. In the codec mode, sets frame sync pulse generated by the TMS320C14/E14 at two clock pulses wide Control bit. Write and read access. Cleared to 0 on reset.

Table 3-19. SCON Register Description (Concluded)

BIT #	FUNCTION	DESCRIPTION
13	Clock source (CSRC)/ Frame sync pulse source	0 = Internal clock/"master" selected in sync mode. Selects "master" in sync mode. Internal clock generator is used as clock source and output on TXD/CLKpin. In codec mode, internal clock generator is used to generate frame sync pulses. In addition to resetting this bit to zero, bits 11 and 12 of TCON must be set to 1 to configure FSX and FSR pins as outputs. 1 = External clock/slave in synchronous mode. External clock source is selected in sync mode and accepted on TXD/CLK pin for both receive and transmit. This configures the serial port as "slave" in sync mode. In the codec mode, external frame sync pulses are selected. In addition to setting this bit to 1, bits 11 and 12 of TCON register must be set to 0 to configure FSX and FSR pins as outputs. In addition, bits 8 and 12 of CCON must be set to 1 to enable positive edge detection on FSR and FSX pins. Control bit. Write and read access. Cleared to 0 on reset.
14	Codec mode (CODEC)	0 = Disables codec mode. Bit 0 of SCON configures serial port as synchronous or asynchronous. 1 = Selects codec mode. Configures serial port in codec codec mode and overrides bit 0 of SCON. Control bit. Write and read access. Cleared to 0 upon reset.
15	Transmit Buffer empty. (TBE)	0 = Transmit buffer not empty. Transmit buffer contains data data that has not been transferred to transmit shift register. 1 = Transmit buffer empty. Transmit buffer is empty and ready to accept new data from the CPU. Status bit. Read access only. Set to 1 on reset.

To facilitate clearing and setting of individual bits in the SCON, two additional registers (SCLR and SSET) are also provided (see Figure Figure 3-30). SCLR and SSET function similarly to BCLR and BSET. They clear or set individual bits in the SCON. A mask of ones written to SCLR will clear the corresponding bits in the SCON. Bit positions in SCLR containing zeroes will not affect corresponding bits in the SCON. Similarly, SSET will set to one, all bits in the SCON that have corresponding ones in SSET. Bit positions having zeroes will not affect corresponding bits in the SCON. The SCLR and SSET have a bank address of 5h and port addresses of 1h and 2h.

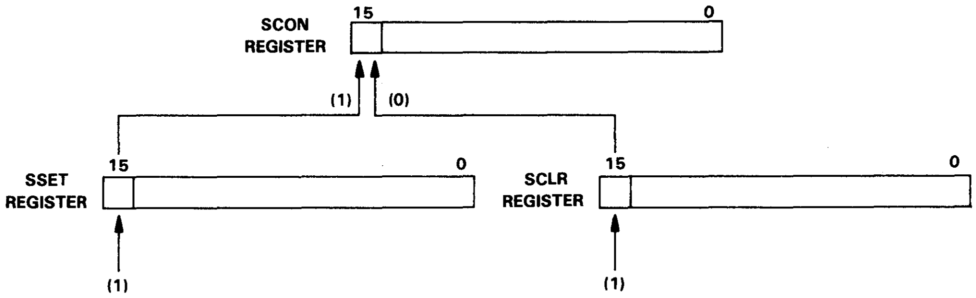


Figure 3-30. SCON Register Control

Table 3-20 shows the basic configuration of the serial port after power-up or a reset.

Table 3-20. Serial Port Key Default Settings

PARAMETER	DESCRIPTION
Mode	Synchronous
Parity	None
Word Length	8 Data bits
Reception	Continuous
Clock	Internal

3.8.2 Serial Port Baud Rate Generator

The serial port baud rate generator provides internal baud rate generation for the asynchronous and synchronous modes. It is not required for the codec mode, or for synchronous 'slave' operation. If the baud rate generator is not required for clock/baud rate generation, it can be used as a general-purpose timer (refer to Section 3.5.3) with it's own interrupt. When used for baud rate/clock generation, bit 12 of the interrupt mask (IM) register should be masked.

The baud rate generator has a 16-bit register (STMR), refer to Figure 3-31, and baud rate generation register (SBRG), and uses CLKOUT as the input. The SBRG register holds a user-installed value that is used in generating the baud rate.

3.8.2.1 *Asynchronous Baud Rate Generation*

The asynchronous baud rate is computed as follows:

$$\text{Baud Rate} = \frac{\text{CLKOUT frequency}}{16 \times (K + 1)}$$

K = value stored in the SBRG register

CLKOUT = CLKIN / 4

The maximum transmission/reception rate is CLKOUT/16, or 400 KHz at 25.6 MHz CLKIN frequency. The following table gives the K values that have to be stored in the SBRG register to obtain the standard baud rates and the deviations from those rates. These calculations use the maximum oscillator frequency of 25.6 MHz.

Table 3-21. SBRG Value For Standard Baud Rates

Desired Baud Rate	SBRG Value (K)	Actual Baud Rate
19.2 Kbps	0014h	19.048 Kbps
9600 bps	0028h	9756.00 bps
4800 bps	0052h	4819.00 bps
2400 bps	00A6h	2395.20 bps
1200 bps	014Ch	1201.20 bps
300 bps	0534h	300.07 bps
110 bps	0E33h	110.011 bps

NOTE: All K values in hex.

The maximum baud rate is obtained when K = 0. The minimum baud rate is generated when the value 65535 (FFFFh) is stored in the SBRG register. The percentage deviation (Pcd) is computed as follows:

Pcd = (Actual rate - desired rate) / desired rate

3.8.2.2 *Synchronous Baud Rate Generation*

In the synchronous mode, the rate of the transmitted clock signal is computed by the following formula:

$$\text{Baud Rate} = \frac{\text{CLKOUT frequency}}{K + 1}$$

K = value stored in SBRG register

CLKOUT = CLKIN / 4

3.8.3 Asynchronous Mode

The asynchronous mode is selected by clearing bits 0 and 14 of the SCON to 0. This is a full-duplex mode, with data being transmitted on the TXD/CLK pin, and data received on the RXD/DATA pin. Figure 3-31 shows the architecture of the serial port for asynchronous communication.

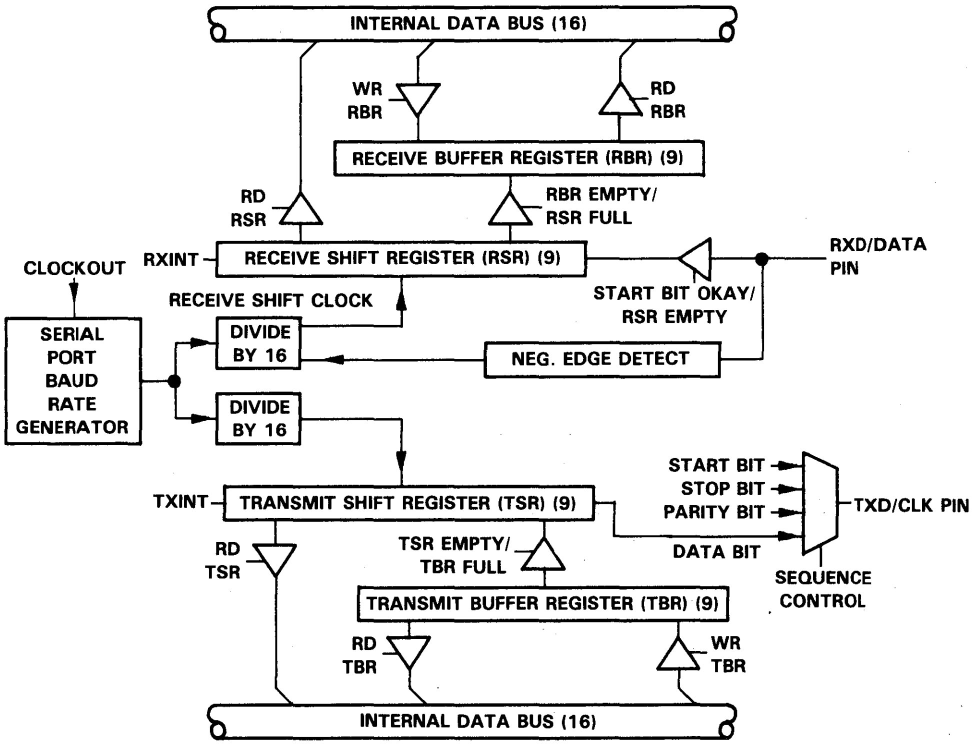


Figure 3-31. Serial Port in Asynchronous Operation

In the asynchronous mode, data is transmitted on a character-by- character basis, with each data frame containing a start bit, 6-9 data bits, a parity bit (if parity is enabled), and a stop bit. Both the transmit and receive sections are double-buffered to allow continuous transceiving. The serial port timer provides the transmit and receive register clock signals, as well as the baud rate.

3.8.3.1 Asynchronous Transmission

The transmit section consists of a 9-bit Transmit Shift register (TSR) and a 9-bit Transmit Buffer register (TBR). Data is always written to the TBR, and then transferred to the TSR. Data to the TBR should be written in right-justified form, irrespective of the number of the bits to be transmitted. It is possible to directly read the TSR. It is not possible to write to the TSR directly. Data from the TSR is shifted out on pin TXD/CLK. The TXD/CLK pin is held at 1 while the serial port is not transmitting. Figure 3-32 shows an asynchronous 8-bit data for transmission with parity.

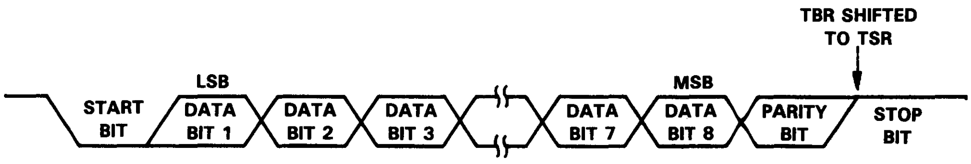


Figure 3-32. Asynchronous Transmission of Eight Bits Plus Parity

Transmission in the serial port is started by writing data to the TBR. If the TSR is empty, data from the TBR is transferred to the TSR. If the TSR is full, then data is kept in the TBR, and existing data in the TSR is shifted out to the Sequence Control logic. At the same time, bit 15 in the SCON register is cleared to 0 to indicate the TBR is not empty. If both the TSR and TBR are full and the CPU tries to write to the TBR, the write is not allowed to take place, and existing data in both registers are maintained. The Sequence Control logic constructs the transmit frame by outputing a start bit followed by the data bits from the TSR, and a parity bit is then added (if required). As soon as the last data or parity bit has left the TSR, TXINT is generated to the CPU, indicating a frame has been sent.

The TSR has a bank address of 7h and a port address of 3h. The TBR has a bank address of 5h and a port address of 3h.

Summary of Asynchronous Mode Transmission:

1) Asynchronous mode is enabled by clearing bit 0 of SCON to 0, and setting bit 11 to 1.

2) Transmission begins by writing data to TBR. If TSR is empty, this data is transferred to TSR.

3) Start bit is transmitted first, followed by 6 to 9 data bits (LSB through MSB).

4) If parity is enabled, parity bit is added after MSB.

5) Stop bit is shifted out and TXINT is generated, indicating end of transmission.

3.8.3.2 Asynchronous Reception

The receive section includes two 9-bit registers: the Receive Buffer register (RBR) and Receive Shift register (RSR). Data is received on the RXD/DATA pin and the Negative Edge Detect logic samples the input for a start bit on the 7th, 8th, and 9th sampling pulse following a detected falling edge (Figure 3-33).

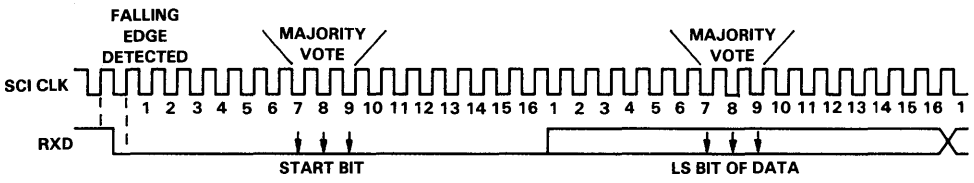


Figure 3-33. Start Bit Detection

A majority vote of 0 by the three sample pulses results in the data being loaded into the RSR. If the majority vote is 1, the data is not accepted, and the sampling continues until a start bit is detected.

After the appropriate number of data bits are received (i.e. 6-9 bits), parity is computed on the data bits (if parity is enabled) and compared with the received parity bit. If the computed parity does not match with the received parity, a parity error is indicated by setting a flag (i.e. bit 3 of SCON register-PERR), to a 1. Normal reception will continue. The user is responsible for checking this bit and clearing it. PERR (parity error) is also double buffered. This means when both buffers are full and PERR is raised, the data in RBR has a parity error. If data in RSR also has a parity error, PERR will be raised again when data is transferred from RSR to RBR.

After the parity bit is received, a stop bit is received, indicating the end of that block. If a stop bit is not received, a framing error is indicated by the setting of bit 4 (FERR) in SCON to a 1. Normal reception will continue, and the receiver will look for the next start bit. The user is responsible for checking this bit and clearing it. Data is then transferred to the RBR and interrupt RXINT is routed to the CPU (Figure 3-34). The RSR is now available to receive another data block, and the Negative Edge Detect logic is activated again.

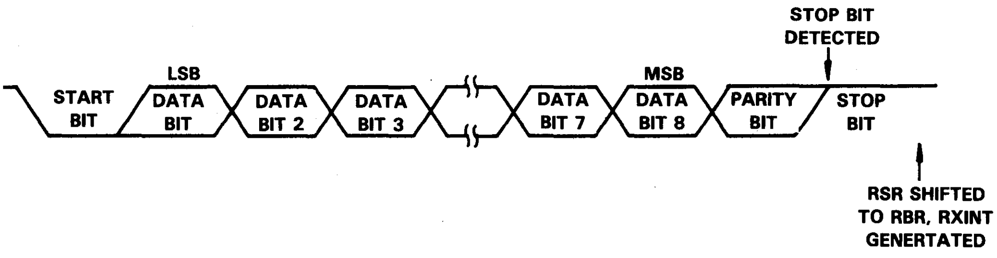


Figure 3-34. Asychronous Reception of Eight Bits Plus Parity

If RBR is not empty when new data has just been received in the RSR, an error flag, BBF (both buffers full) is set in SCON. In this case, data from RSR will not be transferred to RBR and all further reception will be disabled. Existing data will be maintained in both RSR and RBR until the user reads RBR. Reading RBR will automatically clear the BBF flag, and data in the RSR will be transferred into RBR, thus allowing further reception. RSR has a bank address of 7h, and a port address of 4h. RBR has bank address of 5h, and a port address of 4h.

Summary of Asynchronous Mode Reception:

1) A negative edge is received to indicate a start bit. A test is done to indicate whether start bit is valid or not. If invalid, start bit reception is discontinued.

2) If start bit is valid, appropriate number of data bits (as specified by the SCON register) are shifted into RSR.

3) (Optional). Parity is computed on the data bits, and a parity bit is received. Computed parity is compared with received parity. If different, parity error is indicated.

4) A stop bit is received to indicate end of reception. If stop bit is not received, framing error is indicated.

5) An interrupt is generated to the CPU.

6) Data is transferred from RSR to RBR. If RBR is full, a flag is set (both buffers full-BBF).

7) Reception is complete, receiver waits for another negative transition.

3.8.4 Synchronous Mode

The synchronous (sync) mode is selected by setting bit 0 of SCON to 1, and clearing bit 14 of SCON to 0. The Sync mode is half duplex, with one line used for transceiving data, and another line for the clock signal. The clock signal, (provided internally or externally) is used for synchronization between the transmitter and receiver instead of start and stop bits. The master or slave mode is selected by bit 13 (CSRC) of SCON. Figure 3-35 shows the architecture for the serial port in synchronous communication.

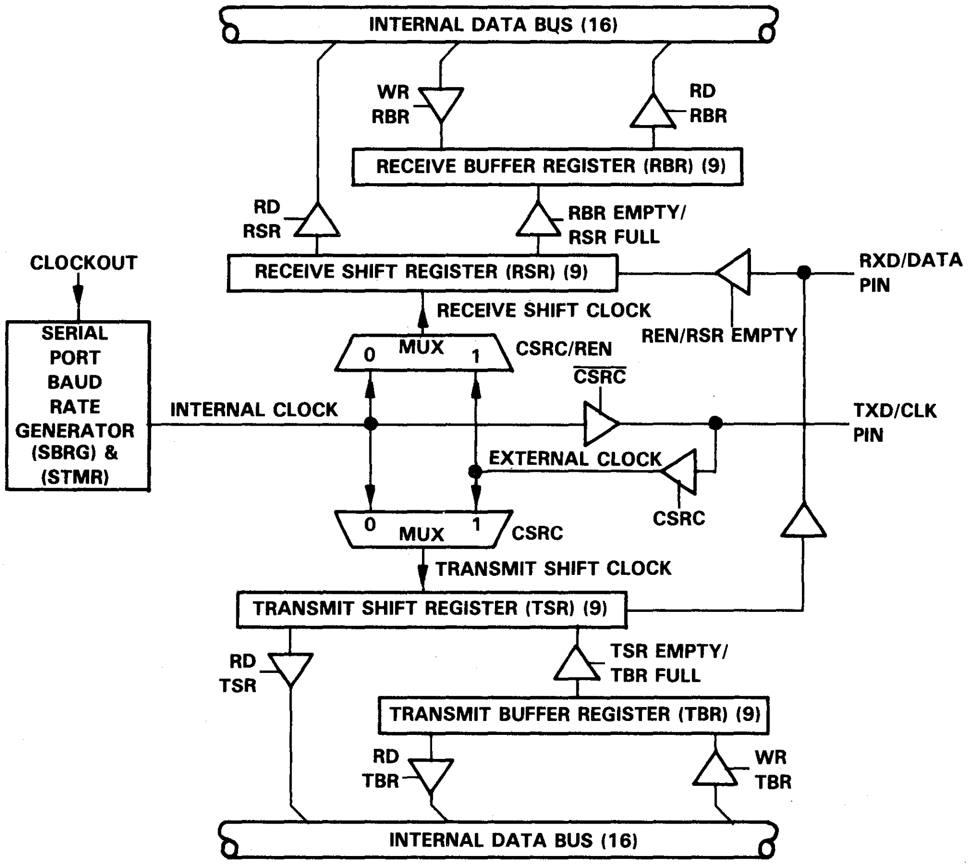


Figure 3-35. Serial Port in Synchronous Operation

Basically, the architecture of the serial port for the sync mode is the same as for the async mode. Both transmit and receive sections are doubled-buffered to allow continuous transceiving. The data, however, is transmitted and received on the same (RXD/DATA) pin. The TXD/CLK pin is used for either transmitting or receiving the clock signal. The maximum allowable clock rate is 6.4 Mhz when internal clock is selected. The synchronous data block can contain from 6-9 bits with an optional parity bit. Start and stop bits are not used. The number of data bits and whether or not parity is enabled are options controlled by the SCON register.

3.8.4.1 *Master/Slave Operation*

The synchronous mode requires that the serial port be configured as either a master or slave. Configured as the master, the serial port's baud rate generator provides the clock signal for transmission and and reception.

In slave operation the clock signal is provided externally from the external device. Master/slave operation is controlled by bit 13 of the SCON register.

3.8.4.2 *Synchronous Transmission*

The transmit section is double-buffered and consists of two 9-bit registers: transmit shift register (TSR) and transmit buffer register (TSR). Data is always loaded into TBR in right-justified form, and shifted out from TSR on the RXD/DATA pin. This pin is held at logic 1 when no transmission is taking place. If master mode or internal clock is selected, the TXD/CLK pin is held at logic one when no transmission is taking place.

Before synchronous transmission can begin, it must be enabled by resetting bits 11 and 12 of the SCON register to 0. Synchronous transmission begins when the CPU writes data into the TBR. In slave operation, this should occur before the arrival of the first rising edge of external clock. This is to prevent the receiving device from reading an erroneous 1, since the RXD/DATA pin is held at logic 1 during no transmission. If the TSR is empty, the data in the TBR is transferred immediately to TSR. If TSR is full, then this data is retained in TBR and the existing data from TSR is shifted out (Figure 3-36 and Figure 3-37). If both TBR and TSR are full and the CPU tries to write data to TBR, the write is not allowed to take place, and existing data in both registers are maintained.

Transmission is controlled by the presence of the clock signal on the CLK pin. Single or continuous data transmission is possible as shown in Figure 3-36 and Figure 3-37 respectively depending on the termination or continuation of the clock signal.

With internal clock, a continuous update of TBR on each TXINT will cause the port to operate in continuous fashion. If the TBR is not updated, data transmit and clock will stop after the last bit in TSR is output.

With external clock, a continuous clock source will cause the TBR to transfer to the TSR at the end of each data word. The same data word will continue to be transmitted if the TBR is not updated.

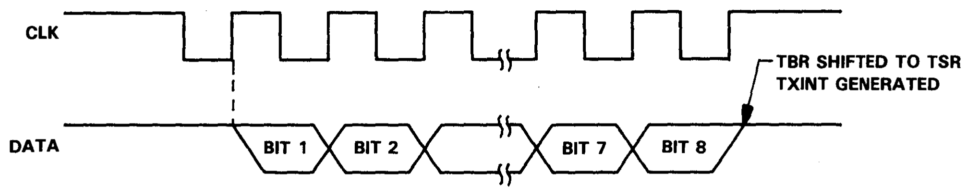


Figure 3-36. Master Mode Single Synchronous Transmission with 8 Data Bits, No Parity

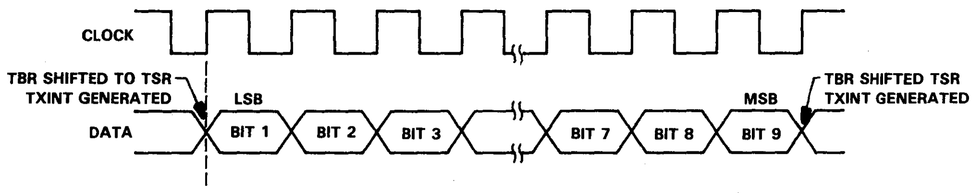


Figure 3-37. Synchronous Continuous Transmission with 9 Data Bits, No Parity

Data is shifted out from the TSR on the rising edge of internal or external clock and transmitted on the RXD/DATA pin. At this time, maskable interrupt TXINT is generated to the CPU. Six to nine data bits are transmitted with an optional parity bit, and without start and stop bits.

Summary of transmission in Synchronous Mode:

1) Synchronous mode is enabled by setting bit 0 of SCON to 1, and bit 14 of SCON to 0.

2) Master mode is selected by selecting internal clock, and slave mode is selected by selecting external clock.

3) Transmission is enabled by resetting bit 11 and 12 of SCON register to 0.

4) Transmission begins by writing data to TBR. If TSR is empty, this data is transferred to TSR.

5) Data is shifted out on next rising edge of internal/external clock, LSB first.

6) 6 to 9 bits of data are transmitted with an optional parity bit.

7) An interrupt (TXINT) is routed to the CPU when the last bit is shifted out.

8) If a write to TBR has been done, i.e. it contains new data, it is transferred to TSR, and new transmission begins.

3.8.4.3 Synchronous Reception

The receive section is double buffered and consists of two 9-bit registers: Receive Shift Register (RSR) and Receive Buffer Register (RBR). Data is clocked into RSR on the RXD/DATA pin, and always read from RBR in right-justified form. If data is less than 9 bits, the upper bits are read as zeroes.

Reception is enabled by setting either of the two receive enable bits (bit 11-CREN) or bit 12-SREN of SCON to 1. CREN allows continous reception (refer to Figure 3-38), while SREN allows a single reception only (refer to Figure 3-39). After reception is enabled, data is sampled on the falling edge of the internally generated clock. Data is shifted in on the RXD/DATA pin into RSR. 6 to 9 data bits are accepted with an optional parity bit. When the complete word is received, the contents of RSR are transferred to RBR and an RXINT to CPU is generated (refer to Figure 3-38 and Figure 3-39).

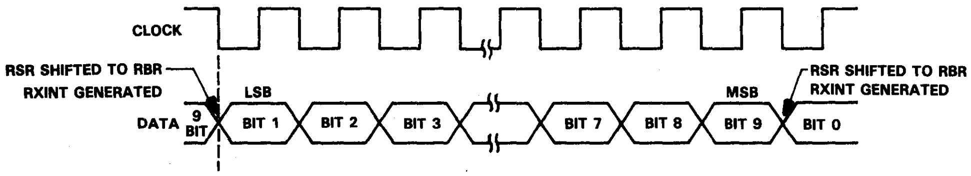


Figure 3-38. Synchronous Reception with 9 Data Bits, No Parity

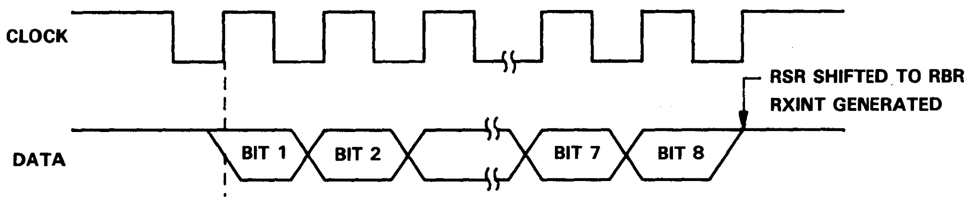


Figure 3-39. Master Mode Single Synchronous Reception with 8 Data Bits, No Parity

If the Receive Buffer register (RBR) is full and another word is received in the RSR register, the BBF (both buffers full) bit is set in the SCON. In this case, contents of RSR will not be transferred into RBR, and both registers will maintain existing data. However, no further reception will be allowed, and all new data coming into the RXD/DATA pin will be ignored. To allow further reception, the RBR must be read by the user. This will automatically clear BBF, and contents of RSR will be transferred into RBR. This will then allow RSR to receive new data.

If additional data is detected on the RXD/DATA pin when both buffers are full (i.e. BBF is 1), ROV (receive overflow flag bit-7), of SCON is set. This indicates that incoming data has been lost. The user must clear this by software. If the received parity does not match the computed parity on the received word, PERR (parity error bit-3), in SCON is set to 1. Again, the user must clear this by software. PERR is also double-buffered. When both buffers are full and PERR flag is raised, the data in RBR has a parity error. If data in RSR also has parity errors, then PERR will be set again when that data is transferred from RSR to RBR.

Master Receive. In master mode, it is possible to have either continous reception or receive only a single word. If bit 12 (SREN) of SCON is set to one, only a single word is received. After receiving the single word, the hardware disables further reception by automatically clearing SREN to 0, and also disables the clock in the master receive mode. Reception will remain disabled until the user enables reception. SREN can also be reset to 0 in the middle of a reception to abort that reception. If continous reception is required, bit 11 (CREN) of SCON register is set to 1. If both SREN and CREN bits are set to 1 (both continous and single reception enabled) CREN takes precedence over SREN and continous reception occurs. In continous reception mode, data words are received continuously without any break. Maskable interrupt RXINT is generated after receiving each data word. CREN is never reset by hardware. The user has to reset it to disable reception.

Slave Receive. In slave mode, reception is enabled only by setting CREN (bit 11) of SCON to 1. The SREN bit has no affect in slave mode. After enabling, receive data is sampled on the falling edge of external clock. Data is shifted in on the RXD/DATA pin into RSR. When the complete block (6-9 bits) is received, the contents of the RSR are transferred to the RBR and interrupt RXINT is routed to the CPU.

Summary of Reception in Synchronous mode

1) Synchronous mode is enabled by setting bit 0 of SCON to 1, and bit 14 of SCON to 0.
2) Master or slave mode is enabled by selecting internal or external clock with bit 13 of SCON.
3) Reception is enabled by setting bit 11 of SCON to 1 in slave mode, and bit 11 or bit 12 of SCON in master mode.
4) Reception starts on next falling edge of internal/external clock, LSB first.
5) 6 to 9 data bits are received with an optional parity bit.
6) RXINT is generated to the CPU, and computed parity on received data is compared against received parity. If different, PERR is set in SCON.
7) Received word is transferred from RSR to RBR. If RBR contains data, BBF (both buffers full) is set in SCON.
8) If transfer to RBR is successful, and continous reception is enabled, RSR is ready to receive new data. Otherwise reception is stopped until RBR is read and/or reception is enabled.

3.8.5 Codec Mode

Codec (COmpanding and DECompanding) mode is selected by setting bit 14 of SCON to a 1. Bit 0 of SCON is ignored if bit 14 is set to 1. The codec mode is full-duplex and allows a direct interface with industry standard codecs like the TCM29C13. Codec supports the communication industry standard protocol of PCM (pulse code modulation). In the codec mode (see Figure 3-40), data is transmitted on the TXD/CLK pin and received on the RXD/DATA pin. Codec mode uses external clock for both transmitting and receiving, so the serial port baud rate generator is not required for baud rate generation. The transmit clock (CLKX), is accepted on pin TCLK2/CLKX, while the receive clock (CLKR), is accepted on pin TCLK1/CLKR.

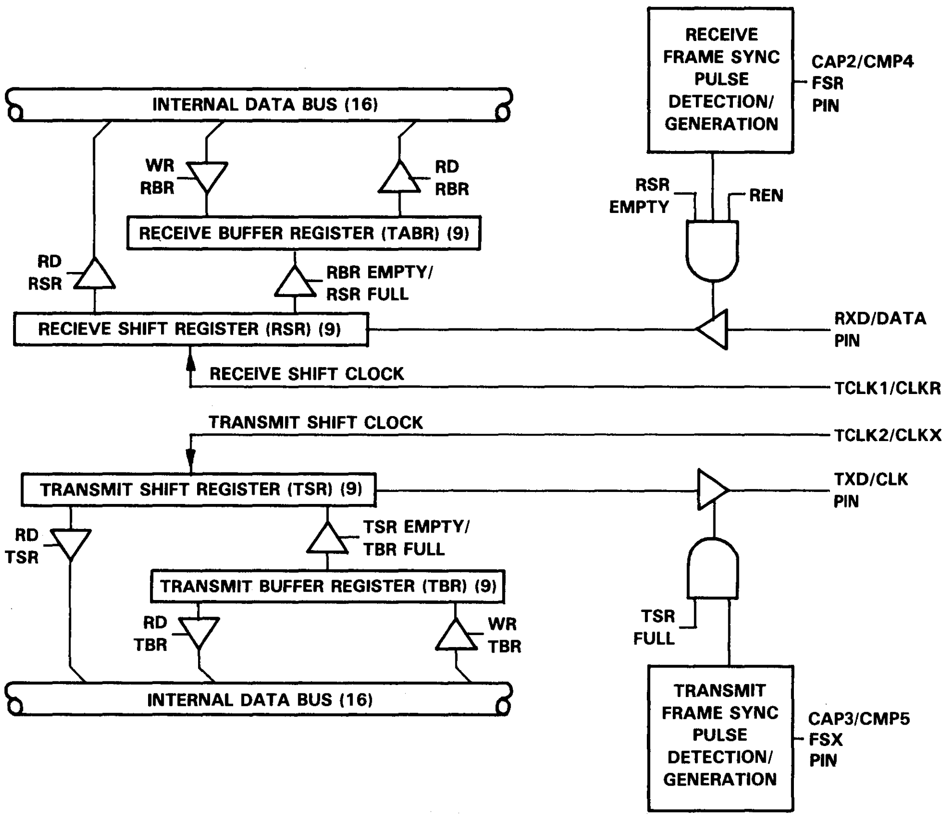


Figure 3-40. Serial Port in Codec Operation

In addition to external clocks, the codec mode also needs frame sync pulses. Frame sync pulses can be internally generated or accepted from external sources. The transmit frame synchronization pulse (FSX), is accepted or output on pin CMP5/CAP3/FSX, and the receive frame synchronization pulse (FSR), is accepted or output on pin CMP4/CAP3/FSR. SCON bit 13 selects between internal or external frame sync pulses. If bit 13 is 1, external frame sync pulse is selected. In addition to setting bit 13 of SCON to 1, bits 11 and 12 of TCON must be set to 0, to configure pins CMP5/CAP3/FSX and CMP4/CAP2/FSR as Schmidt trigger inputs. Also, bits 8 and 9, and bits 12 and 13 of CCON must be set to 10 to enable positive edge detection on these pins.

Internal frame sync pulses are selected by setting bit 13 of SCON to a 0. Bits 11 and 12 of TCON must be set to 1 to configure pins CMP5/CAP3/FSX and bit 12 of SCON. The frame sync pulse is one clock pulse wide if bit 12 is 0, and two clock pulses wide if bit 12 is 1. If external frame sync pulse is selected, than bit 12 of SCON is ignored and the width of the frame sync pulse is determined by the width of the external frame sync pulse.

Both transmit and receive sections are double-buffered and continous transmission/reception is possible. The maximum transmission/reception rate in codec mode is CLKOUT/4. A higher rate of CLKOUT/3 is possible. To achieve this rate, the transmit and receive clocks must be synchronized with the CLKOUT of the TMS320C14/E14. Six to nine bits of data are allowed in the codec mode, with an optional parity bit. In the codec mode, the MSB is transmitted/received first, unlike the sync and async modes which transmit/receive LSB first.

3.8.5.1 Codec Transmission

The transmit section consists of two 9-bit registers: Transmit Shift register (TSR) and Transmit Buffer register (TSR). Transmission begins after the CPU writes to TBR, and a transmit frame sync pulse (FSX) occurs. If TSR is empty, this data is transferred to the TBR. Otherwise existing data in the TSR is shifted out. Data is transmitted MSB first.

Data is shifted out upon detection of the first rising edge of external clock (CLKX) following receipt of the rising edge of external FSX (frame sync pulse) as shown in Figure 3-41

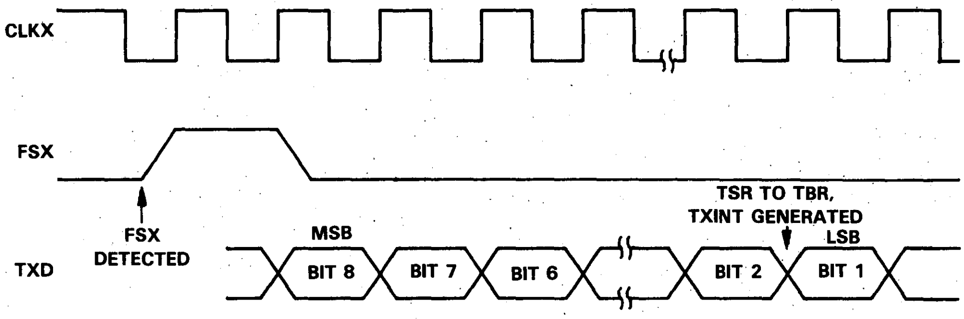


Figure 3-41. Codec Transmit Timing for External Framing

If internal FSX is used, then the data is shifted out on the first rising edge of external clock after TBR has been written to. The data bits are shifted out with an optional parity bit. When the last bit is shifted out, TXINT is generated to the CPU.

To start transmission in codec mode, the TBR has to be written to, and the frame sync pulse (FSX), has to be detected. FSX is detected by its positive edge. If TBR has not been written to by the CPU and FSX is detected, no transmission takes place. Alternatively if FSX is not detected while TBR is written to, transmission will not take place. The user is responsible for insuring TBR is written to before arrival of FSX. If internal frame sync is selected, then FSX will be generated as soon as TBR is written to.

Summary of Transmission in Codec mode:

1) Codec mode is selected by setting bit 14 of SCON to 1.

2) Internal or external FSX is selected with bit 13 of SCON register.

3) If external FSX is selected, the bit 12 of TCON is set to 0 to configure pin CMP5/CAP3/FSX as input. Bit 12 of CCON register is set to 1 to detect positive edge.

4) If internal FSX is selected, the width of FSX is determined by bit 12 of SCON. Bit 12 of TCON is set to 1, to configure pin CMP5/CAP3/FSX as an output.

5) A data word is written to TBR. If TSR is empty, this data is transferred into TSR, otherwise existing data in TSR will be transmitted.

6) If internal FSX is selected, transmission begins on the next rising edge of external clock CLKX.

7) If external FSX is selected, transmission begins on the next rising edge of external clock CLKX, following detection of a rising edge on CMP5/CAP3/FSX pin.

8) 6 to 9 data bits are transmitted followed by an optional parity bit.

9) TXINT is generated to the CPU, and new transmission will begin after TBR is written to.

10) If TBR is already written to, then its data is transferred to TSR. If internal FSX is selected, new transmission will begin on next rising edge of external clock. If external FSX is selected, new transmission will begin after rising edge of FSX is detected.

3.8.5.2 Codec Reception

The receive section consists of two 9-bit registers; Receive Buffer Register (RBR), and Receive Shift Register (RSR). Reception is enabled by setting bit 11 (CREN) in SCON to 1. Data is shifted in to the RSR register on the RXD/DATA pin on the falling edge of CLKR. The external clock is accepted on pin TCLK1/CLKR. Reception is started after a frame sync pulse (FSR) is detected on pin CMP4/CAP2/FSR. Data is received with MSB first.

The selection of internal or external frame sync is common to both transmit and receive sections. If internal FSR (frame sync) is selected, it is output on pin CMP4/CAP/FSR. The width of FSR is controlled by bit 12 of SCON . FSR is sent out on the first positive edge of external clock, after reception is enabled by setting bit 11 (CREN) of SCON to a 1. Data is shifted in upon detection of the second negative edge of external clock following the positive edge of FSR.

If external FSR is selected, bit 13 of SCON must be set to 1. CMP4/CAP2/FSR pin must also be configured as input by setting bit 11 of TCON to 0, and bits 8 and 9 of CCON register must be set to 10 to enable positive edge detection. Once reception is enabled, data is shifted in on the second negative edge of external clock (CLKR), following the rising edge of FSR as shown in Figure 3-42.

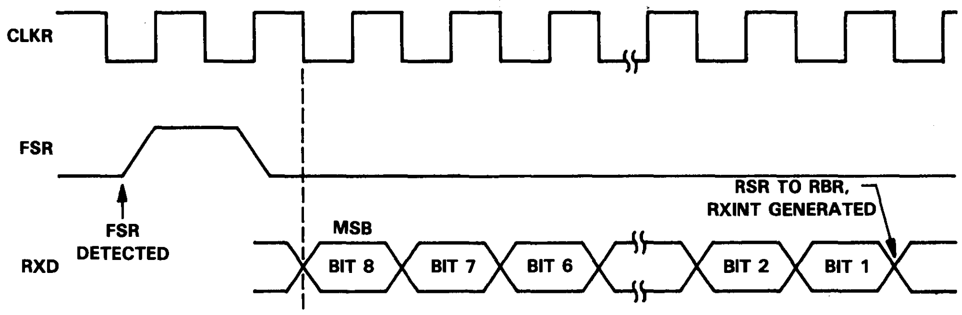


Figure 3-42. Codec Receive Timing for External Framing

The data bits (6-9) are shifted into RSR on successive negative edges of CLKR via the RXD/DATA pin. The data bits are followed by an optional parity bit. After data is shifted into RSR the RXINT is generated to the CPU. If RBR register is empty, the contents of RSR are transferred into RBR.

If RBR is already full, BBF (both buffers full), is set in SCON. In this case contents of RSR will not be transferred into RBR, and both registers will maintain existing data. No further reception will be allowed, and all new data coming into the RXD/DATA pin will be ignored. To allow further reception, RBR must be read by the user. This will automatically clear BBF and contents of RSR will be transferred into RBR, allowingl RSR to receive new data.

If an additional FSR pulse is detected when both buffers are full (i.e. BBF is 1), (ROV, receive overflow flag- bit 7), of SCON is set, indicating incoming data has been lost. The user must clear ROV with in their software.

If the received parity does not match the computed parity on the received word, (PERR (parity error-bit 3), in SCON is set to 1. The user must clear this in their software. PERR is also double buffered. When both buffers are full and PERR is set, the data in RBR has a parity error. If data in RSR also has parity errors, then PERR will be set again when that data is transferred from RSR to RBR

Summary of Reception in Codec Mode

1) Codec mode is selected by setting bit 14 of SCON to 1.

2) Internal or external FSR is selected with bit 13 of SCON

3) If external FSR is selected than bit 11 of TCON register is set to 0 to configure pin CMP4/CAP2/FSR as input, and bit 8 of CCON register is set to 1 to detect positive edge.

4) If internal FSR is selected than width of FSR is determined by bit 12 of SCON register. In addition bit 11 of TCON register must be set to 1 to configure pin CMP4/CAP2/FSR as an output.

5) Reception is enabled by setting bit 11 (CREN) of SCON register to 1.

6) If internal FSR is selected, FSR is output on the first rising edge of external clock, and reception begins on the second falling edge of external clock CLKR.

7) If external FSR is selected, reception begins on the second falling edge of external clock CLKR, following detection of a rising edge on CMP4/CAP2/FSR pin.

8) Six to nine data bits are received followed by an optional parity bit.

9) Interrupt RXINT is generated to the CPU, and computed parity on received data is compared against received parity. If different PERR (bit 3 of SCON) flag is raised.

10) Received word is transferred from RSR to RBR. If RBR contains data, a flag (BBF, both buffers full) is raised in SCON.

11) If transfer to RBR is successful, and continous reception is enabled, RSR is ready to receive new data. Otherwise, reception is stopped until RBR is read and/or reception is enabled.

3.8.6 Communication Protocols

Besides supporting the regular transmission/reception (no protocol), the TMS320C14/E14 supports two communication protocols. These protocols are used for inter-processor communication and allow the processor to ignore all reception until it is being addressed by another device. The first communication protocol, address detect, allows the TMS320C14/E14 to ignore all reception until it detects an address reception. The serial port then wakes up and allows the user to determine via software if the right address was received or not. The second communication protocol, address match allows the serial port to actually match the incoming address with its own address (stored in SMAT register). If the addresses match, the serial port wakes up. If not, all reception is ignored. The communication protocols are selected by bits 9 and 10 of the SCON register.

3.8.6.1 Address Detect Protocol

Address Detect allows the processor to detect when an address is being received in the serial port. This protocol is enabled by setting bit 9, RINTQ1 (Receive interrupt qualifier 1), of the SCON register to 1. This protocol requires that 9-bit data transmission be used. The 9th bit or MSB is used to determine whether address or data is being transmitted. If the 9th bit equals 0, this indicates data is being received, and the serial port ignores the reception. Although data is being shifted into the RSR register, no interrupt is generated to the CPU (i.e. RXINT). If the 9th bit equals 1, interrupt RXINT is generated indicating that an address is being received (see Figure 3-43).

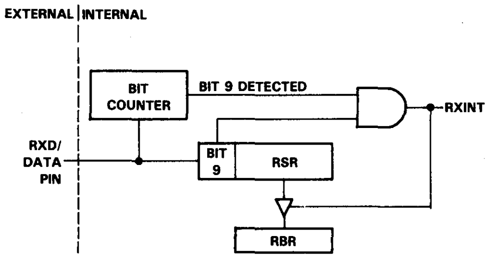


Figure 3-43. Serial Port Using Address Detect Protocol

Upon detection of an address, the serial port wakes up and generates interrupt RXINT to the CPU. The user's software then matches the received address with its own address. If the address does not match, the serial port is allowed to remain in sleep mode. If a match is indicated, the user's software will have to put the serial port in a wake up by setting bit RINTQ1 to 0, disabling the communication protocol. This allows reception of data. When the complete message is received, the RINTQ1 bit can again be set to 1, enabling the protocol and putting the serial port back in a sleep mode. Data is shifted in with the rising edge of the signal. Every ninth clock bit is detected to determine whether the received bits are address or data (see Figure 3-44).

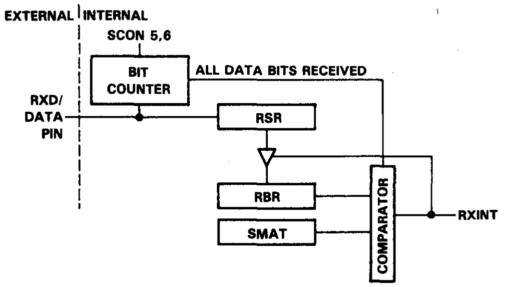


Figure 3-44. Address Detect Reception

The address detect protocol can be used in all three modes; synchronous, asynchronous, and codec modes. However, when this protocol is used, parity must be disabled. The receiver assumes that no parity is being sent along with data.

Summary of Address Detect Protocol:

1) 9 data bits are selected, and parity is disabled.

2) Serial port mode (Synchronous, Asynchronous, or Codec) is selected and reception is enabled.

3) Protocol 1 is selected by setting RINTQ1 bit (bit 9 of SCON) to a 1, also putting serial port in a sleep mode.

4) Data is clocked into the RSR register. The MSB (9th bit) is checked to see if it is 0 or a 1.

5) If msb is 0, incoming word is assumed to be data and nothing happens. Serial port continues in sleep mode.

6) If msb is 1, incoming word is detected as address and interrupt RXINT, is generated to the CPU.

7) User software looks at incoming address and decides whether to wake up or continue sleeping.

8) If it decides to continue sleeping, nothing is done, and serial port will ignore all data clocked in, till another address is detected.

9) If user decides to wake, RINTQ1 (bit 9), of SCON register is set to 0.

10) Normal reception is enabled, and data is clocked into RSR, generating interrupt RXINT each time a word is shifted in.

11) User determines that full message is received and goes back to sleep by setting RINTQ1 bit back to 1, and enabling protocol.

3.8.6.2 Address Match Protocol

Address Match protocol allows the serial port to actually match an incoming address to determine if another device wants to communicate with it. This protocol is enabled by setting bit 10 (RINTQ2 bit) of SCON register to a 1. Address match requires that an address be written to a 9-bit sync match register called SMAT. The incoming word is matched against the value in the SMAT register (see Figure 3-45 and Figure 3-46). If a match is detected, the serial port determines that it is being addressed, and "wakes up". It then generates interrupt RXINT to the CPU. If no match is detected, then data clocked into RSR register is discarded and the serial port remains in the sleep mode.

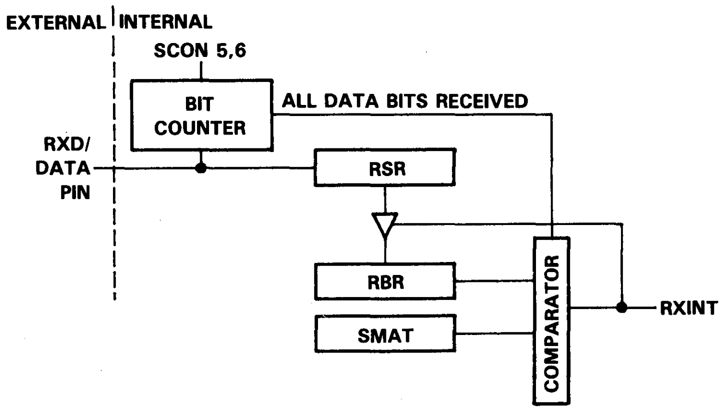
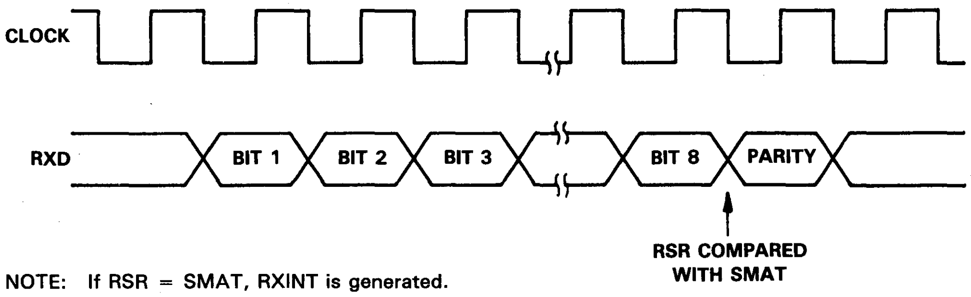


Figure 3-45. Serial Port Using Address Match Protocol

Once an address match is detected, the serial port wakes up by generating an interrupt to the CPU. The user then has to put the serial port in the normal reception mode. This is done by resetting RINTQ2 bit (bit 10) of SCON to a 0. Data will now be clocked into the RSR register, generating interrupt RXINT every time a complete word is received. When the complete message is received, the user can put the serial port back into sleep mode by setting RINTQ2 bit back to 1. Address Match can be used in all three modes of the serial port. It does not require parity to be disabled. Address Match also does not require that the full 9-bit data length be used. If the value written into the SMAT register is less than nine bits, the value must be right-justified, and the remaining upper bits must be zeros. Note that to generate a unique address, the number of bits in the address frame must exceed the number of bits in the data frame.



NOTE: If RSR = SMAT, RXINT is generated.

Figure 3-46. Address Match Reception

Note:

In codec mode, the user must insure that the value written into the SMAT register is in the reverse order (i.e.,the MSB is swapped with the LSB, etc.). This is required because while the SMAT register assumes the standard protocol of LSB transmitted or received first, the codec mode transmits/receives the MSB first.

The SMAT register has a bank address of 7h, and a port address of 2h.

Summary of Address Match Protocol

1) User selects synchronous, asynchronous, or codec modes.
2) Parity and number of data bits are selected.
3) Serial port address (up to 9 bits) is written into SMAT register. If codec mode was selected, then bits are swapped with msb written into lsb of SMAT etc.
4) Protocol 2 is enabled by setting RINTQ2 bit in SCON to 1, also putting serial port in a sleep mode.
5) Reception is enabled.
6) Data is shifted into the RSR register and is compared against the SMAT register.
7) If a match does not occur, the data is discarded and serial port remains in sleep mode.
8) If a match occurs, the serial port wakes up and generates an interrupt (RXINT) to the CPU.
9) User resets RINTQ2 bit to 0 enabling normal reception.
10) Data is clocked into RSR generating an interrupt RXINT, every time a complete word is received.
11) User determines that complete message has been received and puts the serial port back to sleep mode by setting RINTQ2 bit back to 1.
12) Serial port waits for another address match.

Section 4

Assembly Language Instructions

The instruction set of the TMS320C14/E14 processors supports numeric-intensive signal processing operations, such as high-speed control, as well as general-purpose applications. The instruction set shown in Table 4-2 consists primarily of single-cycle, single-word instructions, permitting execution rates of up to 6.25 million instructions per second. Only infrequently used branch and I/O instructions are multicycle.

To support DSP operations, this instruction set, which is essentially the same for all TMS320C1x devices, includes a single-cycle multiply. For ease of use in Harvard architecture, table read (TBLR) and table write (TBLW) instructions are provided, which allow information transfer between data and program memory. The IN and OUT instructions permit a data word to be read into the on-chip RAM from peripherals in only two cycles. The SUBC (conditional subtract) instruction performs the shifting and conditional subtraction necessary to implement a divide efficiently and quickly.

Based on TMS320C10 archetecture, the TMS320C14/E14 is software compatible with the TMS320C1x family, so that software tools may be shared. This section describes the TMS320C14/E14 assembly language instructions. Included in this section are the following major topics:

- Memory Addressing Modes (Section 4.1 on page 4-2)
  Direct addressing
  Indirect addressing (using two auxiliary registers)
  Immediate addressing

- Instruction Set (Section 4.2 on page 4-7)
  Symbols and abbreviations used in the instructions
  Instruction set summary (listed according to function)

- Individual Instruction Descriptions (Section 4.3 on page 4-11)
  Presented in alphabetical order and providing the following:
  - Assembler syntax
  - Operands
  - Execution
  - Encoding
  - Description
  - Words
  - Cycles
  - Example(s)

4.1 Memory Addressing Modes

The TMS320C14/E14 instruction set provides three memory addressing modes:

- Direct addressing mode
- Indirect addressing mode
- Immediate addressing mode.

Both direct and indirect addressing can be used to access data memory. Direct addressing concatenates seven bits of the instruction word with the 1-bit data memory page pointer to form the 8-bit data memory address. Indirect addressing accesses data memory through the two auxiliary registers. In immediate addressing, the data is based on a portion of the instruction word(s). The following sections describe each addressing mode and give the opcode formats and some examples for each mode.

4.1.1 Direct Addressing Mode

In the direct memory addressing mode, the instruction word contains the lower seven bits of the data memory address (dma). This field is concatenated with the one-bit data memory page pointer (DP) register to form the full 8-bit data memory address. This implements a paging scheme in which the first page contains 128 words and the second page also contains 128 words. In a typical application, infrequently accessed system variables, such as those used when performing an interrupt routine, are stored on the second page. The 7-bit address in the instruction points to the specific location within that data memory page. The DP register is loaded through the LDP (load data memory page pointer), LDPK (load data memory page pointer immediate), or LST (load status bits from data memory) instructions. The data page pointer is part of the status register and thus can be stored in data memory.

Note:

The data page pointer is not initialized by reset and is therefore undefined after powerup. The TMS320C14/E14 development tools, however, utilize default values for many parameters, including the data page pointer. Because of this, programs that do not explicitly initialize the data page pointer may execute improperly depending on whether they are executed on a TMS320C14/E14 device or using a development tool. Thus,' it is critical that all programs initialize the data page pointer in software.

Figure 4-1 illustrates how the 8-bit data address is formed.

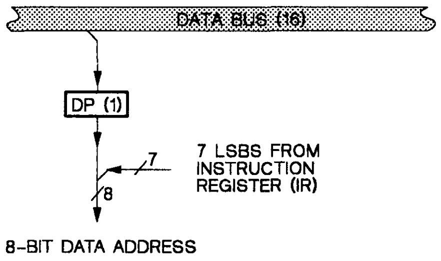


Figure 4-1. Direct Addressing Block Diagram

Direct addressing can be used with all instructions except CALL, branch instructions, immediate operand instructions, and instructions with no operands. The direct addressing format is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode								0	dma						

Bits 15 through 8 contain the opcode. Bit 7 = 0 defines the addressing mode as direct. Bits 6 through 0 contain the data memory address (dma), which can directly address up to 128 words (1 page) of data memory. Use of the data memory page pointer is required to address the full data memory space.

Example of Direct Addressing Format:

ADD 9,5 Add to accumulator the contents of data memory location 9 left-shifted 5 bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	1

The opcode of the ADD 9,5 instruction is 05h and appears in bits 15 through 8. The notation nnh indicates nn is a hexadecimal number. The shift count of 5h appears in bits 11 through 8 of the opcode. The data memory address 09h appears in bits 6 through 0.

4.1.2 Indirect Addressing Mode

Indirect addressing forms the data memory address from the least significant eight bits of one of the two auxiliary registers, AR0 and AR1. This is sufficient to address all the data memory; no paging is necessary with indirect addressing. The Auxiliary Register Pointer (ARP) selects the current auxiliary register. The auxiliary registers can be automatically incremented or decremented in parallel with the execution of any indirect instruction to permit single-cycle manipulation of data tables. The increment/decrement occurs AFTER the current instruction has completed executing.

In indirect addressing, the 8-bit addresses contained in the auxiliary registers may be loaded by the instructions LAR (load auxiliary register) and LARK (load auxiliary register immediate). The auxiliary registers may be modified by the MAR (modify auxiliary register) instruction or, equivalently, by the indirect addressing field of any instruction supporting indirect addressing. AR(ARP) denotes the auxiliary register selected by ARP.

The following symbols are used in indirect addressing:

*	Contents of AR(ARP) are used for data memory address.
*-	Contents of AR(ARP) are used for address, then decremented after data memory access.
*+	Contents of AR(ARP) are used for address, then incremented after data memory access.

The indirect addressing format is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode								1	0	INC	DEC	NAR	0	0	ARP

NOTE: NAR = new auxiliary register control bit.

Bits 15 through 8 contain the opcode, and bit 7 = 1 defines the addressing mode as indirect. Bits 6 through 0 contain the indirect addressing control bits.

Bit 3 and bit 0 control the Auxiliary Register Pointer (ARP). If bit 3 = 0, the contents of bit 0 are loaded into the ARP after execution of the current instruction. If bit 3 = 1, the contents of the ARP remain unchanged. ARP = 0 defines the contents of AR0 as a memory address. ARP = 1 defines the contents of AR1 as a memory address. Note that NAR denotes the new auxiliary register control bit.

Bit 5 and bit 4 control the auxiliary registers. If bit 5 = 1, the current auxiliary register is incremented by 1 after execution. If bit 4 = 1, the current auxiliary register is decremented by 1 after execution. If bit 5 and bit 4 are 0, then neither auxiliary register is incremented nor decremented. Bits 6, 2, and 1 are reserved and should always be programmed to 0.

The auxiliary registers may also be used for temporary storage via the load and store auxiliary register instructions, LAR and SAR, respectively.

The examples that follow illustrate the indirect addressing format. Indirect addressing is indicated by an asterisk (*) in these examples and in the TMS320C1x assembler.

Example 1:

ADD *+,8 Add to the accumulator the contents of the data memory address defined by the contents of the current auxiliary register. This data is left-shifted 8 bits before being added. The current auxiliary register is autoincremented by one. The opcode is 08A8h, as shown below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0

Example 2:

ADD *,8 As in Example 1, but with no autoincrement; the opcode is 0888h.

Example 3:

ADD *-,8 As in Example 1, except that the current auxiliary register is decremented by 1; the opcode is 0898h.

Example 4:

ADD *+,8,1 As in Example 1, except that the auxiliary register pointer is loaded with the value 1 after execution; the opcode is 08A1h.

Example 5:

ADD *+,8,0 As in Example 4, except that the auxiliary register pointer is loaded with the value 0 after execution; the opcode is 08A0h.

4.1.3 Immediate Addressing Mode

Included in the TMS320C1x instruction set are five immediate operand instructions, in which the immediate operand is contained within the instruction word. These instructions execute within a single instruction cycle. The length of the constant operand is instruction-dependent. The immediate instructions are:

LACK	Load accumulator immediate short (8-bit constant)
LARK	Load auxiliary register immediate short (8-bit constant)
LARP	Load auxiliary register pointer immediate(1-bit constant)
LDPK	Load data memory page pointer immediate (1-bit constant)
MPYK	Multiply immediate (13-bit constant)

The following examples illustrate immediate addressing format:

Example 1:

MPYK 2781 Multiply the value 2781 with the contents of the T register. The result is loaded into the P register.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	13-bit constant												

Example 2:

LACK 221 Load the constant 221 in the lower eight bits of the accumulator right-justified. The upper 24 bits of the accumulator are zero.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	0	8-bit constant							

4.2 Instruction Set

The following sections list the symbols and abbreviations used in the TMS320C1x instruction set summary and in the instruction descriptions. The complete instruction set summary is organized according to function. A detailed description of each instruction is listed in the instruction set summary.

4.2.1 Symbols and Abbreviations

Table 4-1 lists symbols and abbreviations used in the instruction set summary (Table 4-2) and the individual instruction descriptions.

Table 4-1. Instruction Symbols

SYMBOL	MEANING
A	Port address
ACC	Accumulator
ARn	Auxiliary Register n (AR0 and AR1) are predefined assembler symbols equal to 0 and 1, respectively.)
ARP	Auxiliary register pointer
B	Branch address
D	Data memory address field
DATn	Label assigned to data memory location n
dma	Data memory address or direct memory address
DP	Data page pointer
I	Addressing mode bit
INTM	Interrupt mode bit
K	Immediate operand field
nnh	Indicates nn is a hexadecimal number. (All others are assumed to be decimal values.)
OVM	Overflow (saturation) mode flag bit
P	Product register
PA	Port address (PA0 through PA7 are predefined assembler symbols equal to 0 through 7, respectively.)
PC	Program counter
pma	Program memory address
PRGn	Label assigned to program memory location n
R	1-bit operand field specifying auxiliary register
S	4-bit left-shift code
T	Temporary register
TOS	Top of stack
X	3-bit accumulator left-shift field
→	Is assigned to
| |	An absolute value
< >	User-defined items
[ ]	Optional items
( )	"Contents of"
{ }	Alternative items, one of which must be entered
< >	Angle brackets back-to-back indicate "not equal".
	Blanks or spaces must be entered where shown.

4.2.2 Instruction Set Summary

Table 4-2 provides the TMS320C1x instruction set summary, arranged according to function and alphabetized within each functional grouping. Additional information is presented in the individual instruction descriptions in the following section.

The instruction set summary consists primarily of single-cycle, single-word instructions. Only infrequently used branch and I/O instructions require multiple cycles.

Table 4-2. Instruction Set Summary

ACCUMULATOR MEMORY REFERENCE INSTRUCTIONS				
Mnemonic and Description		**Cycles**	**Words**	**16-Bit Opcode** **MSB** **LSB**
ABS	Absolute value of accumulator	1	1	0111 1111 1000 1000
ADD	Add to accumulator with shift	1	1	0000 SSSS I DDD DDDD
ADDH	Add to high accumulator	1	1	0110 0000 I DDD DDDD
ADDS	Add to low accumulator with sign-extension suppressed	1	1	0110 0001 I DDD DDDD
AND	AND with accumulator	1	1	0111 1001 I DDD DDDD
LAC	Load accumulator with shift	1	1	0010 SSSS I DDD DDDD
LACK	Load accumulator immediate short	1	1	0111 1110 KKKK KKKK
OR	OR with accumulator	1	1	0111 1010 I DDD DDDD
SACH	Store high accumulator with shift	1	1	0101 1XXX I DDD DDDD
SACL	Store low accumulator	1	1	0101 0000 I DDD DDDD
SUB	Subtract from accumulator with shift	1	1	0001 SSSS I DDD DDDD
SUBC	Conditional subtract	1	1	0110 0100 I DDD DDDD
SUBH	Subtract from high accumulator	1	1	0110 0010 I DDD DDDD
SUBS	Subtract from low accumulator with sign-extension suppressed	1	1	0110 0011 I DDD DDDD
XOR	Exclusive-OR with low accumulator	1	1	0111 1000 I DDD DDDD
ZAC	Zero accumulator	1	1	0111 1111 1000 1001
ZALH	Zero low accumulator and load high accumulator	1	1	0110 0101 I DDD DDDD
ZALS	Zero accumulator and load low accumulator with sign-extension suppressed	1	1	0110 0110 I DDD DDDD

AUXILIARY REGISTER AND DATA PAGE POINTER INSTRUCTIONS				
Mnemonic and Description		**Cycles**	**Words**	**16-Bit Opcode** **MSB** **LSB**
LAR	Load auxiliary register	1	1	0011 100R I DDD DDDD
LARK	Load auxiliary register immediate short	1	1	0111 000R KKKK KKKK
LARP	Load auxiliary register pointer immediate	1	1	0110 1000 1000 000K
LDP	Load data memory page pointer	1	1	0110 1111 I DDD DDDD
LDPK	Load data memory page pointer immediate	1	1	0110 1110 0000 000K
MAR	Modify auxiliary register	1	1	0110 1000 I DDD DDDD
SAR	Store auxiliary register	1	1	0011 000R I DDD DDDD

Table 4-2. Instruction Set Summary (Continued)

T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS				
Mnemonic and Description		Cycles	Words	16-Bit Opcode MSB LSB
APAC	Add P register to accumulator	1	1	0111 1111 1000 1111
LT	Load T register	1	1	0110 1010 I DDD DDDD
LTA	Load T register and accumulate previous product	1	1	0110 1100 I DDD DDDD
LTD	Load T register, accumulate previous product, and move data	1	1	0110 1011 I DDD DDDD
MPY	Multiply (with T register, store product in P register)	1	1	0110 1101 I DDD DDDD
MPYK	Multiply immediate	1	1	100K KKKK KKKK KKKK
PAC	Load accumulator with P register	1	1	0111 1111 1000 1110
SPAC	Subtract P register from accumulator	1	1	0111 1111 1001 0000

BRANCH/CALL INSTRUCTIONS				
Mnemonic and Description		Cycles	Words	16-Bit Opcode MSB LSB
B	Branch unconditionally	2	2	1111 1001 0000 0000 0000 BBBB BBBB BBBB
BANZ	Branch on auxiliary register not zero	2	2	1111 0100 0000 0000 0000 BBBB BBBB BBBB
BGEZ	Branch if accumulator $\geq$ 0	2	2	1111 1101 0000 0000 0000 BBBB BBBB BBBB
BGZ	Branch if accumulator > 0	2	2	1111 1100 0000 0000 .0000 BBBB BBBB BBBB
BLEZ	Branch if accumulator $\leq$ 0	2	2	1111 1011 0000 0000 0000 BBBB BBBB BBBB
BLZ	Branch if accumulator < 0	2	2	1111 1010 0000 0000 0000 BBBB BBBB BBBB
BNZ	Branch if accumulator $\neq$ 0	2	2	1111 1110 0000 0000 0000 BBBB BBBB BBBB
BV	Branch on overflow	2	2	1111 0101 0000 0000 0000 BBBB BBBB BBBB
BZ	Branch if accumulator = 0	2	2	1111 1111 0000 0000 0000 BBBB BBBB BBBB
CALA	Call subroutine indirect	2	1	0111 1111 1000 1100
CALL	Call subroutine	2	2	1111 1000 0000 0000 0000 BBBB BBBB BBBB
RET	Return from subroutine	2	1	0111 1111 1000 1101

Note: The TMS320C14/E14 does not have the $\overline{\text{BIO}}$ pin present on other TMS320C1x devices. An attempt to execute the BIOZ (Branch on $\overline{\text{BIO}}$ low) instruction will result in a two cycle NOP action.

CONTROL INSTRUCTIONS				
Mnemonic and Description		Cycles	Words	16-Bit Opcode MSB LSB
DINT	Disable interrupt	1	1	0111 1111 1000 0001
EINT	Enable interrupt	1	1	0111 1111 1000 0010
LST	Load status register from data memory	1	1	0111 1011 I DDD DDDD
NOP	No operation	1	1	0111 1111 1000 0000
POP	Pop top of stack to low accumulator	2	1	0111 1111 1001 1101
PUSH	Push low accumulator onto stack	2	1	0111 1111 1001 1100
ROVM	Reset overflow mode	1	1	0111 1111 1000 1010
SOVM	Set overflow mode	1	1	0111 1111 1000 1011
SST	Store status register	1	1	0111 1100 I DDD DDDD

Table 4-2. Instruction Set Summary (Concluded)

I/O AND DATA MEMORY OPERATIONS							
Mnemonic and Description		Cycles	Words	16-Bit Opcode			
				MSB			LSB
DMOV	Data move in data memory	1	1	0110	1001	I DDD	DDDD
IN	Input data from port	2	1	0100	0AAA	I DDD	DDDD
OUT	Output data to port	2	1	0100	1AAA	I DDD	DDDD
TBLR	Table read	3	1	0110	0111	I DDD	DDDD
TBLW	Table write	3	1	0111	1101	I DDD	DDDD

4.3 Individual Instruction Descriptions

Each instruction in the instruction set summary is described in the following pages. Instructions are listed in alphabetical order. Information, such as assembler syntax, operands, execution, encoding, description, words, cycles, and examples, is provided for each instruction. An example instruction is provided on the next two pages to familiarize the user with the special format used and explain its content. Refer to Section 4.1 for further information on memory addressing. Code examples using many of the instructions are given in Section 5 on Software Applications.

Syntax

Direct:	[<label>] EXAMPLE <dma>[,<shift>]
Indirect:	[<label>] EXAMPLE {*|*+|*-}[,<shift>[,<next ARP>]]
Immediate:	[<label>] EXAMPLE [<constant>]

Each instruction begins with an assembler syntax expression. The optional comment field that concludes the syntax is not included in the syntax expression. Space(s) are required between each field (label, command, operand, and comment fields) as shown in the syntax. The syntax example illustrates both direct and indirect addressing, as well as immediate addressing in which the operand field includes <constant>.

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1
$0 \leq \text{constant} \leq 255$

Operands may be constants or assembly-time expressions referring to memory, I/O and register addresses, pointers, shift counts, and a variety of constants. The operand values used in the example syntax are shown.

Execution

$(\text{PC}) + 1 \rightarrow \text{PC}$
$(\text{ACC}) + (\text{dma}) \times 2^{\text{shift}} \rightarrow \text{ACC}$

$1 \rightarrow$ interrupt mode (INTM) status bit
Affects INTM.

This section provides an example of the instruction operation sequence, describing the processing that takes place when the instruction is executed. Conditional effects of status register specified modes are also given. In addition, those bits in the status registers that are affected by the instruction are listed.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	0	0	0	Shift				0	Data Memory Address						
Indirect:	0	0	0	0	Shift				1	See Section 4.1						
Immediate:	1	0	0	13-Bit Constant												

Opcode examples are shown of both direct and indirect addressing or of the use of an immediate operand.

Description This section decribes the instruction execution and its effect on the rest of the processor or memory contents. Any constraints on the operands imposed by the processor or the assembler are also described here. The description parallels and supplements the information given by the execution block.

Words 1

The digit specifies the number of memory words required to store the instruction and its extension words.

Cycles 1

The digit specifies the number of cycles required to execute the instruction.

Example 1

```
ADD    DAT1,3    (DP = 0)
or
ADD    *,3       If current auxiliary register contains 1.
```

	Before Instruction		After Instruction
Data Memory 1	2h	Data Memory 1	2h
ACC	7h	ACC	17h

The sample code presented in the above format shows the effect of the code on memory and/or registers.

Syntax [<label>] ABS

Operands None

Execution
(PC) + 1 → PC
If (ACC) < 0:
 Then −(ACC) → ACC
Affects OV; affected by OVM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0

Description If the contents of the accumulator are greater than or equal to zero, the accumulator is unchanged by the execution of ABS. If the contents of the accumulator are less than zero, the accumulator is replaced by its two's-complement value.

Note that 80000000h is a special case. When the overflow mode is not set, the ABS of 80000000h is 80000000h. When in the overflow mode, the ABS of 80000000h is 7FFFFFFFh.

Words 1

Cycles 1

Example

```
ABS
```

	Before Instruction		After Instruction
ACC	1234h	ACC	1234h
ACC	FFFFFFFFh	ACC	1h

Add to Accumulator with Shift — ADD

Syntax

Direct: [<label>] ADD <dma>[,<shift>]

Indirect: [<label>] ADD {*|*+|*-}[,<shift>[,<next ARP>]]

Operands

$0 \leq \text{dma} \leq 127$

ARP = 0 or 1

Execution

$(PC) + 1 \rightarrow PC$

$(ACC) + (dma) \times 2^{shift} \rightarrow ACC$

Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	0	0	0	Shift				0	Data Memory Address						
Indirect:	0	0	0	0	Shift				1	See Section 4.1						

Description

Contents of the addressed data memory location are left-shifted and added to the accumulator. During shifting, low-order bits are zero-filled, and high-order bits are sign-extended. The result is stored in the accumulator.

Words 1

Cycles 1

Example

```
ADD   DAT1,3      (DP = 0)
or
ADD   *,3         If current auxiliary register contains 1.
```

	Before Instruction		After Instruction
Data Memory 1	2h	Data Memory 1	2h
ACC	7h	ACC	17h

Example

```
ADD   DAT2,4      (DP = 0)
or
ADD   *,4         If current auxiliary register contains 2.
```

	Before Instruction		After Instruction
Data Memory 2	8B0Eh	Data Memory 2	8B0Eh
ACC	0h	ACC	FFF8B0E0h

Syntax

Direct: [<label>] ADDH <dma>
Indirect: [<label>] ADDH {*|*+|*-}[,<next ARP>]

Operands $0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution (PC) + 1 → PC
(ACC) + (dma) x 2^{16} → ACC
Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	0	0	0	0	Data Memory Address						
Indirect:	0	1	1	0	0	0	0	0	1	See Section 4.1						

Description Contents of the addressed data memory location are added to the upper half of the accumulator (bits 31 through 16). Low-order bits are unaffected by ADDH.

The ADDH instruction may be used in performing 32-bit arithmetic.

Words 1

Cycles 1

Example 1

```
ADDH   DAT5    (DP = 0)
or
ADDH   *       If current auxiliary register contains 5.
```

	Before Instruction		After Instruction
Data Memory 5	4h	Data Memory 5	4h
ACC	13h	ACC	40013h

Add to Accumulator with Sign-Extension Suppressed

Syntax

Direct: [<label>] ADDS <dma>
Indirect: [<label>] ADDS {*|*+|*-}[,<next ARP>]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution

(PC) + 1 → PC
(ACC) + (dma) → ACC
(dma) is a 16-bit unsigned number.
Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	0	0	1	0	Data Memory Address						
Indirect:	0	1	1	0	0	0	0	1	1	See Section 4.1						

Description

Contents of the specified data memory location are added with sign-extension suppressed. The data is treated as a 16-bit unsigned number rather than a two's-complement number. Therefore, there is no sign-extension as with the ADD instruction.

The ADDS instruction can be used in implementing 32-bit arithmetic.

Words 1

Cycles 1

Example 2

```
ADDS    DAT11    (DP = 0)
or
ADDS    *        If current auxiliary register contains 11.
```

	Before Instruction		After Instruction
Data Memory 11	0F006h	Data Memory 11	0F006h
ACC	3h	ACC	0F009h

Syntax

Direct: [<label>] AND <dma>
Indirect: [<label>] AND {*|*+|*-}[,<next ARP>]

Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Execution

(PC) + 1 → PC
(ACC(15-0)).AND.(dma) → ACC(15-0)
0 → ACC(31-16)

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	1	1	0	0	1	0	Data Memory Address						
Indirect:	0	1	1	1	1	0	0	1	1	See Section 4.1						

Description

The lower half of the accumulator is ANDed with the contents of the addressed data memory location. The upper half of the accumulator is ANDed with all zeroes. Therefore, the upper half of the accumulator is always zeroed by the AND instruction.

Words

1

Cycles

1

Example 1

```
AND     DAT16   (DP = 0)
or
AND     *       If current auxiliary register contains 16.
```

	Before Instruction		After Instruction
Data Memory 16	0FFh	Data Memory 16	0FFh
ACC	12345678h	ACC	78h

Add P Register to Accumulator — APAC

Syntax [<label>] APAC

Operands None

Execution (PC) + 1 → PC
(ACC) + (P register) → ACC
Affects OV; affected by OVM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1

Description The contents of the P register, the result of a multiply, are added to the contents of the accumulator. The result is stored in the accumulator.

The APAC instruction is a subset of the LTA and LTD instructions.

Words 1

Cycles 1

Example `APAC`

	Before Instruction		After Instruction
P	40h	P	40h
ACC	20h	ACC	60h

Syntax [<label>] B <pma>

Operands $0 \leq \text{pma} \leq 4095$

Execution pma → PC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
Program Memory Address															

Description Control passes to the designated program memory address (pma). Pma can be either a symbolic or a numeric address.

Words 2

Cycles 2

Example

```
B     PRG191     191 is loaded into the program counter,
                 and the program continues running from
                 that location.
```

Syntax [<label>] BANZ <pma>

Operands $0 \leq \text{pma} \leq 4095$

Execution
If (AR bits 8-0) $\neq$ 0:
 Then pma $\rightarrow$ PC;
 Else (PC) + 2 $\rightarrow$ PC
(AR) - 1 $\rightarrow$ AR.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the lower nine bits of the current auxiliary register are not equal to zero, then the address contained in the following word is loaded into the program counter. If these bits are equal to zero, the current program counter is incremented by two. In either case, the auxiliary register is decremented. Note that the test for zero is performed before decrementing the auxiliary register. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example

```
BANZ    PRG35
```

	Before Instruction		After Instruction
AR	1h	AR	0h
PC	46h	PC	35h

or

	Before Instruction		After Instruction
AR	0h	AR	0FFFFh
PC	46h	PC	48h

Note:

BANZ is designed for loop control using the auxiliary registers as loop counters. The auxiliary register is decremented after testing for zero. The auxiliary registers also behave as modulo 512 counters.

Branch if Accumulator Greater Than or Equal to Zero

Syntax [<label>] BGEZ <pma>

Operands $0 \leq \text{pma} \leq 4095$

Execution If (ACC) ≥ 0:
Then pma → PC;
Else (PC) + 2 → PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the contents of the accumulator are greater than or equal to zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example

```
BGEZ    PRG217      217 is loaded into the program counter
                    if the accumulator is greater than or
                    equal to zero.
```

Branch if Accumulator Greater Than Zero BGZ

Syntax [<label>] BGZ <pma>

Operands $0 \leq \text{pma} \leq 4095$

Execution
```
If (ACC) > 0:
  Then pma → PC;
  Else (PC) + 2 → PC.
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the contents of the accumulator are greater than zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example 1
```
BGZ     PRG342     342 is loaded into the program counter
                   if the accumulator is greater than zero.
```

Branch if Accumulator Less Than or Equal to Zero

Syntax [<label>] BLEZ <pma>

Operands $0 \leq \text{pma} \leq 4095$

Execution If (ACC) $\leq$ 0:
Then pma $\rightarrow$ PC;
Else (PC) + 2 $\rightarrow$ PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the contents of the accumulator are less than or equal to zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example 2

```
BLEZ    PRG63     63 is loaded into the program counter if
                  the accumulator is less than or equal to
                  zero.
```

Branch if Accumulator Less Than Zero BLZ

Syntax [<label>] BLZ <pma>

Operands 0 ≤ pma ≤ 4095

Execution
If (ACC) < 0:
 Then pma → PC;
 Else (PC) + 2 → PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the contents of the accumulator are less than zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example 1

```
BLZ    PRG481    481 is loaded into the program counter if
                 the accumulator is less than zero.
```

Branch if Accumulator Not Equal to Zero

Syntax [<label>] BNZ <pma>

Operands $0 \leq \text{pma} \leq 4095$

Execution
If $(\text{ACC}) \neq 0$:
 Then pma $\rightarrow$ PC;
 Else (PC) + 2 $\rightarrow$ PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the contents of the accumulator are not equal to zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example

```
BNZ     PRG320     320 is loaded into the program counter
                   if the accumulator does not equal zero.
```

Branch on Overflow — BV

Syntax [<label>] BV <pma>

Operands $0 \leq \text{pma} \leq 4095$

Execution
If overflow (OV) status bit = 1:
 Then pma → PC and 0 → OV;
 Else (PC) + 2 → PC.
Affects OV; affected by OV.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the overflow (OV) flag has been set, then a branch to the specified program memory location occurs and the overflow flag is cleared. Otherwise, the program counter is incremented to the next instruction. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example

```
BV      PRG610     If an overflow has occurred since the
                   overflow flag was last cleared, then 610
                   is loaded into the program counter and
                   OV is cleared. Otherwise, the program
                   counter is incremented.
```

Syntax [<label>] BZ <pma>

Operands $0 \leq \text{pma} \leq 4095$

Execution
If (ACC) = 0:
 Then pma → PC;
 Else (PC) + 2 → PC.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the contents of the accumulator are equal to zero, then branch to the specified program memory location. The branch to a location in program is specified by the program memory address (pma). Pma can be either a symbolic or numeric address.

Words 2

Cycles 2

Example

```
BZ      PRG102      102 is loaded into the program counter
                    if the accumulator is equal to zero.
```

Call Subroutine Indirect — CALA

Syntax [<label>] CALA

Operands None

Execution (PC) + 1 → TOS
(ACC(11-0)) → PC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0

Description The current program counter is incremented and pushed onto the top of the stack. Then, the contents of the 12 least significant bits of the accumulator are loaded into the PC.

The CALA instruction is used to perform computed subroutine calls.

Words 1

Cycles 2

Example

```
CALA
```

	Before Instruction		After Instruction
PC	25h	PC	83h
ACC	83h	ACC	83h
Stack	32h 75h 84h 49h	Stack	26h 32h 75h 84h

Syntax [<label>] CALL <pma>

Operands 0 ≤ pma ≤ 4095

Execution (PC) + 2 → TOS
pma → PC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description The current program counter is incremented by two and pushed onto the top of the stack. The specified program memory address (pma) is then loaded into the PC. Pma can be either a symbolic or a numeric address.

Words 2

Cycles 2

Example

```
CALL    PRG109
```

	Before Instruction		After Instruction
PC	33h	PC	6Dh
Stack	71h 48h 16h 80h	Stack	35h 71h 48h 16h

Disable Interrupt   DINT

Syntax [<label>] DINT

Operands None

Execution (PC) + 1 → PC
1 → interrupt mode (INTM) status bit
Affects INTM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1

Description The interrupt mode (INTM) status bit is set to logic 1. Maskable interrupts are disabled immediately after the DINT instruction executes. Interrupts are also disabled by a reset. Note that the LST instruction does not affect INTM.

Note that $\overline{\text{RS}}$ and $\overline{\text{NMI}}$ is not disabled by this instruction.

Words 1

Cycles 1

Example

```
DINT            Maskable interrupts are disabled, and INTM
                is set to one.
```

Syntax

Direct: [<label>] DMOV <dma>
Indirect: [<label>] DMOV {*|*+|*-}[,<next ARP>]

Operands $0 \le \text{dma} \le 127$
ARP = 0 or 1

Execution (PC) + 1 $\rightarrow$ PC
(dma) $\rightarrow$ dma + 1

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	1	0	0	1	0	Data Memory Address						
Indirect:	0	1	1	0	1	0	0	1	1	See Section 4.1						

Description The contents of the specified data memory address are copied into the contents of the next higher address. When data is copied from the addressed location to the next higher location, the contents of the addressed location remain unaltered.

The data move function is useful in implementing the z^{-1} delay encountered in digital signal processing. The DMOV function is included in the LTD instruction (see LTD for more information).

Words 1

Cycles 1

Example

```
DMOV    DAT8
or
DMOV    *       If current auxiliary register contains 8.
```

	Before Instruction		After Instruction
Data Memory 8	43h	Data Memory 8	43h
Data Memory 9	2h	Data Memory 9	43h

Syntax [<label>] EINT

Operands None

Execution (PC) + 1 → PC
0 → interrupt mode (INTM) status bit
Affects INTM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0

Description The interrupt mode (INTM) status bit is cleared to logic 0. Maskable interrupts are enabled after the instruction following EINT executes. This allows an interrupt service routine to re-enable interrupts and execute a RET instruction before any other pending interrupts are processed. Note that the EINT instruction should not be used immediately preceding a branch instruction.

The LST instruction does not affect INTM. (See the DINT instruction for further information.)

Words 1

Cycles 1

Example

```
EINT             Maskable interrupts are enabled, and INTM
                 is set to zero.
```

Syntax

Direct: [<label>] IN <dma>,<PA>
Indirect: [<label>] IN {*|*+|*-},<PA>[,<next ARP>]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1
$0 \leq$ port address PA ≤ 7

Execution

(PC) + 1 → PC
Port address → address lines A2/PA2-A0/PA0
1 → address bus A11-A3
Data bus D15-D0 → dma

Encoding

	15	14	13	12	11	10 9 8	7	6 5 4 3 2 1 0
Direct:	0	1	0	0	0	Port Address	0	Data Memory Address
Indirect:	0	1	0	0	0	Port Address	1	See Section 4.1

Description

The IN instruction reads data from a peripheral and places it in data memory. This is a two-cycle instruction. During the first cycle, the port address is sent to address lines A2/PA2-A0/PA0. $\overline{\text{REN}}$ goes low during the same cycle, strobing in the data that the addressed peripheral places on the data bus D15-D0. The upper address lines A11 - A3 are held high.

Words

1

Cycles

2

Example

```
IN      STAT,PA5      Read in word from peripheral on port
                      address 5.  Store in data memory
                      location STAT.
```

or

```
LARK    1,20          Load AR1 with decimal 20.
LARP    1             Load ARP with decimal 1.
IN      *-,PA1,0      Read in word from peripheral on port
                      address 1.  Store in data memory
                      location 20.  Decrement AR1 to 19.
                      Load the ARP with 0.
```

Load Accumulator with Shift — LAC

Syntax

Direct: [<label>] LAC <dma>[,<shift>]
Indirect: [<label>] LAC {*|*+|*-}[,<shift>[,<next ARP>]]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1
$0 \leq \text{shift} \leq 15$ (defaults to 0)

Execution

$(\text{PC}) + 1 \rightarrow \text{PC}$
$(\text{dma}) \times 2^{\text{shift}} \rightarrow \text{ACC}$

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	0	1	0	Shift				0	Data Memory Address						
Indirect:	0	0	1	0	Shift				1	See Section 4.1						

Description

Contents of the specified data memory address are left-shifted and loaded into the accumulator. During shifting, low-order bits are zero-filled. High-order bits are sign-extended.

Words 1

Cycles 1

Example

```
LAC    DAT6,4   (DP = 0)
```
or
```
LAC    *,4      If current auxiliary register contains 6.
```

	Before Instruction		After Instruction
Data Memory 6	1h	Data Memory 6	1h
ACC	0h	ACC	10h

Syntax [<label>] LACK <constant>

Operands 0 ≤ constant ≤ 255

Execution (PC) + 1 → PC
8-bit positive constant → ACC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	0	8-Bit Constant							

Description The 8-bit constant is loaded into the accumulator right-justified. The upper 24 bits of the accumulator are zeroed (i.e., sign extension is suppressed).

Words 1

Cycles 1

Example

```
LACK    15h
```

	Before Instruction		After Instruction
ACC	31h	ACC	15h

Syntax

Direct: [<label>] LAR <AR>,<dma>
Indirect: [<label>] LAR <AR>,{*|*+|*-}[,<next ARP>]

Operands

$0 \leq \text{dma} \leq 127$
auxiliary register AR = 0 or 1
ARP = 0 or 1

Execution

$(\text{PC}) + 1 \rightarrow \text{PC}$
$(\text{dma}) \rightarrow$ auxiliary register AR

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	0	1	1	1	0	0	AR	0	Data Memory Address						
Indirect:	0	0	1	1	1	0	0	AR	1	See Section 4.1						

Description

The contents of the specified data memory address are loaded into the designated auxiliary register. The LAR and SAR (store auxiliary register) instructions can be used to load and store the auxiliary registers during subroutine calls and interrupts. If an auxiliary register is not being used for indirect addressing, LAR and SAR enable the register to be used as an additional storage register, especially for swapping values between data memory locations without affecting the contents of the accumulator.

If indirect addressing with autodecrement is used with LAR to load the current auxiliary register, the new value of the auxiliary register is not decremented as a result of instruction execution. The analagous case is true with autoincrement.

Words

1

Cycles

1

Example

```
LAR     AR0,DAT19
```

	Before Instruction		After Instruction
Data Memory 19	18h	Data Memory 19	18h
AR0	6h	AR0	18h

also,

```
LARP   0
LAR    AR0,*-
```

Data Memory 7	32h	Data Memory 7	32h

AR0	7h	AR0	32h

Syntax [<label>] LARK <AR>,<constant>

Operands 0 ≤ constant ≤ 255
auxiliary register AR = 0 or 1

Execution (PC) + 1 → PC
8-bit constant → auxiliary register AR

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	AR	8-Bit Constant							

Description The 8-bit positive constant is loaded into the designated auxiliary register right-justified and zero-filled (i.e., sign-extension suppressed).

LARK is useful for loading an initial loop counter value into an auxiliary register for use with the BANZ instruction.

Words 1

Cycles 1

Example

```
LARK    AR0,21h
```

	Before Instruction		After Instruction
AR0	0h	AR0	21h

Syntax [<label>] LARP <constant>

Operands 0 ≤ constant ≤ 1

Execution (PC) + 1 → PC
Constant → ARP
Affects ARP

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	ARP

Description The auxiliary register pointer is loaded with the one-bit constant identifying the desired auxiliary register. ARP can also be modified by the LST and MAR instructions, as well as any instruction that is used in the indirect addressing mode.

The LARP instruction is a subset of MAR; i.e., the opcode is the same as MAR in the indirect addressing mode. The instruction `MAR *,<next ARP>` has the same effect as LARP.

Words 1

Cycles 1

Example

```
LARP     1        Any succeeding instructions will use
                  auxiliary register AR1 for indirect
                  addressing.
```

Syntax

Direct: [<label>] LDP <dma>
Indirect: [<label>] LDP {*|*+|*-}[,<next ARP>]

Operands 0 ≤ dma ≤ 127
ARP = 0 or 1

Execution (PC) + 1 → PC
LSB of (dma) → data memory page pointer (DP = 0 or 1)
Affects DP.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	1	1	1	1	0	Data Memory Address						
Indirect:	0	1	1	0	1	1	1	1	1	See Section 4.1						

Description The least significant bit of the contents of the specified data memory address is loaded into the DP (data memory page pointer) register. All higher-order bits are ignored in the data word. DP = 0 defines page 0 that contains words 0-127. DP = 1 defines page 1 that contains words 128-255. The DP may also be loaded by the LST and LDPK instructions.

Words 1

Cycles 1

Example

```
LDP    DAT1     LSB of location DAT1 is loaded into DP.
or
LDP    *,1      LSB of location currently addressed by
                auxiliary register is loaded into DP.
                ARP is set to 1.
```

	Before Instruction		After Instruction
Data Memory 1	0FEDCh	Data Memory 1	0FEDCh
DP	1h	DP	0h

Syntax [<label>] LDPK <constant>

Operands 0 ≤ constant ≤ 1

Execution (PC) + 1 → PC
Constant → data memory page pointer (DP)
Affects DP.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	DP

Description The DP (data memory page pointer) register is loaded with a 1-bit constant. DP = 0 defines page 0 that contains words 0-127. DP = 1 defines page 1 that contains words 128-255. The DP may also be loaded by the LST and LDP instructions.

Words 1

Cycles 1

Example

```
LDPK   0    The data page pointer is set to 0.
```

Load Status Register from Data Memory — LST

Syntax

Direct: [<label>] LST <dma>
Indirect: [<label>] LST {*|*+|*-}[,<next ARP>]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution

(PC) + 1 → PC
(dma) → status register bits
Affects ARP, OV, OVM, and DP.
Does not affect INTM.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	1	1	0	1	1	0	Data Memory Address						
Indirect:	0	1	1	1	1	0	1	1	1	See Section 4.1						

Description

The status register is loaded with the addressed data memory value. Note that the INTM (interrupt mode) bit is unaffected by LST.

The LST instruction is used to load the status register after interrupts and subroutine calls. The status register contains the status bits: OV (overflow flag) bit, OVM (overflow mode) bit, ARP (auxiliary register pointer), and DP (data memory page pointer). These bits were stored (by the SST instruction) in the data memory word as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OV	OVM	INTM	1	1	1	1	ARP	1	1	1	1	1	1	0	DP

Words 1

Cycles 1

Example

```
LARP    0
LST     *,1     The data memory word addressed by the
                contents of auxiliary register AR0
                replaces the status bits. ARP becomes 1.
```

Note:

When using direct addressing, the SST instruction always saves status on page 1. The LST instruction will not automatically restore status from page 1. Therefore, the user must specify the correct data page pointer.

Syntax

Direct: [<label>] LT <dma>
Indirect: [<label>] LT {*|*+|*-}[,<next ARP>]

Operands 0 ≤ dma ≤ 127
ARP = 0 or 1

Execution (PC) + 1 → PC
(dma) → T register

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	1	0	1	0	0	Data Memory Address						
Indirect:	0	1	1	0	1	0	1	0	1	See Section 4.1						

Description The T register is loaded with the contents of the specified data memory location. The LT instruction may be used to load the T register in preparation for multiplication (see the LTA, LTD, MPY, and MPYK instructions).

Words 1

Cycles 1

Example

```
LT    DAT24    (DP = 0)
```
or
```
LT    *        If current auxiliary register contains 24.
```

	Before Instruction		After Instruction
Data Memory 24	62h	Data Memory 24	62h
T	3h	T	62h

Syntax

Direct: [<label>] LTA <dma>
Indirect: [<label>] LTA {*|*+|*-}[,<next ARP>]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution

(PC) + 1 → PC
(dma) → T register
(ACC) + (P register) → ACC
Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	1	1	0	0	0	Data Memory Address						
Indirect:	0	1	1	0	1	1	0	0	1	See Section 4.1						

Description

The T register is loaded with the contents of the specified data memory address. The P register, containing the previous product of the multiply operation, is added to the accumulator, and the result is stored in the accumulator.

The function of the LTA instruction is included in the LTD instruction.

Words 1

Cycles 1

Example

```
LTA    DAT24    (DP = 0)
or
LTA    *        If current auxiliary register contains 24.
```

	Before Instruction		After Instruction
Data Memory 24	62h	Data Memory 24	62h
T	3h	T	62h
P	0Fh	P	0Fh
ACC	5h	ACC	14h

Load T Register, Accumulate Previous Product, and Move Data

Syntax

Direct: [<label>] LTD <dma>
Indirect: [<label>] LTD {*|*+|*-}[,<next ARP>]

Operands $0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution
(PC) + 1 → PC
(dma) → T register
(dma) → dma + 1
(ACC) + (P register) → ACC
Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	1	0	1	1	0	Data Memory Address						
Indirect:	0	1	1	0	1	0	1	1	1	See Section 4.1						

Description The T register is loaded with the contents of the specified data memory address. The contents of the P register are added to the accumulator, and the result is placed in the accumulator. The contents of the specified data memory address are also copied to the next higher data memory address. This function is described under the instruction DMOV.

Words 1

Cycles 1

Example

```
LTD     DAT24     (DP = 0)
or
LTD     *         If current auxiliary register contains 24.
```

	Before Instruction		After Instruction
Data Memory 24	62h	Data Memory 24	62h
Data Memory 25	0h	Data Memory 25	62h
T	3h	T	62h
P	0Fh	P	0Fh
ACC	5h	ACC	14h

Syntax
Direct: [<label>] MAR <dma>
Indirect: [<label>] MAR {*|*+|*-}[,<next ARP>]

Operands $0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution (PC) + 1 → PC
Modifies AR(ARP), ARP as specified by the indirect addressing field (acts as a NOP in direct addressing).

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	1	0	0	0	0	Data Memory Address						
Indirect:	0	1	1	0	1	0	0	0	1	See Section 4.1						

Description In the indirect addressing mode, the auxiliary registers are either incremented or decremented and the ARP is modified; however, no use is made of the memory being referenced. MAR is used only to modify the auxiliary registers or the ARP. ARP may also be loaded by an LST instruction.

MAR acts as a no-operation (NOP) instruction in the direct addressing mode. Also, the LARP instruction is a subset of MAR (i.e., MAR *,0 performs the same function as LARP 0).

Words 1

Cycles 1

Example 1

```
MAR   *,1   Load the ARP with 1.
```

	Before Instruction		After Instruction
ARP	0h	ARP	1h

Example 2

```
MAR   *-    Decrement current auxiliary register (in this
            case, AR1)
```

	Before Instruction		After Instruction
AR1	35h	AR1	34h

Example 3

```
MAR  *+,0  Increment current auxiliary register (AR1) and
           load ARP with 0.
```

	Before Instruction		After Instruction
AR1	34h	AR1	35h
ARP	1h	ARP	0h

Syntax

Direct: [<label>] MPY <dma>
Indirect: [<label>] MPY {*|*+|*-}[,<next ARP>]

Operands $0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution (PC) + 1 → PC
(T register) x (dma) → P register

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	1	1	0	1	0	Data Memory Address						
Indirect:	0	1	1	0	1	1	0	1	1	See Section 4.1						

Description The contents of the T register are multiplied by the contents of the addressed data memory location. The result is placed in the P register.

During an interrupt, all registers except the P register can be saved and restored directly. However, the first-generation TMS320 devices have hardware protection against servicing an interrupt between an MPY or MPYK instruction and the following instruction. For this reason, it is advisable to follow MPY and MPYK with LTA, LTD, PAC, APAC, or SPAC.

Note that no provisions are made for the condition of 8000h x 8000h. If this condition arises, the product will be C0000000h.

Words 1

Cycles 1

Example 1

```
MPY    DAT13    (DP = 0)
```
or
```
MPY    *        If current auxiliary register contains 13.
```

	Before Instruction		After Instruction
Data Memory 13	7h	Data Memory 13	7h
T	6h	T	6h
P	36h	P	2Ah

MPYK — Multiply Immediate

Syntax [<label>] MPYK <constant>

Operands $-2^{12} \leq \text{constant} < 2^{12}$

Execution (PC) + 1 → PC
(T register) x constant → P register

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	13-Bit Constant												

Description The contents of the T register are multiplied by the signed 13-bit constant. The result is loaded into the P register.

During an interrupt, all registers except the P register can be saved and restored directly. Since no provision is made to save the contents of the P register during an interrupt, the MPYK instruction should be followed by one of the following instructions: PAC, APAC, SPAC, LTA, or LTD. Provision is made in hardware to inhibit interrupt during MPYK until the next instruction is executed.

Words 1

Cycles 1

Example

```
MPYK    -9
```

	Before Instruction		After Instruction
T	7h	T	7h
P	2Ah	P	0FFFFFFC1h

No Operation NOP

Syntax [<label>] NOP

Operands None

Execution (PC) + 1 → PC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0

Description No operation is performed. NOP affects only the PC.

NOP is useful as a pad or temporary instruction during program development.

Words 1

Cycles 1

Example

```
NOP
```

Syntax

Direct: [<label>] OR <dma>
Indirect: [<label>] OR {*|*+|*-}[,<next ARP>]

Operands $0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution (PC) + 1 → PC
(ACC(15-0)) .OR.dma → ACC(15-0)
(ACC(31-16)) → ACC(31-16)

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	1	1	0	1	0	0	Data Memory Address						
Indirect:	0	1	1	1	1	0	1	0	1	See Section 4.1						

Description The low-order bits of the accumulator are ORed with the contents of the addressed data memory location. The high-order bits of the accumulator are ORed with all zeroes. Therefore, the upper half of the accumulator is unaffected by this instruction. The result is stored in the accumulator.

The OR instruction is useful for comparing selected bits of a data word.

Words 1

Cycles 1

Example

```
OR     DAT88    (DP = 0)
or
OR     *        Where current auxiliary register contains 88.
```

	Before Instruction		After Instruction
Data Memory 88	0F000h	Data Memory 88	0F000h
ACC	100002h	ACC	10F002h

Output Data to Port OUT

Syntax

Direct: [<label>] OUT <dma>,<PA>
Indirect: [<label>] OUT {*|*+|*-},<PA>[,<next ARP>]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1
$0 \leq$ port address PA ≤ 7

Execution

(PC) + 1 → PC
Port address PA → address bus A2/PA2-A0/PA0
1 → address bus A11-A3
(dma) → data bus D15-D0

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	0	0	1	Port Address			0	Data Memory Address						
Indirect:	0	1	0	0	1	Port Address			1	See Section 4.1						

Description

The OUT instruction transfers data from data memory to an external peripheral. The first cycle of this instruction places the port address onto address lines A2/PA2-A0/PA0. During the same cycle, $\overline{\text{WE}}$ goes low and the data word is placed on the data bus D15-D0. The upper address lines A11 - A3 are held high.

Words

1

Cycles

2

Example

```
OUT    120,7     Output data word stored in data memory
                 location 120 to peripheral on port
                 address 7.
OUT    *,5       Output data word referenced by current
                 auxiliary register to peripheral on port
                 address 5.
```

PAC — Load Accumulator with P Register

Syntax [<label>] PAC

Operands None

Execution (PC) + 1 → PC
(P register) → ACC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0

Description The contents of the P register resulting from a multiply are loaded into the accumulator.

Words 1

Cycles 1

Example

```
PAC
```

	Before Instruction		After Instruction
P	144h	P	144h
ACC	23h	ACC	144h

Pop Top of Stack to Low Accumulator — POP

Syntax [<label>] POP

Operands None

Execution
(PC) + 1 → PC
(TOS) → ACC(11-0)
0 → ACC(31-12)
Pop stack one level.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1

Description The contents of the top of the stack (TOS) are copied to the low accumulator, and the stack popped after the contents are copied. The next element on the stack becomes the top of the stack. The upper bits (31-12) of the accumulator are zeroed. The hardware stack is a last-in, first-out stack with four locations. Any time a pop occurs, every stack value is copied to the next higher stack location, and the top value is removed from the stack. After a pop, the bottom two stack words will have the same value. Because each stack value is copied, if more than three pops (due to POP or RET instructions) occur before any pushes occur, all levels of the stack contain the same value.

Words 1

Cycles 2

Example `POP`

	Before Instruction		After Instruction
ACC	82h	ACC	45h
Stack	45h 16h 7h 33h	Stack	16h 7h 33h 33h

Syntax [<label>] PUSH

Operands None

Execution (PC) + 1 → PC
Push all stack locations down one level.
(ACC(11-0)) → TOS

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0

Description The contents of the lower 12 bits (11-0) of the accumulator are copied onto the top of the hardware stack. The stack is pushed down before the accumulator value is copied. The hardware stack is a last-in, first-out stack with four locations. If more than four pushes (due to CALA, CALL, PUSH, TBLR, or TBLW instructions or interrupts) occur before a pop, the first data values written will be lost with each succeeding push.

Words 1

Cycles 2

Example `PUSH`

	Before Instruction		After Instruction
ACC	7h	ACC	7h
Stack	2h 5h 3h 0h	Stack	7h 2h 5h 3h

Syntax [<label>] RET

Operands None

Execution (TOS) → PC
Pop stack one level.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1

Description The contents of the top of stack are copied into the program counter. The stack is then popped one level. RET is used in conjunction with CALA and CALL for subroutines and interrupts.

Words 1

Cycles 2

Example

```
RET
```

	Before Instruction		After Instruction
PC	96h	PC	37h
Stack	37h 45h 75h 75h	Stack	45h 75h 75h 75h

Syntax [<label>] ROVM

Operands None

Execution (PC) + 1 → PC
0 → OVM status bit
Affects OVM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	0

Description The OVM status bit is reset to logic zero. This disables the overflow mode, in which the device was placed by the SOVM instruction. If an overflow occurs with OVM reset, the OV (overflow flag) is set, and the overflowed result is placed in the accumulator. OVM may also be loaded by the LST and SOVM instructions (see the SOVM instruction).

Words 1

Cycles 1

Example

```
ROVM                The overflow mode bit OVM is reset,
                    disabling the overflow mode on any
                    subsequent arithmetic operations.
```

Store High Accumulator with Shift — SACH

Syntax

Direct: [<label>] SACH <dma>[,<shift>]

Indirect: [<label>] SACH {*|*+|*-}[,<shift>[,<next ARP>]]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1
shift = 0, 1, or 4

Execution

(PC) + 1 → PC
16 MSBs of (ACC) x 2^{shift} → dma

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	0	1	1	Shift			0	Data Memory Address						
Indirect:	0	1	0	1	1	Shift			1	See Section 4.1						

Description

The SACH instruction copies the entire accumulator into a shifter. It then left-shifts this entire 32-bit number 0, 1, or 4 bits, and copies the upper 16 bits of the shifted value into data memory. The accumulator itself remains unaffected.

Words 1

Cycles 1

Example

```
SACH    DAT70,1   (DP = 0)
or
SACH    *,1       If current auxiliary register contains 70.
```

	Before Instruction		After Instruction
ACC	4208001h	ACC	4208001h
Data Memory 70	0h	Data Memory 70	841h

Syntax

Direct: [<label>] SACL <dma>
Indirect: [<label>] SACL {*|*+|*-}[,<0>[,<next ARP>]]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1
shift = 0

Execution

(PC) + 1 → PC
(ACC(15-0)) → dma

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	0	1	0	0	0	0	0	Data Memory Address						
Indirect:	0	1	0	1	0	0	0	0	1	See Section 4.1						

Description

The low-order bits of the accumulator are stored in data memory. There is no shift associated with this instruction, although a shift code of zero MUST be specified if the ARP is to be changed.

Words

1

Cycles

1

Example

```
SACL    DAT71     (DP = 0)
```
or
```
SACL    *         If current auxiliary register contains 71.
```

	Before Instruction		After Instruction
Data Memory 71	5h	Data Memory 71	8421h
ACC	7C638421h	ACC	7C638421h

Syntax

Direct: [<label>] SAR <AR>,<dma>
Indirect: [<label>] SAR <AR>,{*|*+|*-}[,<next ARP>]

Operands $0 \leq \text{dma} \leq 127$
auxiliary register AR = 0 or 1
ARP = 0 or 1

Execution (PC) + 1 → PC
(auxiliary register AR) → dma

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	0	1	1	0	0	0	AR	0	Data Memory Address						
Indirect:	0	0	1	1	0	0	0	AR	1	See Section 4.1						

Description The contents of the designated auxiliary register are stored in the addressed data memory location. For more information, see the LAR instruction.

Words 1

Cycles 1

Example

```
SAR   AR0,DAT30   (DP = 0)
or
SAR   AR0,*       If current auxiliary register contains 30.
```

	Before Instruction		After Instruction
AR0	37h	AR0	37h
Data Memory 30	18h	Data Memory 30	37h

Example

```
LARP    AR0
SAR     AR0,*+
```

	Before Instruction		After Instruction
AR0	5h	AR0	6h
Data Memory 5	0h	Data Memory 5	6h

Warning:

Special problems arise when SAR is used to store the current auxiliary register with indirect addressing if auto-increment/decrement is used.

```
LARP   AR0
LARK   AR0,10
SAR    AR0,*+  or  SAR   AR0,*-
```

In this case, SAR AR0,*+ will cause the value 11 to be stored in location 10. SAR AR0,*- will cause the value 9 to be stored in location 10.

Syntax [<label>] SOVM

Operands None

Execution (PC) + 1 → PC
1 → overflow mode (OVM) status bit
Affects OVM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1

Description The OVM status bit is set to logic 1, which enables the overflow (saturation) mode. If an overflow occurs with OVM set, the overflow flag OV is set, and the accumulator is set to the largest representable 32-bit positive (7FFFFFFFh) or negative (80000000h) number according to the direction of overflow. OVM may also be loaded by the LST and ROVM instructions. (See the ROVM instruction for further information.)

Words 1

Cycles 1

Example 1

```
SOVM                The overflow mode bit OVM is set, enabling
                    the overflow mode on any subsequent
                    arithmetic operations.
```

Syntax [<label>] SPAC

Operands None

Execution (PC) + 1 → PC
(ACC) - (P register) → ACC
Affects OV; affected by OVM.

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0

Description The contents of the P register are subtracted from the contents of the accumulator. The result is stored in the accumulator. Note that the P register is always sign-extended.

Words 1

Cycles 1

Example `SPAC`

	Before Instruction		After Instruction
P	24h	P	24h
ACC	3Ch	ACC	18h

Syntax

Direct: [<label>] SST <dma>
Indirect: [<label>] SST {*|*+|*-}[,<next ARP>]

Operands 0 ≤ dma ≤ 127
ARP = 0 or 1

Execution (PC) + 1 → PC
(status register) → specified dma (page 1 only in direct addressing)

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	1	1	1	0	0	0	Data Memory Address						
Indirect:	0	1	1	1	1	1	0	0	1	See Section 4.1						

Description The status bits are saved into the specified data memory address (page 1 only if direct memory addressing is used).

In the direct addressing mode, the status register is always stored in page 1 regardless of the value of the DP register. The processor automatically forces the page to be 1, and the specific location within that page is defined in the instruction. Note that the DP register is not physically modified. This allows storage of the DP register in the data memory on interrupts, etc., in the direct addressing mode without having to change the DP. In the indirect addressing mode, the data memory address is obtained from the auxiliary register selected. (See the LST instruction for more information.)

The SST instruction can be used to store the status bits after interrupts and subroutine calls. These status bits include the OV (overflow flag) bit, OVM (overflow mode) bit, INTM (interrupt mode) bit, ARP (auxiliary register pointer) bit, and DP (data memory page pointer) bit. The status bits are stored in the data memory word as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OV	OVM	INTM	1	1	1	1	ARP	1	1	1	1	1	1	X	DP

X = reserved

Words 1

Cycles 1

Example

```
SST   DAT1    (DP = don't care) or
SST   *,1     If current auxiliary register contains 1.
```

	Before Instruction		After Instruction
Status Register	5EFEh	Status Register	5EFEh
Data Memory 1	Ah	Data Memory 1	5EFEh

Syntax

Direct: [<label>] SUB <dma>[,<shift>]
Indirect: [<label>] SUB {*|*+|*-}[,<shift>[,<next ARP>]]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1
$0 \leq \text{shift} \leq 15$ (defaults to 0)

Execution

$(\text{PC}) + 1 \rightarrow \text{PC}$
$(\text{ACC}) - [(\text{dma}) \times 2^{\text{shift}}] \rightarrow \text{ACC}$
Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	0	0	1	Shift				0	Data Memory Address						
Indirect:	0	0	0	1	Shift				1	See Section 4.1						

Description

The contents of the addressed data memory location are left-shifted and subtracted from the accumulator. During shifting, the low-order bits are zero-filled. The high-order bit is sign-extended. The result is stored in the accumulator.

Words 1

Cycles 1

Example

```
SUB     DAT59      (DP = 0)
or
SUB     *          If current auxiliary register contains 59.
```

	Before Instruction		After Instruction
ACC	24h	ACC	13h
Data Memory 59	11h	Data Memory 59	11h

Syntax

Direct: [<label>] SUBC <dma>
Indirect: [<label>] SUBC {*|*+|*-}[,<next ARP>]

Operands $0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution
(PC) + 1 → PC
(ACC) - [(dma) x 2^{15}] → ALU output
If ALU output ≥ 0:
 Then (ALU output) x 2 + 1 → ACC;
 Else (ACC) x 2 → ACC.
Affects OV but NOT affected by OVM (no saturation).

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	1	0	0	0	Data Memory Address						
Indirect:	0	1	1	0	0	1	0	0	1	See Section 4.1						

Description The SUBC instruction performs conditional subtraction, which may be used for division. The 16-bit dividend is placed in the low accumulator, and the high accumulator is zeroed. The divisor is in data memory. SUBC is executed 16 times for 16-bit division. After completion of the last SUBC, the quotient of the division is in the lower-order 16-bit field of the accumulator, and the remainder is in the high-order 16 bits of the accumulator. SUBC assumes the divisor and the dividend are both positive.

If the 16-bit dividend contains less than 16 significant bits, the dividend may be placed in the accumulator left-shifted by the number of leading non-significant zeroes. The number of executions of SUBC is reduced from 16 by that number. However, at least one leading zero must always be present since both operands of the SUBC instruction must be positive. Note that the next instruction after SUBC cannot use the accumulator.

The SUBC instruction affects OV but is not affected by OVM. Therefore, the accumulator does not saturate upon positive or negative overflows when executing this instruction.

The above description is for 16-bit integer division. SUBC can also be used in fixed-point division.

Words 1

Cycles 1

Example

```
        LARP    AR0
        LARK    AR0,15
DIV     SUBC    DAT2        (DP = 0)
        BANZ    DIV
```

	Before Instruction		After Instruction
Data Memory 2	7h	Data Memory 2	7h
ACC	41h	ACC	20009h

The results above show the execution of all the instructions in the code example.

Syntax

Direct: [<label>] SUBH <dma>
Indirect: [<label>] SUBH {*|*+|*-}[,<next ARP>]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution

$(\text{PC}) + 1 \rightarrow \text{PC}$
$(\text{ACC}) - [(\text{dma}) \times 2^{16}] \rightarrow \text{ACC}$
Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6 5 4 3 2 1 0
Direct:	0	1	1	0	0	0	1	0	0	Data Memory Address
Indirect:	0	1	1	0	0	0	1	0	1	See Section 4.1

Description

The contents of the addressed data memory location are subtracted from the upper 16 bits of the accumulator. The 16 low-order bits of the accumulator are unaffected. The result is stored in the accumulator.

The SUBH instruction can be used for performing 32-bit arithmetic.

Words 1

Cycles 1

Example

```
SUBH    DAT33     (DP = 0)
```
or
```
SUBH    *         If current auxiliary register contains 33.
```

	Before Instruction		After Instruction
Data Memory 33	4h	Data Memory 33	4h
ACC	0A0013h	ACC	60013h

Subtract from Low Accumulator with Sign-Extension Suppressed

Syntax

Direct: [<label>] SUBS <dma>
Indirect: [<label>] SUBS {*|*+|*-}[,<next ARP>]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution

(PC) + 1 → PC
(ACC) - (dma) → ACC
Affects OV; affected by OVM.

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	0	1	1	0	Data Memory Address						
Indirect:	0	1	1	0	0	0	1	1	1	See Section 4.1						

Description

The contents of the addressed data memory location are subtracted from the accumulator with sign-extension suppressed. The data is treated as a 16-bit unsigned number, rather than a two's-complement number. The accumulator behaves as a signed number.

Words 1

Cycles 1

Example

```
SUBS    DAT2     (DP = 0)
or
SUBS    *        If current auxiliary register contains 2.
```

	Before Instruction		After Instruction
Data Memory 2	0F003h	Data Memory 2	0F003h

	Before Instruction		After Instruction
ACC	0F105h	ACC	102h

Syntax

Direct: [<label>] TBLR <dma>
Indirect: [<label>] TBLR {*|*+|*-}[,<next ARP>]

Operands $0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution
(PC) + 1 → TOS
(ACC(11-0)) → PC
(pma) → dma
Modify AR(ARP) and ARP as specified
(TOS) → PC

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	1	1	1	0	Data Memory Address						
Indirect:	0	1	1	0	0	1	1	1	1	See Section 4.1						

Description The TBLR instruction transfers a word from a location in program memory to a data memory location specified by the instruction. The program memory address is defined by the low-order 12 bits of the accumulator. For this operation, a read from program memory is performed, followed by a write to data memory. The contents of the lowest stack location are lost when using TBLW.

The TBLR instruction is useful for reading coefficients that have have been stored in program ROM, or time-dependent data stored in RAM.

Words 1

Cycles 3

Example 1

```
TBLR    DAT6    (DP = 0)
TBLR    *       If current auxiliary register contains 6.
```

	Before Instruction		After Instruction
ACC	9h	ACC	9h
Program Memory 9	306h	Program Memory 9	306h
Data Memory 6	75h	Data Memory 6	306h
Stack	71h 48h 16h 80h	Stack	71h 48h 16h 16h

Syntax

Direct: [<label>] TBLW <dma>
Indirect: [<label>] TBLW {*|*+|*-}[,<next ARP>]

Operands $0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution
(PC) + 1 → TOS
(ACC(11-0)) → PC
(dma) → pma
Modify AR(ARP) and ARP as specified
(TOS) → PC

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	1	1	1	0	1	0	Data Memory Address						
Indirect:	0	1	1	1	1	1	0	1	1	See Section 4.1						

Description The TBLW instruction transfers a word in data memory to program memory. The data memory address is specified by the instruction, and the program memory address is specified by the lower 12 bits of the accumulator. A read from data memory is followed by a write to program memory to complete the instruction. The contents of the lowest stack location are lost when using TBLW.

Words 1

Cycles 3

Example

```
TBLW    DAT5      (DP = 0)
TBLW    *         If current auxiliary register contains 5.
```

	Before Instruction		After Instruction
Data Memory 5	4339h	Data Memory 5	4339h
Program Memory 8	306h	Program Memory 8	4339h
ACC	8h	ACC	8h
Stack	34h 23h 11h 97h	Stack	34h 23h 11h 11h

Syntax

Direct: [<label>] XOR <dma>
Indirect: [<label>] XOR {*|*+|*-}[,<next ARP>]

Operands $0 \leq dma \leq 127$
ARP = 0 or 1

Execution (PC) + 1 → PC
(ACC(15-0)).XOR.dma → ACC(15-0)
(ACC(31-16)) → ACC(31-16)

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	1	1	0	0	0	0	Data Memory Address						
Indirect:	0	1	1	1	1	0	0	0	1	See Section 4.1						

Description The low half of the accumulator is exclusive-ORed with the contents of the addressed data memory location. The upper half of the accumulator is not affected by this instruction.

The XOR instruction is useful for toggling or setting bits of a word for high-speed control. In addition, the one's complement of a word can be found by exclusive-ORing it with all ones.

Words 1

Cycles 1

Example

```
XOR     DAT127   (DP = 0)
or
XOR     *        If current auxiliary register contains 127.
```

	Before Instruction		After Instruction
Data Memory 127	0F0F0h	Data Memory 127	0F0F0h
ACC	12345678h	ACC	1234A688h

Syntax	[<label>] ZAC
Operands	None
Execution	(PC) + 1 → PC 0 → ACC

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	1

Description	The contents of the accumulator are replaced with zero.
Words	1
Cycles	1
Example	`ZAC`

	Before Instruction		After Instruction
ACC	0A5A5A5A5h	ACC	0h

Zero Low Accumulator and Load High Accumulator

Syntax

Direct: [<label>] ZALH <dma>
Indirect: [<label>] ZALH {*|*+|*-}[,<next ARP>]

Operands

0 ≤ dma ≤ 127
ARP = 0 or 1

Execution

(PC) + 1 → PC
0 → ACC(15-0)
(dma) → ACC(31-16)

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	1	0	1	0	Data Memory Address						
Indirect:	0	1	1	0	0	1	0	1	1	See Section 4.1						

Description

ZALH loads a data memory value into the high-order half of the accumulator. The low-order bits of the accumulator are zeroed.

ZALH is useful for 32-bit arithmetic operations.

Words 1

Cycles 1

Example

```
ZALH     DAT3     (DP = 0)
```
or
```
ZALH     *        If current auxiliary register contains 3.
```

	Before Instruction		After Instruction
Data Memory 3	3F01h	Data Memory 3	3F01h
ACC	77FFFFh	ACC	3F010000h

Zero Accumulator, Load Low Accumulator with Sign-Extension Suppressed

Syntax

Direct: [<label>] ZALS <dma>
Indirect: [<label>] ZALS {*|*+|*-}[,<next ARP>]

Operands

$0 \leq \text{dma} \leq 127$
ARP = 0 or 1

Execution

(PC) + 1 → PC
0 → ACC(31-16)
(dma) → ACC(15-0)

Encoding

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Direct:	0	1	1	0	0	1	1	0	0	Data Memory Address						
Indirect:	0	1	1	0	0	1	1	0	1	See Section 4.1						

Description

The contents of the addressed data memory location are loaded into the 16 low-order bits of the accumulator. The upper half of the accumulator is zeroed. The data is treated as a 16-bit unsigned number rather than a two's-complement number. Therefore, there is no sign-extension with this instruction.

ZALS is useful for 32-bit arithmetic operations.

Words 1

Cycles 1

Example

```
ZALS    DAT1    (DP = 0)
or
ZALS    *       If current auxiliary register contains 1.
```

	Before Instruction		After Instruction
Data Memory 1	0F7FFh	Data Memory 1	0F7FFh
ACC	7FF00033h	ACC	0F7FFh

Section 5
Software Applications

This section provides examples of how to use the various architectural and instruction set features of the TMS320C14/E14. For more information about specific applications, refer to the book *Digital Signal Processing Applications with the TMS320 Family* (SPRA012A).

Major topics discussed in this section include:

5.1 Processor Initialization

It is necessary to initilize the processor prior to execution of a DSP algorithm. Initilization is normally done following a reset so that the processor and peripherals will meet the requirements of the system. In initializing the TMS320C14/E14, the system requirements of the following functions should be considered:

- Auxiliary register pointer
- Data memory page pointer
- Overflow mode
- Interrupt structure
- Bit I/O
- Serial port
- Watchdog timer
- General purpose timer
- Event manager (compare/capture)

Note that although a hardware reset forces the program counter to zero and initializes various registers and status bits, the overflow mode bit (OVM), interrupt mode bit (INTM), auxiliary register pointer (ARP), and data memory page pointer bit (DP) must be initialized by the programmer.

As described in Section 3.1.4, the on-chip peripherals are accessed with a bank address and a port address. Table 5-1 shows how the I/O registers are mapped.

Table 5-1. I/O Register Map

PORT	BANK0	BANK1	BANK2	BANK3	BANK4	BANK5	BANK6	BANK7	BANKFFFF
0	IOP	WDT	TMR1	CMPR0	ACT0	SCON	FIFO0		EXT. I/O
1	DDR	WPER	TPR1	CMPR1	ACT1	SSET	FIFO1		EXT. I/O
2	BSET	WTPL4	TMR2	CMPR2	ACT2	SCLR	FIFO2	SMAT	EXT. I/O
3	BCLR	SYSCON	TPR2	CMPR3	ACT3	TBR	FIFO3	TSR	EXT. I/O
4	IF		TCON	CMPR4	ACT4	RBR	CCON	RSR	EXT. I/O
5	IM			CMPR5	ACT5	SBRG	CCLR	STMR	EXT. I/O
6	FCLR								EXT. I/O
7	BSR	BSR	BSR	BSR	BSR	BSR	BSR	BSR	BSR

The assembly language code presented here initializes the TMS320C14/E14 to the following machine state:

- Overflow mode disabled
- Data memory page pointer set to zero
- Auxiliary register pointer set to zero
- Internal memory filled with zero
- Bit I/O
  - Bits 15 through 8 configured as inputs
  - Bits 7 through 0 configured as outputs
- Serial port configured as:
  - Asynchronous mode
  - Seven data bits
  - Even parity
  - 19.2 kbps bit rate
- Watchdog timer set to time out once every 1 ms.
- Use compare subsystem and set compare data to a specific pattern. If match occurs, output pin is toggled appropriately as defined by action register.
- Interrupt enabled
- Timer 1 and 2 used by the compare subsystem are initialized to 400 μs.

5.1.1 Header File Defining Constants

The following header file (Example 5-1), is included in each of the following program listing example in this section. It defines constants used for symbolic access to registers and bit field maps of control registers. The following example listings refer to the header file simply as "C14INC".

Example 5-1. Header File Constants

```
*
                .title  'HEADER FILE CONSTANTS'
*
*               FILE NAME: C14INC
*
*               THIS HEADER FILE DEFINES CONSTANTS FOR SYMBOLIC ACCESS
*               TO REGISTERS AND BIT MAP FIELDS OF CONTROL REGISTERS
*
*
*               DEFINE CONSTANTS FOR BITS
*
BIT0            .set    0000000000000001B
BIT1            .set    0000000000000010B
BIT2            .set    0000000000000100B
BIT3            .set    0000000000001000B
BIT4            .set    0000000000010000B
BIT5            .set    0000000000100000B
BIT6            .set    0000000001000000B
BIT7            .set    0000000010000000B
BIT8            .set    0000000100000000B
BIT9            .set    0000001000000000B
BIT10           .set    0000010000000000B
BIT11           .set    0000100000000000B
BIT12           .set    0001000000000000B
BIT13           .set    0010000000000000B
BIT14           .set    0100000000000000B
BIT15           .set    1000000000000000B
*
MASK0           .set    1111111111111110B
MASK1           .set    1111111111111101B
MASK2           .set    1111111111111011B
MASK3           .set    1111111111110111B
MASK4           .set    1111111111101111B
MASK5           .set    1111111111011111B
MASK6           .set    1111111110111111B
MASK7           .set    1111111101111111B
MASK8           .set    1111111011111111B
MASK9           .set    1111110111111111B
MASK10          .set    1111101111111111B
MASK11          .set    1111011111111111B
MASK12          .set    1110111111111111B
MASK13          .set    1101111111111111B
MASK14          .set    1011111111111111B
MASK15          .set    0111111111111111B
*
*
BitIOBank       .set    0
InterruptBank   .set    0
WDTBank         .set    1
SYSBank         .set    1
TimerBank       .set    2
CompareBank     .set    3
ActionBank      .set    4
SerialPort      .set    5
CaptureBank     .set    6
SerialPortRegs  .set    7
*
```

```
ExternalIO       .set     0FFFFH
*
BSR              .set     7
*
*      DEFINE REGISTERS IN BitIOBank
*
IOP              .set     0
DDR              .set     1
BSET             .set     2
BCLR             .set     3
*
*      DEFINE REGISTERS FOR InterruptBank
*
IF               .set     4
IM               .set     5
FCLR             .set     6
*
*      DEFINE REGISTERS IN WDTBank
*
WDT              .set     0
TMR4             .set     0
WPER             .set     1
TPR4             .set     1
WTPL             .set     2
*
*      DEFINE REGISTER IN SYSBank
*
SYSCON           .set     3
*
*      DEFINE REGISTERS IN TimerBank
*
TMR1             .set     0
TPR1             .set     1
TMR2             .set     2
TPR2             .set     3
TCON             .set     4
*
*      DEFINE REGISTERS IN CompareBank
*
CMPR0            .set     0
CMPR1            .set     1
CMPR2            .set     2
CMPR3            .set     3
CMPR4            .set     4
CMPR5            .set     5
*
*      DEFINE REGISTERS IN ActionBank
*
ACT0             .set     0
ACT1             .set     1
ACT2             .set     2
ACT3             .set     3
ACT4             .set     4
ACT5             .set     5
*
```

```
*      DEFINE REGISTERS IN SerialPort
*
SCON             .set    0
SSET             .set    1
SCLR             .set    2
TBR              .set    3
RBR              .set    4
SBRG             .set    5
*
*      DEFINE REGISTERS IN CaptureBank
*
FIFO0            .set    0
FIFO1            .set    1
FIFO2            .set    2
FIFO3            .set    3
CCON             .set    4
CCLR             .set    5
*
*      DEFINE SerialPortRegs
*
SMAT             .set    2
TSR              .set    3
RSR              .set    4
STMR             .set    5
*
*      DEFINE BIT MAP FOR SYSCON
*
MCMPbit          .set    BIT0
*
*      DEFINE BIT MAP FOR STATUS REGISTER
*
_OV              .set    BIT15
_OVM             .set    BIT14
_INTM            .set    BIT13
_ARP             .set    BIT8
_DP              .set    BIT0
*
*      DEFINE BIT MAP FOR INTERRUPT FLAG REGISTER
*
_NMI             .set    BIT15
_INT             .set    BIT14
_STMRINT         .set    BIT12
_CAPINT3         .set    BIT11
_CAPINT2         .set    BIT10
_CAPINT1         .set    BIT9
_CAPINT0         .set    BIT8
_CMPINT1         .set    BIT7
_CMPINT0         .set    BIT6
_TIMINT2         .set    BIT5
_TIMINT1         .set    BIT4
_RXINT           .set    BIT3
_TXINT           .set    BIT2
_WDTINT          .set    BIT1
_IOPINT          .set    BIT0
*
```

```
*
      DEFINE SHIFTS FOR INTERRUPT BIT MAP
*
NMI_BIT         .set    15
INT_BIT         .set    14
STMRINT_BIT     .set    12
CAPINT3_BIT     .set    11
CAPINT2_BIT     .set    10
CAPINT1_BIT     .set    9
CAPINT0_BIT     .set    8
CAPINT1_BIT     .set    7
CMPINT0_BIT     .set    6
TIMINT2_BIT     .set    5
TIMINT1_BIT     .set    4
RXINT_BIT       .set    3
TXINT_BIT       .set    2
WDTINT_BIT      .set    1
IOPINT_BIT      .set    0
*
*     DEFINE BIT MAP FOR TCON
                            TCON BIT
*
CAPIntOnFirst   .set    BIT15     ; 15
CAPIntOnThird   .set    0
*
CaptureEnable   .set    0         ; 14
CaptureDisable  .set    BIT14
*
CAPTMR2Select   .set    BIT13     ; 13
CAPTMR1Select   .set    0
*
CMP5Enable      .set    Bit12     ; 12
CAP3Enable      .set    0
*
CMP4Enable      .set    BIT11     ; 11
CAP2Enable      .set    0
*
CMPEnable       .set    BIT10     ; 10
*
CMPTMR2Select   .set    BIT9      ; 9
CMPTMR1Select   .set    0
*
CMPPWMMode      .set    BIT8      ; 8
CMPNormalMode   .set    0
*
TMR2Enable      .set    BIT6      ; 7,6 TIMER 2 MODE FIELD
*
TMR2Internal    .set    0         ; 5,4 TIMER 2 SOURCE
TMR2External    .set    BIT4+BIT5
TMR2fromTMR1    .set    BIT5
*
TMR1Enable      .set    BIT1      ; 2,1 TIMER 1 MODE FIELD
*
TMR1Internal    .set    0         ; 0 TIMER 1 SOURCE
TMR1External    .set    1
*
```

```
*
*    TIMER 1 AND 2 MODES
TMRStop         .set    0
TMRby4          .set    1
TMRby16         .set    2
TMRby1          .set    3
*
*     BIT MAP FOR ACTION REGISTER
*
*    ACTION CODES
*
CMPNoaction     .set    0
CMPReset        .set    1
CMPSet          .set    2
CMPToggle       .set    3
*
*    COMPARE OUTPUT ACTION FIELDS
*
CMP0Act         .set    BIT14
CMP1Act         .set    BIT12
CMP2Act         .set    BIT10
CMP3Act         .set    BIT8
CMP4Act         .set    BIT6
CMP5Act         .set    BIT4
*
*    COMPARE INTERRUPT ENABLES
*
CMPINT1Enable   .set    BIT3
CMPINT2Enable   .set    BIT2
ActionDefault   .set    BIT1+BIT0
*
*     BIT MAP FOR CCON
*
*     CAPTURE CONTROL BIT FIELD DEFINITIONS
*
*     FIFO STATUS BITS
*
FIFOFull        .set    8
FIFONotEmpty    .set    4
*
*    CAPTURE MODE DEFINITIONS
*
CAPDisable      .set    0
PosEdgeDetect   .set    1
NegEdgeDetect   .set    2
PosNegDetect    .set    3
*
*    CAPTURE CONTROL FIELDS
*
CAP3Mode        .set    BIT12
CAP2Mode        .set    BIT8
CAP1Mode        .set    BIT4
CAP0Mode        .set    BIT0
*
```

```
*
*      BIT MAP FOR SCON
                                  SCON BIT
*
*
SynchMode        .set    BIT0     ; 0
AsynchMode       .set    0
*
ParityEnable     .set    BIT1     ; 1
ParityDisble     .set    0
*
OddParity        .set    BIT2     ; 2
EvenParity       .set    0
*
ParityError      .set    BIT3     ; 3
*
FERR             .set    BIT4     ; 4
*
*     NUMBER OF DATA BITS SELECT FIELD
*
SixDataBits      .set    0        ; 5,6
SevenDataBits    .set    BIT5
EightDataBits    .set    BIT6
NineDataBits     .set    BIT5+BIT6
*
RxOverflow       .set    BIT7     ; 7
*
RxFull           .set    BIT8     ; 8
*
*     CODEC MODE FIELD
*           SEE TABLE 3-19 FOR DEFINITIONS
*
CodecMode        .set    BIT9     ; 9,10,11,12,13
*
CodecEnable      .set    BIT14    ; 14
*
TxEmpty          .set    BIT15    ; 15
*
*     EQUATES FOR BAUD RATE WITH 25.6 MHZ CLOCK
*
SBRG19200        .set    20
SBRG9600         .set    40
SBRG4800         .set    82
SBRG2400         .set    166
SBRG1200         .set    332
SBRG300          .set    1332
*
*      EQUATES FOR WDT RESET
*
WDTmagic1        .set    0ABCDh
WDTmagic2        .set    2345h
```

Example 5-2. TMS320C14/E14 Processor Initialization

```
*
*     INITIALIZATION FOR THE C14/E14
*
               .title 'C14/E14 PROCESSOR INITIALIZATION'
*
               .include "C14INC"  ; INCLUDE HEADER FILE
*
WDT1mS         .set   782         ; COUNTS FOR 1 MS WITH MAX CLOCK RATE
TMR2Period     .set   3125        ; PERIOD FOR TIMER2 (400 MICROSEC)
IOPortData     .set   0A5h
InterruptMask  .set   0FFFCh      ; MASK FOR IM TO ENABLE WDT
*                                 ; AND IOP INTERRUPTS
*
SerialPortMode .set   AsynchMode+SevenDataBits+ParityEnable+EvenParity
CaptureMode    .set   CAPIntOnFirst+CAPTMR2Select
CompareMode    .set   CMP5Enable+CMPEnable
Timer2Mode     .set   TMR2Enable*TMRby1+TMR2Internal
Timer1Mode     .set   TMR1Enable*TMRStop
ActionMode     .set   CMP5Act*CMPToggle+ActionDefault
CaptureControl .set   PosNegDetect*CAP2Mode
*
               .bss   ONE,1
               .bss   TMP,1
               .bss   BANK,1
               .bss   REG,1
               .bss   VALUE,1
               .bss   WDTVAL1,1
               .bss   WDTVAL2,1
               .bss   INTMSK,1
*
               .ref   ISR
*
*   RESET AND INTERRUPT BRANCH INSTRUCTIONS
*
               .text
*
               B      INIT
               B      ISR
*
*   PLACE INIT TABLE IN INITIALIZED DATA SPACE
               .data
*
InitTable:
             .word WDT1mS
             .word WDTmagic1
             .word WDTmagic2
             .word SerialPortMode ; SERIAL PORT MODE
             .word SBRG19200
             .word CaptureMode+CompareMode+Timer1Mode+Timer2Mode
             .word TMR2Period
             .word InterruptMask
             .word ActionMode
             .word CaptureControl
```

```
*
* BACK TO PROGRAM SPACE
*
           .text
*
INIT:
           ROVM                     ; DISABLE OVERFLOW MODE
           LDPK    0                ; POINT TO DATA PAGE 0
           LARK    AR0,255          ; AR0 = 255
*
* CLEAR DATA RAM
*
           ZAC
           LARP    AR0              ; ARP -> AR0
LOOP:
           SACL    *                ; CLEAR MEMORY POINTED TO BY AR0
           BANZ    LOOP             ; DECREMENT AR0 AND BRANCH TO LOOP
                                    ; IF AR0 <> 0
*
           LACK    1
           SACL    ONE              ; ONE = 1
*
           LT      ONE
           MPYK    InitTable
           PAC                      ; ACC = ->InitTable
                                    ; SINCE LACK ONLY REFERENCES AN 8 BIT
                                    ; CONSTANT AND MPYK REFERENCES A 13 BIT
                                    ; CONSTANT, MULTIPLYING BY 1 AND THEN
                                    ; SAVING THE RESULT IN THE ACC THROUGH
                                    ; THE P REGISTER SOLVES THE PROBLEM OF
                                    ; GETTING A 12 BIT ADDRESS INTO THE ACC
           TBLR    VALUE            ; VALUE = InitTable[0]
           SACL    TMP              ; TMP = ->InitTable[0]
*
           LACK    WDTBank          ; INITIALIZE THE WDT PERIOD REGISTER
           SACL    BANK             ; LOAD WPER WITH COUNTS FOR 1MS
           OUT     BANK,BSR         ; TIMEOUT
           OUT     VALUE,WPER
*
           LAC     TMP
           ADD     ONE
           SACL    TMP              ; TMP  = ->InitTable[1]
           TBLR    WDTVAL1          ; INITIALIZE WDT MAGIC VALUE 1
*
           LAC     TMP
           ADD     ONE
           SACL    TMP              ; TMP  = ->InitTable[2]
           SACL    WDTVAL2          ; INITIALIZE WDT MAGIC VALUE 2
*
           LAC     TMP
           ADD     ONE
           SACL    TMP              ; TMP  = ->InitTable[3]
           TBLR    VALUE            ; INITIALIZE SCON REGISTER
           LACK    SerialPort       ; ASYNCHRONOUS MODE
           SACL    BANK             ; 7 DATA BITS
           OUT     BANK,BSR         ; EVEN PARITY
           OUT     VALUE,SCON
```

```
*
        LAC     TMP
        ADD     ONE
        SACL    TMP             ; TMP = ->InitTable[4]
        TBLR    VALUE           ; SET BAUD RATE GENERATOR FOR 19200 BAUD
        OUT     VALUE,SBRG
*
        LAC     TMP
        ADD     ONE
        SACL    TMP             ; TMP = ->InitTable[5]
        TBLR    VALUE           ; INITIALIZE TCON
        LACK    TimerBank       ; ENABLE CAPTURE, INTERRUPT ON FIRST INPUT
        SACL    BANK            ; COMPARE 5: TOGGLE MODE
        OUT     BANK,BSR        ; TIMER 1 STOP
        OUT     VALUE,TCON      ; TIMER 2 DIVIDE BY 1, INTERNAL CLOCK
*
        LAC     TMP
        ADD     ONE
        SACL    TMP             ; TMP = ->InitTable[6]
        TBLR    VALUE           ; INITIALIZE TMR2 PERIOD
        OUT     VALUE,TPR2      ; LOAD WITH COUNTS FOR 400 MICROSEC PERIOD
                                ; WITH 25.6 MHZ CLOCK INPUT
*
        LAC     TMP
        ADD     ONE
        SACL    TMP             ; TMP  = ->InitTable[7]
        TBLR    VALUE           ; INITIALIZE INTERRUPT MASK
        LACK    InterruptBank; ENABLE WDT AND IOP INTERRUPTS
        SACL    BANK
        OUT     BANK,BSR
        OUT     VALUE,IM
*
        LAC     TMP
        ADD     ONE
        SACL    TMP             ; TMP  = ->InitTable[8]
        TBLR    VALUE           ; INITIALIZE ACTION MODE FOR CMP5
        LACK    ActionBank      ; TOGGLE COMPARE OUTPUT 5
        SACL    BANK
        OUT     BANK,BSR
        OUT     VALUE,ACT5
*
        LAC     TMP
        ADD     ONE
        SACL    TMP             ; TMP = ->InitTable[9]
        TBLR    VALUE           ; INITIALIZE CAPTURE CONTROL
        LACK    CaptureBank     ; DETECT RISING OR FALLING EDGES ON CAP2
        SACL    BANK
        OUT     BANK,BSR
        OUT     VALUE,CCON
```

```
*
*               INITIALIZE DDR
*
*               CONFIGURE BITS 0-7 OF IOP AS OUTPUTS AND
*               8-15 OF IOP AS INPUTS
*
                LACK    BitIOBank
                SACL    BANK
                OUT     BANK,BSR
                LACK    11111111B
                SACL    VALUE
                OUT     VALUE,DDR
*
*               THIS COMPLETES DEVICE INITIALIZATION
*
                EINT            ; ENABLE INTERRUPTS
*
                END
```

5.2 Interrupt Management

The interrupt function allows current CPU processing to be suspended in order to perform a more critical function. The TMS320C14/E14 provides a total of 15 external and internal interrupts. Two interrupts are dedicated for external sources. The remaining interrupts are used to service the on-chip peripherals. All interrupts, with the exception $\overline{\text{NMI}}$, are maskable through an interrupt mask register (IM). The interrupts are synchronized and multiplexed into the master interrupt circuitry and have the same priority. Software polling techniques are used to determine which input caused the interrupt.

Processing in the Interrupt Service Routine (ISR) must assure that the processor context is saved before execution and restored when the routine is finished.

Interrupt processing on the TMS320C14/E14 begins as follows:

1) The EINT (enable interrupt) instruction is executed, setting the INTM (interrupt mode) bit to 0 so that the interrupts can be received.
2) When an interrupt occurs, a bit in the Interrupt Flag (IF) register corresponding to that interrupt is set to a 1.
3) If the corresponding interrupt mask (IM) bit is zero, the CPU interrupt is generated.

As an interrupt is generated (either by an internal or external source) the following events occur automatically:

1) The INTM bit is set to 1 to disable further interrupts.

2) The current PC is pushed onto the top of stack (TOS).

3) The new PC is set to 2.

During the servicing of the interrupt, the following operations are commonly performed by the user in software:

1) Program memory address 2 will either have a service routine to save the context of the machine or branch to the interrupt service routine.

2) The interrupt service routine is executed. The context of the machine may be stored and then restored later if required. The following can be used to select which interrupt to service:

   a) Use software polling techniques to determine which one of the 15 flags has been set in the control register.
   b) Check for corresponding mask bits before proceeding (optional).
   c) Clear the flag (reset to 0) through the FCLR register and service the source of the flag.

3) The EINT instruction is executed, clearing the INTM bit to 0.

4) The RET instruction is executed.

Although all interrupts have the same hardware priority, the user can control the polling of the interrupt flags. The ISR should clear the interrupt flag before executing the EINT instruction or enabling interrupts. Note that clearing the flag register requires writing a one to the FCLR register. Writing a zero has no effect. The following interrupt service routine example is for a system with five active interrupts, and includes polling.

Example 5-3. Interrupt Service Routine

```
*
*
*   THIS IS AN EXAMPLE INTERRUPT SERVICE ROUTINE
*
*   THIS ROUTINE MAY BE LOCATED AT LOCATION 2 TO BE INVOKED THROUGH
*   A BRANCH LOCATED AT LOCATION 2.  THIS MODULE IS DESIGNED AS A
*   DISPATCHER FOR THE VARIOUS SERVICE ROUTINES THAT WOULD BE REQUIRED
*   TO IMPLEMENT THE DESIRED RESPONSES TO SYSTEM INTERRUPTS.
*
      .include "C14INC"           ; INCLUDE HEADER FILE
*
      .def    ISR,ISRexit
*
      .ref    Rxisr,Txisr,WDTisr,NMIisr,INTisr,IOPisr
      .ref    ONE                 ; INITIALIZED TO 1
*
      .ref    BANK                ; THESE 7 LOCAL VARIABLES SHOULD BE
      .ref    STATUS              ; PLACED ON PAGE 1
      .ref    ACCL
      .ref    ACCH
      .ref    BankSave
      .ref    TMP
      .ref    IFimage
*
      .text
*
ISR:
*
SAVE ENVIRONMENT
*
      SST     STATUS              ; SAVE STATUS ON DATA PAGE 1
      LDPK    1                   ; DP NOW POINTS TO DATA PAGE 1
      SACL    ACCL                ; SAVE ACCUMULATOR
      SACH    ACCH
      IN      BankSave,BSR        ; SAVE BSR
*
*   BRANCH TO THE APPROPRIATE INTERRUPT SERVICE ROUTINE BASED
*   ON BIT MAP OF IF
*
      OUT     InterruptBank,BSR   ; GET INTERRUPT FLAGS
      IN      IFimage,IF          ; STORE IN IFimage
      LAC     ONE,15              ; TMP = 7FFFh
      SUB     ONE
      SACL    TMP
      OUT     TMP,FCLR            ; CLEAR ALL INTERRUPTS
*
*   TEST FOR INTERRUPTS FROM:
*                           NMI
*                           TRANSMITTER AND RECEIVE SECTIONS
*                              OF SERIAL PORT
*                           WDT
*                           BIT I/O PORT
*                           INT PIN
*


*          IGNORE ALL OTHER INTERRUPTS
*
           LAC     ONE,NMI_BIT
           LAC     ONE,NMI_BIT
           AND     IFimage
           BZ      SkipNMI          ; IF (NMI) NMIisr()
           CALL    NMIisr
SkipNMI:
           LAC     ONE,RXINT_BIT
           AND     IFimage
           BZ      SkipRx           ; IF (RECEIVE PORT INTERRUPT) Rxisr()
           CALL    Rxisr
SkipRx:
           LAC     ONE,TXINT_BIT
           AND     IFimage
           BZ      SkipTx           ; IF (TRANSMIT PORT INTERRUPT) Txisr()
           CALL    Txisr
SkipTx:
           LAC     ONE,WDINT_BIT
           AND     IFimage
           BZ      SkipWDT          ; IF (WDT INTERRUPT) WDTisr
           CALL    WDTisr
SkipWDT:
           LAC     ONE,IOPINT_BIT
           AND     IFimage
           BZ      SkipIOP          ; IF (I/O PORT INTERRUPT) IOPisr
           CALL    IOPisr
SkipIOP:
           LAC     ONE,INT_BIT
           AND     IFimage
           BZ      SkipINT          ; IF (INT) INTisr
           CALL    INTisr
SkipINT:
*
*          EXIT INTERRUPT SERVICE ROUTINE
*
ISRexit:
*
*          RESTORE ENVIRONMENT
*
RESTORE:
           OUT     BankSave,BSR     ; RESTORE BSR
           ZALH    ACCH             ; RESTORE UPPER ACCUMULATOR
           ADDS    ACCL             ; RESTORE LOWER ACCUMULATOR
           LST     STATUS           ; RESTORE STATUS
           EINT                     ; ENABLE INTERRUPTS
           RET                      ; RETURN TO INTERRUPTED CODE
*
           END
```

5.3 Memory Management

The TMS320C14/E14 has a modified Harvard architecture in which program information and data information reside in two separate memories. Therefore, the next instruction fetch can occur while the current instruction is fetching data and executing the operation. The use of the Harvard architecture increases the speed of the device, but requires the use of special instructions to transfer a word between program memory and data memory.

The data memory consists of 256 words of on-chip RAM. Three forms of data memory addressing are available for use; direct, indirect, and immediate. Direct addressing uses the seven lower bits of the instruction word concatenated with the data page pointer to form the data memory address. Indirect addressing uses the lower eight bits of the auxilary registers as the data memory address. Immediate addressing uses part of the instruction word for data rather than data RAM.

The TMS320C14/E14 provides two options of program memory usage: either 4K words of internal ROM/EPROM can be used in the microcomputer mode, or the same amount of external memory can be used in the microprocessor mode.

The TMS320C14/E14 microprocessor and microcomputer modes are configurable through hardware or software. The hardware method uses the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ and $\overline{\text{RS}}$ pins to set the mode (Section 3.2.1). The software method uses bit 0 of the SYSCON register to set the mode. A 1 written to this bit sets the CPU in the microprocessor mode.

The initial configuration of the CPU is not binding, that is, it can be changed later into the other mode and then back again if desired. This affectively doubles the amount of program memory to 8K words. In Example 5-4, the CPU is set to the microprocessor mode.

Example 5-4. Memory Expansion Routine

```
*
*   THIS IS AN EXAMPLE OF HOW TO MANIPULATE THE MC/MP BIT IN THE
*   SYSCON REGISTER TO FORCE MICROPROCESSOR MODE.  THAT IS, TO FORCE
*   PROGRAM ACCESS TO EXTERNAL MEMORY.
*
          .include   "C14INC"    ; INCLUDE HEADER FILE
*
*   IT IS ASSUMED THAT BANK AND TMP ARE ALLOCATED IN THE CURRENT
*   MEMORY.
*
          .ref    BANK,TMP
*
*   IT IS ALSO ASSUMED THAT 'EXTERNAL' IS THE ADDRESS IN EXTERNAL
*   MEMORY WHICH IS TO BE EXECUTED AFTER THE MC/MP BIT CHANGE.
*
          .ref    EXTERNAL
*
          .text
*
          .
          .
          .
*
          LACK    SYSBank
          SACL    BANK
          OUT     BANK,BSR
          ZAC
          SACL    TMP
          OUT     TMP,SYSCON
          CALL    EXTERNAL       ; CALL SUBROUTINE IN EXTERNAL MEMORY.
                                 ; PROGRAM FLOW RETURNS HERE IF
                                 ; MC/MP BIT IS SET TO 1 PRIOR
                                 ; TO THE 'RET' INSTRUCTION
          END
```

5.4 Logical and Arithmetic Operations

Although the TMS320C14/E14 instruction set is oriented toward digital signal processing, the same fundamental operations of a general-purpose processor, such as bit manipulation, logical and arithmetic operations, logical and arithmetic shifts, and overflow management, are included. Explanations and examples of how to use instructions for scaling, convolution operations, fixed-point multiplication/division/addition, and floating-point arithmetic are also included in this section.

The contents of the accumulator may be stored in data memory using the SACH and SACL instructions or stored in the stack by using the PUSH instruction. Note that PUSH and POP only affect the lower 12 bits of the accumulator. The accumulator may be loaded from data memory using the ZALH, ZALS, and LAC instructions, which zero the accumulator before loading the data value. The ZAC instruction zeroes the accumulator. POP can be used to restore the accumulator contents from the stack. The accumulator is also affected by the execution of the ABS instruction, which replaces the contents of the accumulator with its absolute value.

5.4.1 Bit Manipulation

A specified bit of a word from data memory can either be set, cleared, or tested. Such bit manipulations are accomplished by using the accumulator, the hardware shifter and the logic instructions, AND, OR, and XOR. In Example 5-5, operations on single bits are performed on the data word VALUE. In this and the following example, data memory location ONE contains the value 1 and MINUS contains the value -1 (all bits set).

Example 5-5. Single-Bit Manipulation

```
*
* CLEAR BIT 5 OF DATA MEMORY LOCATION VALUE. MEMORY
* LOCATION ONE CONTAINS CONSTANT 1. MEMORY LOCATION MINUS
* CONTAINS -1 OR 0FFFFh.
*
        LAC   ONE,5    ; ACC = 00000020h
        XOR   MINUS    ; INVERT ACCUMULATOR; ACC = 0000FFDFh
        AND   VALUE    ; BIT 5 OF VALUE IS ZEROED
        SACL  VALUE
*
* SET BIT 12 OF VALUE.
*
        LAC   ONE,12   ; ACC = 00001000h
        OR    VALUE    ; BIT 12 OF VALUE
        SACL  VALUE
*
* TEST BIT 3 OF VALUE.
*
        LAC   ONE,3    ; ACC = 00000008h
        AND   VALUE    ; TEST BIT 3 OF VALUE
        BZ    BIT3Z    ; BRANCH TO BIT3Z IF BIT IS CLEAR
```

More than one bit can be set, cleared, or tested at one time if the necessary mask exists in data memory. In Example 5-6, the six low-order bits in the word VALUE are cleared if MASK contains the value 63.

Example 5-6. Multiple-Bit Manipulation

```
*
* CLEAR LOWER SIX BITS OF VALUE. MEMORY LOCATION MASK
* CONTAINS THE MASK TO CLEAR THE BITS. MEMORY LOCATION
* MINUS CONTAINS -1 OR 0FFFFh.
*
        LAC   MASK    ; ACC = 0000003Fh
        XOR   MINUS   ; INVERT ACCUMULATOR; ACC = 0000FFC0h
        AND   VALUE   ; CLEAR LOWER SIX BITS
        SACL  VALUE
```

5.4.2 Overflow Management

The TMS320C14/E14 has two features that can be used to handle overflow management. These include the branch on overflow conditions and accumulator saturation (overflow mode). These features provide several options for overflow protection within an algorithm.

A program can branch to an error handler routine on an overflow of the accumulator by using the BV (branch on overflow) instruction. This instruction can be performed after any ALU operation that may cause an accumulator overflow.

The overflow mode is a feature useful for DSP applications. This mode emulates the saturation effect characteristic of analog systems. When enabled, any overflow in the accumulator results in the accumulator contents being replaced with the largest positive value (7FFFFFFFh) if the overflowed number is positive, or the largest negative value (80000000h) if negative. The overflow mode is controlled by the OVM bit of the status register and can be changed by the SOVM (set overflow mode), ROVM (reset overflow mode), or LST (load status register) instructions. Overflows can be detected in software by testing the OV (overflow) bit in the status register. When a branch is used to test the overflow bit, OV is automatically reset. Note that the OV bit does not function as a carry bit. It is set only when the absolute value of a number is too large to be represented in the accumulator, and it is not reset except by specific instructions. The overflow mode feature affects all arithmetic operations in the ALU.

In Example 5-7, the accumulator saturates to 7FFFFFFFh or the largest positive value. The BV instruction also clears the OV bit.

Example 5-7. Overflow Management

```
* THE ACCUMULATOR WILL SATURATE TO THE HIGHEST POSITIVE
* VALUE WHEN OVERFLOW OCCURS. THE ACCUMULATOR CONTAINS
* 7FFFF423h. MEMORY LOCATION A CONTAINS 74EDh. MEMORY
* LOCATION B CONTAINS 67AFh.
*
        SOVM          ; SET OVERFLOW MODE
        LT    A       ; T = 74EDh
        MPY   B       ; P = 2F5B4903h
        APAC          ; ACC = 7FFFFFFFh
        BV    OVRFLW  ; CHECK OV BIT
*                     ; BRANCH TO OVERFLOW HANDLING ROUTINE
```

The effect on the accumulator before and after the code execution is shown as follows:

	Before Code Execution	After Code Execution
ACC	7FFFF423h	7FFFFFFFh

5.4.3 Scaling

Scaling the data coming into the accumulator or already in the accumulator is useful in signal processing algorithms. This is frequently necessary in adaptation or other algorithms that must compute and apply correction factors or normalize intermediate results. Scaling and normalizing are implemented on the TMS320C14/E14 via shifts of data on the incoming path to the accumulator.

There are two types of shifts: logical and arithmetic. A logical shift is implemented by filling the empty bits to the left of the MSB with zeros, regardless of the value of the MSB. An arithmetic shift fills the empty bits to the left of the MSB with ones if the MSB is one, or with zeros if the MBS is zero. The second type of bit padding is referred to as sign extension.

Data can be left-shifted 0 to 16 bits when the accumulator is loaded, and left-shifted 0, 1, or 4 bits when storing from the accumulator using the SACH instruction. These shifts can be used for loading numbers into the high 16 bits of the accumulator and renormalizing the result of a multiply. The incoming left shift of 0 to 16 bits is supplied in the instruction itself. Left shifts of data fetched from data memory are available for loading the accumulator (LAC), adding to the accumulator (ADD), and subtracting from the accumulator (SUB). When data is left-shifted 16 bits, the ZALH, ADDH, and SUBH instructions are used. The left-shift of 0, 1, or 4, available with the SACH instruction, is used to shift out the extra sign bits when fractional multiplication is used (see Section 5.4.5).

The hardware shift, which is built into the ADD, SUB, and LAC instructions, performs an arithmetic left-shift on a 16-bit word. This feature can also be used to perform right-shifts. A right-shift of n is implemented by performing a left-shift of 16-n and saving the upper word of the accumulator. Example 5-8 performs an arithmetic right-shift of 7 on a 16-bit number in the accumulator.

Example 5-8. Arithmetic Right-Shift

```
SACL TEMP    ; MOVE NUMBER TO MEMORY
LAC  TEMP,9  ; SHIFT LEFT (16-7)
SACH TEMP    ; SAVE HIGH WORD IN MEMORY
LAC  TEMP    ; RETURN NUMBER BACK TO ACCUMULATOR
```

The effect on the accumulator before and after the code execution is shown as follows:

	Before Code Execution	After Code Execution
ACC	FFFFA452h	FFFFFF48h

A logical right-shift of 4 on a 32-bit number stored in the accumulator is shown in Example 5-9. The 32-bit results of the shift are then stored in data memory. In this example, the accumulator initially contains the hexadecimal number, 9D84C1B2h. The variables, SHIFTH and SHIFTL, will receive the high word (09D8h) and low word (4C1Bh) of the shifted results.

Example 5-9. Logical Right-Shift

```
*
* SHIFT THE LOWER WORD. MEMORY LOCATION MINUS CONTAINS -1
* OR FFFFh.
*
        SACH SHIFTH     ; SHIFTH = 9D84h    INITIAL VALUES
        SACL SHIFTL     ; SHIFTL = 0C1B2h
        LAC  SHIFTL,12  ; ACC = 0FC1B2000h
        SACH SHIFTL     ; SHIFTL = 0FC1Bh
        LAC  MINUS,12   ; ACC = 0FFFFF000h
        XOR  MINUS      ; ACC = 0FFFF0FFFh
        AND  SHIFTL     ; ACC = 00000C1Bh
*
* SHIFT THE UPPER WORD.
*
        ADD  SHIFTH,12  ; ACC = 0F9D84C1Bh
        SACL SHIFTL     ; SHIFTL = 4C1Bh    FINAL LOW VALUE
        SACH SHIFTH     ; SHIFTH = 0F9D8h
        LAC  MINUS,12   ; ACC = 0FFFFF000h
        XOR  MINUS      ; ACC = 0FFFF0FFFh
        AND  SHIFTH     ; ACC = 000009D8h
        SACL SHIFTH     ; SHIFTH = 09D8h    FINAL HIGH VALUE
        ZALH SHIFTH
        ADDS SHIFTL
```

The accumulator is affected before and after the code execution as follows:

	Before Code Execution	After Code Execution
ACC	9D84C1B2h	09D84C1Bh

An arithmetic right-shift of 4 can be implemented using the same routine as shown above, except with the last four lines omitted.

5.4.4 Convolution Operations

Many DSP applications must perform convolution operations or other operations similar in form. These operations require data to be shifted or delayed. The DMOV and LTD instructions can perform the needed data moves for convolution.

The data move function is used for on-chip data memory. It allows a word to be copied from the currently addressed data memory location in on-chip RAM to the next higher location while the data from the addressed location is being operated upon (e.g., by the CALU). The data move and the CALU operation are performed in the same cycle. The data move function is useful in implementing algorithms, such as convolutions and digital filtering, where data is being passed through a time window. It models the z^{-1} delay operation encountered in those applications.

5.4.5 Multiplication

The TMS320C14/E14 hardware multiplier normally performs two's-complement 16-bit by 16-bit multiplies and produces a 32-bit result in a single processor cycle. To multiply two operands, one operand must be loaded into the T register. The second operand is moved by the multiply instruction to the multiplier, which then produces the product in the P register. Before another multiply can be performed, the contents of the P register must be moved to the accumulator. By pipelining multiplies and P-register moves, most multiply operations can be performed with a single instruction.

Computation on the TMS320C14/E14 is based on a fixed-point two's-complement representation of numbers. Each 16-bit number is evaluated with a sign bit, i integer bits, and 15-i fractional bits. Thus, the number

```
0 0000010 10100000
         |
         └─ binary point
```

has a value of 2.625. This particular number is said to be represented in a Q8 format (8 fractional bits). Its range is between -128 (1000000000000000) and 127.996 (0111111111111111). The fractional accuracy of a Q8 number is about 0.004 (one part in 2^8 or 256).

Although particular situations (e.g., a combination of dynamic range and accuracy requirements) must use mixed notations, it is more common to work entirely with fractions represented in a Q15 format or integers in a Q0 format. This is especially true for signal processing algorithms where multiply and accumulate operations are dominant. The result of a fraction times a fraction remains a fraction, and the result of an integer times an integer remains an integer. No overflows are possible.

Q format is a number representation commonly used when performing operations on noninteger numbers. In Q format, the Q number (15 in Q15) denotes how many bits are located to the right of the binary point. A 16-bit number in Q15 format, therefore, has an assumed binary point immediately to the right of the most significant bit. Since the most significant bit constitutes the sign of the number, then numbers represented in Q15 may take on values from +1 (represented by +0.99997...) to -1.

A wide variety of situations may be encountered when multiplying two numbers. Three of these situations are provided in Example 5-10, Example 5-11, and Example 5-12.

Example 5-10. Fraction × Fraction

```
                          0100000000000000   = 0.5  in Q15
                        × 0100000000000000   = 0.5  in Q15
    ----------------------------------------
    00 01000000000000 0000000000000000       = 0.25 in Q30
      |
      ├─ binary point
```

Two sign bits remain after the multiply. Generally, a single-precision (16-bit) result is saved, rather than maintaining the full intermediate precision. The upper half of the result does not contain a full 15 bits of fractional precision since the multiply operation actually creates a second sign bit. In order to recover that precision, the product must be shifted left by one bit, as shown in the following code excerpt:

```
LT      OP1   ; OP1 = 4000h (0.5 in Q15)
MPY     OP2   ; OP2 = 4000h (0.5 in Q15)
PAC
SACH    ANS,1 ; ANS = 2000h (0.5 in Q15)
```

The MPYK instruction provides a multiply by a 13-bit signed constant. In fractional notation, this means that a Q15 number can be multiplied by a Q12 number. The resulting number must be left-shifted by four bits to maintain full precision.

```
LT      OP1   ; OP1 = 4000h (0.5  in Q15)
MPYK    2048  ; OP2 = 0800h (0.5  in Q12)
PAC
SACH    ANS,4 ; ANS = 2000h (0.25 in Q15)
```

Example 5-11. Integer × Integer

```
                        0000000000010001  =  17 in Q0
                      × 1111111111111011  =  -5 in Q0
    --------------------------------------
    1111111111111111 1111111110101011  = -85 in Q0
                                      |
                                      ├─ binary point
```

In this case, the extra sign bits do not change the result, and the desired product is entirely in the lower half of the product, as shown in the following program:

```
LT      OP1   ; OP1 = 0011h ( 17 in Q0)
MPY     OP2   ; OP2 = FFFBh ( -5 in Q0)
PAC
SACH    ANS   ; ANS = FFABh (-85 in Q0)
```

Example 5-12. Mixed Notation (Q14 × Q14 = Q28)

```
                      0110000000000000   = 1.50  in Q14
                    × 0011000000000000   = 0.75  in Q14
     ─────────────────────────────────
0001  001000000000  0000000000000000   = 1.125 in Q28
    └─ binary point
```

The maximum magnitude of a Q14 number is two. Thus, the maximum magnitude of the product of two Q14 numbers is four. Two integer bits are required to allow for this possibility, leaving a maximum precision for the product of 13 bits. In general, the following rule applies: The product of a number with i integer bits and f fractional bits and a second number with j integer bits and g fractional bits will be a number with (i+j) integer bits and (f+g) fractional bits. The highest precision possible for a 16-bit representation of this number will have (i+j) integer bits and (15-i-j) fractional bits.

If the physical system being modelled is well understood, the precision with which the number is modelled can be increased. For example, if it is known that the above product can be no more than 1.8, the product can be represented as a Q14 number rather than the theoretical worst case of Q13, shown in the following program:

```
LT      OP1       ; OP1 = 6000h (1.5   in Q14)
MPY     OP2       ; OP2 = 3000h (0.75  in Q14)
PAC
SACH    ANS,1     ; ANS = 2400h(1.125 in Q13)
```

The techniques illustrated in the previous three examples all truncate the result of the multiplication to the desired precision. The error generated as a result can be as much as minus one full LSB. This is true whether the truncated number is positive or negative. It is possible to implement a simple rounding technique to reduce this potential error by a factor of two, as shown in the code sequence of Example 5-13. The maximum error generated in this example is plus one-half LSB whether ANS is positive or negative.

Example 5-13. Multiplication Rounding Technique

```
LT      OP1
MPY     OP2       ; OP1 * OP2
PAC
ADD     ONE,14    ; ROUND UP
SACH    ANS,1
```

A common operation in DSP algorithms is the summation of products. The contents of the P register are added to the accumulator, and two values simultaneously read and multiplied. A data memory value is multiplied by a second data memory value. Example 5-14 shows an implementation of multiplies and accumulates using the LTA-MPY instruction pair.

Example 5-14. Multiply and Accumulate

```
*                 CLOCK   TOTAL CLOCK   PROGRAM   TOTAL PROGRAM
*                 CYCLES    CYCLES      MEMORY       MEMORY
*
     ZAC            1  ┐                   1  ┐
     LT     D1      1  │                   1  │
     MPY    C1      1  │                   1  │
     LTA    D2      1  │                   1  │
     MPY    C2      1  ├─ 2N               1  ├─ 2N
     .                 │                      │
     .                 │                      │
     .                 │                      │
     LTA    DN      1  │                   1  │
     MPY    CN      1  ┘                   1  ┘
     APAC           1       2 + 2N         1       2 + 2N
```

5.4.6 Division

Binary division is the inverse of multiplication. Multiplication consists of a series of shift and add operations, while division can be broken into a series of subtracts and shifts. Although the first-generation TMS320 does not have an explicit divide instruction, it is possible to implement an efficient flexible divide capability using the conditional subtract instruction, SUBC. SUBC implements binary division in the same manner as is commonly done in long division. Given a 16-bit positive dividend and divisor, the repetition of the SUBC command 16 times produces a 16-bit quotient in the low accumulator and a 16-bit remainder in the high accumulator. With each SUBC, the divisor is left-shifted 15 bits and subtracted from the accumulator (or divided). For each subtract not producing a negative answer, the results are stored in the accumulator which is then shifted and a one is put in the LSB of the accumulator. For each subtract producing a negative answer, the accumulator is simply left-shifted. The shifting of the remainder and quotient after each subtract produces the separation of the quotient and remainder in the low and high halves of the accumulator. The similarities between long division and the SUBC method of division are shown in Figure 5-1 where 33 is divided by 5.

LONG DIVISION:

```
                  0000000000000110      Quotient
0000000000000101 )0000000000100001
                              -101
                               110
                              -101
                                11      Remainder
```

SUBC METHOD:

```
32  HIGH ACC          LOW ACC     0                  COMMENT
|                |  |                |

0000000000000000  0000000000100001  (1)  Dividend is loaded into ACC. The
             -10  1000000000000000       divisor is left-shifted 15 and sub-
             -10  0111111111011111       tracted from ACC. The subtraction
                                         is negative, so discard the result
|                |  |                |   and shift left the ACC one bit.
0000000000000000  0000000001000010  (2)  2nd subtract produces negative
             -10  1000000000000000       answer, so discard result and shift
             -10  0111111110111110       ACC (dividend) left.

                 •                                    •
                 •                                    •
                 •                                    •
|                |  |                |
0000000000000100  0010000000000000  (14) 14th SUBC command. The result
             -10  1000000000000000       is positive. Shift result left and
0000000000000001  1010000000000000       replace LSB with '1'.

|                |  |                |
0000000000000011  0100000000000001  (15) Result is again positive. Shift
             -10  1000000000000000       result left and replace LSB with '1'.
0000000000000000  1100000000000001

|                |  |                |
0000000000000001  1000000000000011  (16) Last subtract. Negative answer, so
             -10  1000000000000000       discard result and shift ACC left.
                - 1111111111111101

0000000000000011  0000000000000110       Answer reached after 16 SUBC
                                         instructions.
|   REMAINDER    |  |   QUOTIENT     |
```

Figure 5-1. Long Division and SUBC Division

The condition of the divisor, less than the shifted dividend, is determined by the sign of the result. The only restriction for the use of the SUBC instruction is that both the dividend and divisor MUST be positive. Thus, the sign of the quotient must be determined and the quotient computed using the absolute value of the dividend and divisor. In addition, when implementing a divide algorithm, it is important to know if the quotient can be represented as a fraction and the degree of accuracy to which the quotient is to be computed. Each of these considerations can affect how the SUBC instruction is used (see Example 5-15 and Example 5-16). Note that the next instruction after SUBC cannot use the accumulator.

Example 5-15. Using SUBC With Numerator < Denominator

```
* THIS ROUTINE DIVIDES TWO BINARY, TWO'S-COMPLEMENT NUMBERS
* OF ANY SIGN WHERE THE NUMERATOR IS LESS THAN THE
* DENOMINATOR.
*
*                 BEFORE        AFTER
*                EXECUTION    EXECUTION
*
* NUMERA           21            21
* DENOM            42            42
* QUOT              0            0.5
*                              (0.1 0 0)
*
DIV    LARP 0
       LT   NUMERA   ; GET SIGN OF QUOTIENT
       MPY  DENOM
       PAC
       SACH TEMSGN   ; SAVE SIGN OF QUOTIENT
       LAC  DENOM
       ABS
       SACL DENOM    ; MAKE DENOMINATOR POSITIVE
       ZALH NUMERA   ; ALIGN NUMERATOR
       ABS           ; MAKE NUMERATOR POSITIVE
       LARK 0,14
*
* IF DIVISOR AND DIVIDEND ARE ALIGNED, DIVISION CAN START
* HERE.
*
KPDVNG SUBC DENOM    ; 15-CYCLE DIVIDE LOOP
       BANZ KPDVNG
*
       SACL QUOT
       LAC  TEMSGN
       BGEZ DONE     ; DONE IF SIGN IS POSITIVE
       ZAC
       SUB  QUOT
       SACL QUOT     ; NEGATE QUOTIENT IF NEGATIVE
DONE   RET           ; RETURN TO MAIN PROGRAM
```

Example 5-16. Using SUBC With Specified Quotient Accuracy

```
* THIS ROUTINE DIVIDES TWO BINARY, TWO'S-COMPLEMENT NUMBERS
* OF ANY SIGN, SPECIFYING THE FRACTIONAL ACCURACY OF THE
* QUOTIENT (FRAC).
*
*                  BEFORE       AFTER
*                EXECUTION    EXECUTION
*
* NUMERA            11           11
* DENOM              8            8
* FRAC               3            3
* QUOT              17          1.375
*                              (1.0 1 1)
*
DN1     LT   NUMERA    ; GET SIGN OF QUOTIENT
        MPY  DENOM
        PAC
        SACH TEMSGN    ; SAVE SIGN OF QUOTIENT
        LAC  DENOM
        ABS
        SACL DENOM     ; MAKE DENOMINATOR POSITIVE
        LACK 15
        ADD  FRAC
        SACL FRAC      ; COMPUTE LOOP COUNT
        LAC  NUMERA    ; ALIGN NUMERATOR
        ABS            ; MAKE NUMERATOR POSITIVE
        LAR  0,FRAC
*
* IF DIVISOR AND DIVIDEND ARE ALIGNED, DIVISION CAN START
* HERE.
*
KPDVNG SUBC DENOM      ; 16 + FRAC CYCLE DIVIDE LOOP
       BANZ KPDVNG
*
        SACL QUOT
        LAC  TEMSGN
        BGEZ DONE      ; DONE IF SIGN IS POSITIVE
        ZAC
        SUB  QUOT
        SACL QUOT      ; NEGATE QUOTIENT IF NEGATIVE
DONE    RET            ; RETURN TO MAIN PROGRAM
```

5.4.7 Addition

Both operands in addition must be represented in the same Q format. Enough room must be allowed in the result to accommodate bit growth or there must be some preparation to handle overflows. If the operands are only 16 bits long, the result may have to be represented as a double-precision number. Example 5-17 and Example 5-18 illustrate two approaches to adding 16-bit numbers.

Example 5-17. Maintaining 32-Bit Results

```
        LAC  OP1      ; Q15
        ADD  OP2      ; Q15
        SACH ANSHI    ; HIGH-ORDER 16 BITS OF RESULT
        SACL ANSLO    ; LOW-ORDER 16 BITS OF RESULT
```

Example 5-18. Adjusted Binary Point for 16-Bit Results

```
LAC  OP1,15   ; Q14 NUMBER IN ACCH
ADD  OP2,15   ; Q14 NUMBER IN ACCH
SACH ANS      ; Q14
```

Double-precision operands present a more complex problem since actual arithmetic overflows or underflows may occur. The BV (branch on overflow) instruction can be used to check for the occurrence of these conditions. A second technique is the use of saturation mode operations, which will saturate the result of overflowing accumulations to the most positive or most negative number. Both techniques, however, result in a loss of precision. The best technique involves a thorough understanding of the underlying physical process and care in selecting number representations.

5.4.8 Floating-Point Arithmetic

Although the TMS320C14/E14 devices are fixed-point 16/32-bit microprocessors, they can also perform floating-point computations. Using the floating-point single-precision standard proposed by the IEEE, the TMS320C14/E14 can perform a floating-point multiplication in 8.4 µs and a floating-point addition in 17.2 µs. For a detailed discussion of floating-point arithmetic and TMS320 source code, refer to "Floating-Point Arithmetic with the TMS32010," an application report in the book, *Digital Signal Processing Applications with the TMS320 Family.*

Floating-point numbers are often represented on microprocessors in a two-word format of mantissa and exponent. The mantissa is stored in one word. The exponent, the second word, indicates how many bit positions from the left the binary point is located. If the mantissa is 16 bits, a 4-bit exponent is sufficient to express the location of the binary point. Because of its 16-bit word size, the 16/4-bit floating-point format functions most efficiently on the TMS320C14/E14.

Operations in the TMS320C14/E14 central ALU are performed in two's-complement fixed-point notation. To implement floating-point arithmetic, operands must be converted to fixed point for arithmetic operations, and then converted back to floating point. Conversion to floating-point notation is performed by normalizing the input data (i.e., shifting the MSB of the data word into the MSB of the internal memory word). The exponent word then indicates how many shifts are required. To multiply two floating-point numbers, the mantissas are multiplied and the exponents added. The resulting mantissa must be renormalized. (Since the input operands are normalized, no more then one left shift is required to normalize the result.)

Floating-point addition or subtraction requires shifting the mantissa so that the exponents of the two operands match. The difference between the exponents is used to left-shift the lower power operand before adding. Then, the output of the add must be renormalized.

Instructions useful in floating-point operations are the LAC, LACK, ADD, and SUB instructions. The mantissas are often maintained in Q15 format. Q format is a number representation commonly used when performing operations on noninteger numbers. In Q format, the Q number (15 in Q15) denotes how many digits are located to the right of the binary point. A 16-bit number in Q15 format, therefore, has an assumed binary point immediately to the right of the most significant bit. Since the most significant bit constitutes the sign of the number, then numbers represented in Q15 may take on values from +1 (represented by +0.99997...) to -1.

5.5 PID Control

Control systems are concerned with regulating a process and achieving a desired behavior or output from the process. A control system consists of three main components: sensors, actuators, and a controller. Sensors measure the behavior of the system. Actuators supply the driving force to ensure the desired behavior. The controller generates actuator commands corresponding to the error conditions observed by the sensors and the control algorithms programmed in the controller. The controller typically consists of an analog or digital processor.

Analog control systems are usually based on fixed components and are not programmable. They are also limited to using single-purpose characteristics of the error signal, such as P (proportional), I (integral), and D (derivative), or their combination. These limitations, along with other disadvantages of analog systems such as component aging and temperature drift, are causing digital control systems to increasingly replace analog systems in most control applications.

Digital control systems that use a microprocessor/microcontroller are able to implement more sophisticated algorithms of modern control theory, such as state models, deadbeat control, state estimation, optimal control, and adaptive control. Digital control algorithms deal with the processing of digital signals and are similar to DSP algorithms. The TMS320C1x instruction set can therefore be used very effectively in digital control systems.

The most commonly used algorithm in both analog and digital control systems is the PID (Proportional, Integral, and Derivative) algorithm. The classical PID algorithm is given by

$$u(t) = K_p\, e(t) + K_i \int edt + K_d\, de/dt$$

The PID algorithm must be converted into a digital form for implementation on a microprocessor. Using a rectangular approximation for the integral, the PID algorithm can be approximated as

$$u(n) = u(n\text{-}1) + K_0\, e(n) + K_1\, e(n\text{-}1) + K_2\, e(n\text{-}2)$$

This algorithm is implemented in Example 5-19.

Example 5-19. PID Control

```
        .title  'PID CONTROL'
        .def    PID
*
* THIS ROUTINE IMPLEMENTS A PID ALGORITHM.
*
        .ref    UN       ; OUTPUT OF CONTROLLER
        .ref    E0       ; LATEST ERROR SAMPLE
        .ref    E1       ; PREVIOUS ERROR SAMPLE
        .ref    E2       ; OLDEST ERROR SAMPLE
        .ref    K0       ; GAIN CONSTANT
        .ref    K1       ; GAIN CONSTANT
        .ref    K2       ; GAIN CONSTANT
*
* ASSUME DATA PAGE 0 IS SELECTED.
*
        .text
PID     IN      E0,PA0   ; READ NEW ERROR SAMPLE
        LAC     UN       ; ACC = u(n-1)
        LT      E2       ; LOAD T REG WITH OLDEST SAMPLE
        MPY     K2       ; P = K2*e(n-2)
        LTD     E1       ; ACC = u(n-1)+K2*e(n-2)
        MPY     K1       ; P = K1*e(n-1)
        LTD     E0       ; ACC = u(n-1)+K1*e(n-1)+K2*e(n-2)
        MPY     K0       ; P = K0*e(n)
        APAC             ; ACC = u(n-1)+K0*e(n)+K1*e(n-1)
*                        ;       +K2*e(n-2)
        SACH    UN,1     ; STORE OUTPUT
        OUT     UN,PA1   ; SEND IT
```

The PID loop takes 13 cycles to execute or 2.03 μs at a 25.6-MHz clock rate. The TMS320 can also be used to implement more sophisticated algorithms such as state modeling, adaptive control, state estimation, Kalman filtering, and optimal control. Other functions that can be implemented are noise filtering, stability analysis, and additional control loops.

5.6 PWM Generation

The TMS320C14/E14 can be configured to generate high-precision Pulse Width Modulation (PWM) outputs. In this function, the compare output pins are controlled directly by the compare registers. The contents of the compare registers control the duration of the high portion of the PWM waveform. The PWM output is useful in controlling stepper motors.

The following example contains code that could be used in adjusting the phase response of a three phase motor. By controlling the width of the PWM waveform output a designer can control the relative phase of a three phase motor. Timer 2 is used to determine the duration of the PWM pulse width.

Example 5-20. Using Compare Outputs For Motor control

```
*
*   PWM GENERATION WITH COMPARE OUTPUTS EXAMPLE.  SETTING BIT 8
*   OF THE TCON REGISTER OVERRIDES ANY OTHER CONFIGURATION.  THE
*   COMPARE REGISTER(S) MUST BE SET UP WITH THE DURATION OF THE
*   HIGH PORTION OF THE PWM OUTPUT.  IF THESE VALUES ARE TO BE
*   CHANGED, WRITING TO THE PROPER ACTION REGISTER WILL ACCOMPLISH
*   THE TASK IMMEDIATELY AT THE END OF THE CURRENT CYCLE.
*
           .include "C14INC"       ; INCLUDE HEADER FILE
*
*   IT IS ASSUMED THAT 'BANK' AND 'TMP' ARE AVAILABLE TO STORE
*   VALUES TO THE BSR AND THE CONTROL REGISTER WHICH IS TO BE
*   ALTERED.  IT IS ALSO ASSUMED THAT 'ONE' HOLDS THE CONSTANT 1.
*
           .ref     BANK,TMP,ONE
*
*   IT IS ALSO ASSUMED THAT 'PWMperiod' HOLDS THE DESIRED PERIOD
*   AND 'PWMwidth' HOLDS THE WIDTH.
*
           .ref     PWMperiod,PWMwidth
*
*          .
*          .
*          .
           .data
*
TABLE:
           .word    CMPPWMMode+CMPTMR1Select+TMR1Enable*TMRby1+TMR1Internal
*          .
*          .
*          .
           .text
*
*   SET UP PERIOD REGISTER OF TIMER 1
*
           LACK     TimerBank
           SACL     BANK
           OUT      BANK,BSR
           OUT      PWMperiod,TPR1
*
*   GET MODE WORD FOR TCON
*
           LT       ONE
           MPYK     TABLE
           PAC
           TBLR     TMP
           OUT      TMP,TCON        ; LOAD TCON
*
           LACK     ActionBank
           SACL     BANK
           OUT      BANK,BSR
           OUT      PWMwidth,ACT5   ; UPDATE PWM WIDTH
*
*          .
*          .
*          .
           END
```

5.7 Speed/Position Measurement

The capture subsystem on the TMS320C14/E14 is capable of detecting and storing any transition on it's capture input pins. Using optical encoders, a digital representation of an analog process such as speed measurement can be obtained. An incremental encoder creates a series of square waves. The number of square waves corresponds to the mechanical increment required. For an encoder that supplies 1024 square wave cycles per revolution (360 degrees), each increment of one corresponds to 0.351 degrees of revolution. Using a counter, the rotation of the shaft can be calculated.

A dual channel incremental encoder can be used for position sensing, as shown in Figure 5-2. Most incremental systems use two output channels in quadrature to allow the designer to count the transitions from a high state to a low state, and view the state of the opposite channel during these transitions. Using this information, you can determine if channel A leads channel B and derive the direction.

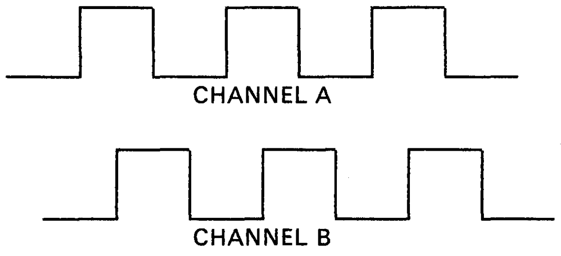


Figure 5-2. Dual Channel Optical Encoder Outputs

Example 5-21 instructs the CPU to read the capture input once every 0.5 ms, compare the input to a value (contents of Timer 1) and toggle the Bit I/O pins 0 and 1 for every match.

Example 5-21. Using the Capture Inputs To Detect Speed

```
*
*   USE CAPTURE INPUTS TO DETERMINE SPEED.  BITS 11 THROUGH 15 OF TCON
*   ARE PRESUMED TO BE 0.  THIS THEN IMPLIES THAT TIMER 1 IS USED FOR
*   CAPTURE OPERATIONS WITH INTERNAL CLOCK SOURCE WITH AN INTERRUPT
*   GENERATED ON THE FIRST CAPTURE ENTRY RECEIVED IN FIFO0.  THE OPTICAL
*   ENCODER OUTPUTS A AND B ARE CONNECTED TO CAP0 AND CAP1 INPUTS.  ANY
*   TRANSITION FROM HIGH TO LOW WILL TRIGGER A CAPTURE.  THIS ROUTINE
*   COMPUTES MEAN TIME BETWEEN A AND B PULSES AND DETERMINES SIGN OF
*   FORWARD DIFFERENCE OF TIME BETWEEN AN A PULSE AND A B PULSE.
*
*    INCLUDE C14 EQUATES
*
     .include "C14INC"     ; INCLUDE HEADER FILE
*
     .def  CaptureInit,CaptureISR
*
     .data
*
TABLE:
     .word  CAP0Mode*NegEdgeDetect+CAP1Mode*NegEdgeDetect
     .word  (_CAPINT0 + _CAPINT1) ∧ 0FFFFh   ; CONSTRUCT CAPTURE MASK
*
*
     .ref IFimage          ; INTERRUPT FLAG IMAGE
     .ref IMimage          ; INTERRUPT MASK IMAGE
     .ref BANK             ; TEMPORARY STORAGE FOR VALUE FOR BSR
     .ref TMP              ; SCRATCH SPACE
     .ref POINT            ; TEMPORARY STORAGE FOR TABLE POINTER
     .ref ONE              ; THE VALUE 1 SAVED AS A CONSTANT
     .ref SampleRate       ; NUMBER OF CLOCK TICKS FOR 0.5 MS
     .ref MeanATime        ; MEAN CLOCK TICKS BETWEEN TRANSITIONS ON A
     .ref MeanBTime        ; MEAN CLOCK TICKS BETWEEN TRANSITIONS ON B
     .ref FIFO0value       ; LAST FIFO0 VALUE
     .ref FIFO1value       ; LAST FIFO1 VALUE
     .ref ABdifference     ; FIFO0 - FIFO1
     .ref IntBankSave      ; PRESUMED INITIALIZED TO InterruptBank
*
     .bss CapBankSave,1    ; DEFINE CAPTURE BANK
     .bss TimBankSave,1    ; DEFINE TIMER BANK
*
*  THE FOLLOWING CODE INITIALIZES CCON.  IT WOULD BE REFERENCED DURING
*  INITIALIZATION.  USE TIMER 1 WITH INTERNAL CLOCK SOURCE WITH X1 PRE-
*  SCALE.  SET TIMER PERIOD TO 3125 (ABOUT 0.5 MS FOR A 25 MHZ DEVICE).
*
     .text
*
CaptureInit:
*
     LT   ONE
     MPYK TABLE
     PAC
     SACL POINT
     TBLR TMP
     LACK CaptureBank
     SACL CapBankSave      ; CapBankSave = CaptureBank
     OUT  CapBankSave,BSR
     OUT  TMP,CCON         ; INITIALIZE CCON FROM TABLE
*
     LAC  POINT
     ADD  ONE
     SACL POINT
     SACL POINT
     LACK TimerBank
     SACL TimBankSave      ; TimBankSave = TimerBank
     OUT  TimBankSave,BSR
     OUT  TMP,TPR1         ; INITIALIZE TIMER 1 PERIOD
*
     LAC  POINT
     ADD  ONE
     SACL POINT            ; POINT = ->TABLE[1]
     TBLR TMP              ; TMP = TABLE[1]
     OUT  IntBankSave,BSR
     IN   IMimage,IM       ; READ IM AND STORE IN IMimage
     LAC  TMP
     AND  IMimage
     SACL IMimage
     OUT  IMimage,IM       ; ENABLE CAPINT0 AND CAPINT1
     RET
*
*   CAPTURE ISR PRESUMES THAT A CAPTURE INTERRUPT WAS DETECTED,
*   AND THAT THE ENVIRONMENT HAS BEEN SAVED PRIOR TO THE DISPATCH
*   OF THIS ROUTINE. IT IS PRESUMED THAT THE IF REGISTER WAS SAVED AT
*   IFimage PRIOR TO IMAGE ENTRY INTO THE ROUTINE.  UPON COMPLETION OF
*   HANDLING THE CAPTURE INTERRUPT, THE ROUTINE EXECUTES A 'RET'
*   INSTRUCTION, RETURNING IT TO THE POINT WHERE IT WAS INVOKED FOR THE
*   RESTORATION OF THE ENVIRONMENT AND/OR FURTHER INTERRUPT PROCESSING.
*
CaptureISR:
     OUT  CapBankSave,BSR ; BSR -> CAPTURE BANK
*
     LAC  ONE,CAPINT0_BIT
     AND  IFimage
     BNZ  Capture0         ; IF (CAPINT0) THEN PROCESS IT
     LAC  ONE,CAPINT1_BIT
     AND  IFimage
     BNZ  Capture1         ; ELSE IF (CAPINT1) THEN PROCESS IT
     RET                   ; RETURN TO CALLING ROUTINE
*
Capture0:
     IN   TMP,FIFO0
     LAC  TMP
     SUB  FIFO0value
     ADD  MeanATime
     SACL MeanATime
     LAC  TMP
     SACL FIFO0value       ; FIFO0value = FIFO0
     LAC  MeanATime,15
     SACH MeanATime        ; MeanATime = (MeanATime + FIFO0dif)/2
*
     LAC  FIFO0value
     SUB  FIFO1value
     SACL ABdifference     ; ABdifference = FIFO0value - FIFO1value
     RET
*
Capture1:
     IN   TMP,FIFO1        ; Get FIFO 1 VALUE
     LAC  TMP
     SUB  FIFO1value       ; SUBTRACT PREVIOUS FIFO1
     ADD  MeanBTime
     SACL MeanBTime
     LAC  TMP
     SACL FIFO1value       ; UPDATE FIFO1value
     LAC  MeanBTime,15
     SACH MeanBTime        ; MeanBTime = (MeanBTime + FIFO1dif)/2
```

```
        SACL    FIFO1value    ; UPDATE FIFO1value
        LAC     MeanBTime,15
        SACH    MeanBTime     ; MeanBTime = (MeanBTime + FIFO1dif)/2
*
        RET
*
        END
```

5.8 Using the Serial Port

The TMS320C14/E14 has a dedicated Universal Synchronous/Asynchronous Receiver/Transmitter (USART) that can operate in either a synchronous, asynchronous, or codec mode. An independent timer is used for baud rate generation, if required. Two interrupts (receiver buffer full or transmitter buffer empty) are generated by the serial port to initiate transceiving.

The synchronous mode of operation supports half-duplex operation with a maximum rate of CLKOUT bps (= 6.4 Mbps @ 25.6 MHz). When configured in the slave mode, the serial port clock is external. The serial port can be configured in receive mode with protocol #2 active, meaning that the serial port is automatically decoding incoming data until it is addressed. When configured to operate in protocol #1 mode, it is the responsibility of the user to determine an address match. To start transmitting, bit 11 of the SCON register must be set to 0 by writing a mask of 1 to the SCLR register.

The asynchronous operation is full-duplex with internal baud rate generation, with a maximum transmission rate of CLKOUT/16 bps (= 400 Kbps @ 25.6 MHz). In this mode the serial port is capable of detecting parity, framing, and frame overrun errors.

The codec mode of operation is full duplex and is designed for direct interface to industry standard codecs, with a maximum transmission rate of CLKOUT/4 bps (= 1.6 Mbps @ 25.6 MHz). The frame sync pulses can be configured to be either internal or external. The received data in this mode is logarithmic code output and must be converted to linear. For further details on the conversion process refer to *Digital Signal Processing Applications with the TMS320 Family* (SPRA012A) pages 171-211.

5.8.1 Asynchronous Configuration

Example 5-22 below sets up the serial port in the asynchronous mode with 7 data bits, even parity, and normal reception. A software polling technique is used to determine if the receiver or the transmitter requires servicing, and data is either read from the receive buffer or written to the transmit buffer, depending on the source of the interrupt.

Example 5-22. Configuring for Asynchronous Operation

```
*
*   EXAMPLE PROGRAM SEGMENT TO INITIALIZE SERIAL PORT FOR
*   ASYNCHRONOUS OPERATION AND TO RESPOND TO INTERRUPTS FROM
*   TRANSMIT AND RECEIVE SECTIONS OF THE SUBSYSTEMS.
*
*   INCLUDE C14 EQUATES
*
        .Include"C14INC"         ; INCLUDE HEADER FILE
*
        .def    AsynchInit,AsynchISR
        .data
*
TABLE:
        .word   SBRG19200        ; CONSTANT FOR 19.2 KBAUD
        .word   AsynchMode+SevenDataBits+ParityEnable+EvenParity
        .word   (_RXINT + _TXINT) ∧ 0FFFFh
                                 ; CONSTRUCT IM MASK FOR SERIAL PORT
*
        .ref    IFimage          ; INTERRUPT FLAG IMAGE
        .ref    IMimage          ; INTERRUPT MASK IMAGE
        .ref    BANK             ; TEMPOARAY STORAGE FOR BSR VALUE
        .ref    TMP              ; SCRATCH SPACE
        .ref    POINT            ; TEMPORARY STORAGE FOR TABLE POINTER
        .ref    ONE              ; THE VALUE 1 SAVED AS A CONSTANT
*
        .ref    PortBankSave     ; INITIALIZED TO SerialPort BANK
                                 ; TO SPEED EXECUTION DURING
                                 ; INTERRUPT SERVICE ROUTINE
*
        .ref    IntBankSave      ; INITIALIZED TO InterruptBank
                                 ; TO SPEED EXECUTION DURING
                                 ; INTERRUPT SERVICE ROUTINE
        .text
*
*   THE FOLLOWING CODE INITIALIZES SCON. IT WOULD BE REFERENCED
*   DURING INITIALIZATION.
*
AsynchInit:
*
        LT      ONE
        MPYK    TABLE
        PAC
        SACL    POINT            ; POINT = ->TABLE[0]
        TBLR    TMP
        LACK    SerialPort
        SACL    PortBankSave     ; PortBankSAve = SerialPort
        OUT     PortBankSave,BSR
        OUT     TMP,SBRG         ; SET SBRG FOR 19.2 KBAUD
*
        LAC     POINT
        ADD     ONE
        SACL    POINT            ; POINT = ->TABLE[1]
        TBLR    TMP              ; TMP   = TABLE[1]
        OUT     TMP,SCON         ; INITIALIZE SCON FROM TABLE
*
        LAC     POINT
        ADD     ONE
        SACL    POINT            ; POINT = ->TABLE[2]
        TBLR    TMP              ; TMP = TABLE[2]
        LACK    InterruptBank
        SACL    IntBankSave
        OUT     IntBankSave,BSR     ; IntBankSAve = InterruptBank
        OUT     TMP,IM              ; ENABLE RXINT AND TXINT
*
        RET
*
*    THIS ROUTINE IS PRESUMED TO BE INVOKED FROM AN INTERRUPT
*    SERVICE DISPATCHER WHICH SAVES AND RESTORES ENVIRONMENT
*
AsynchISR:
        OUT     PortBankSave,BSR    ; BSR -> SERIAL PORT
*
        LAC     ONE,RXINT_BIT
        AND     IFimage
        BZ      Txtest              ; IF (RXINT) THEN PROCESS IT
*
*    PROCESS INTERRUPT FROM RECEIVER
*
*       .
*       .
*       .
*
Txtest:
        LAC     ONE,TXINT_BIT
        AND     IFimage
        BZ      AsynchExit          ; ELSE IF (TXINT) THEN PROCESS IT
*
*    PROCESS INTERRUPT FROM TRANSMITTER
*
*       .
*       .
*       .
*
AsynchExit:
        OUT     PortBankSave,BSR
        ZAC
        LAC     ONE,RXINT_BIT
        ADD     ONE,TXINT_BIT
        SACL    TMP
        OUT     TMP,FCLR            ; CLEAR INTERRUPT FLAGS FOR TX AND RX
        RET                         ; RETURN TO CALLING ROUTINE
*
        END
```

Section 6

Hardware Applications

The TMS320C14/E14 has been designed for a wide range of applications in digital signal processing and digital control. A large number of on-chip peripherals have been integrated into the TMS320C14/E14 giving it direct-connect capability with various external devices. This can reduce and even eliminate interfacing hardware.

Major hardware applications discussed in this section are listed below.

The following buses, ports, and control signals provide system interfacing to the TMS320C14/E14.

- 12-bit address bus (A11 - A0)
- 16-bit data bus (D15 - D0)
- 3-bit port address bus (PA2 - PA0)
- Enable signals; read enable ($\overline{\text{REN}}$) and write enable ($\overline{\text{WE}}$)
- Memory control and non-maskable interrupt signals ($\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$) signals.
- Interrupt ($\overline{\text{INT}}$)
- 16 bit I/O pins (IOP15-IOP0)
- Serial port (RXD/DATA, TXD/CLK, TCLK1/CLKR, TCLK2/CLKX, CMP4/CAP2/FSR, CMP5/CAP3/FSX)
- Timers (WDT, TCLK/CLKR, TCLK2/CLKX)
- Event manager (CMP0-CMP3, CAP0-CAP1, CMP4/CAP2/FSR, CMP5/ CAP3/FSX)
- Reset ($\overline{\text{RS}}$)

6.1 System Control Circuitry

The system control circuitry performs functions that are critical for proper system initialization and operation. A powerup reset circuit design and a crystal oscillator circuit design are presented in this section. The powerup reset circuit assures that a reset of the part occurs only after the oscillator is running and stabilized.

6.1.1 Powerup Reset Circuit

The reset circuit shown in Figure 6-1 performs a powerup reset; i.e., the TMS320C14/E14 is reset when power is applied. Note that the switch circuit may include debounce circuitry. Driving the $\overline{RS}$ signal low initializes the processor. Reset affects several registers and status bits. (refer to Section 3.2.2 for a detailed description of the effect of a reset on processor status).

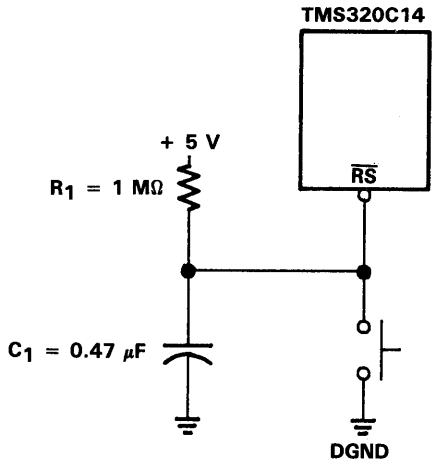


Figure 6-1. Powerup Reset Circuit

For proper system intialization, the reset signal must be applied for at least five CLKOUT cycles, i.e., 800 ns for a TMS320C14/E14 operating at 25.6 MHz.

Upon powerup, it can take up to one hundred milliseconds before the system oscillator reaches a stable operating state. Therefore, the powerup reset circuit should generate a low pulse on the reset line until the oscillator is stable (i.e., 100 to 200 ms).

The voltage on the reset pin ($\overline{\text{RS}}$) is controlled by the R_1C_1 network (see Figure 6-1). After a reset, this voltage rises exponentially according to the time constant R_1C_1, as shown in Figure 6-2.

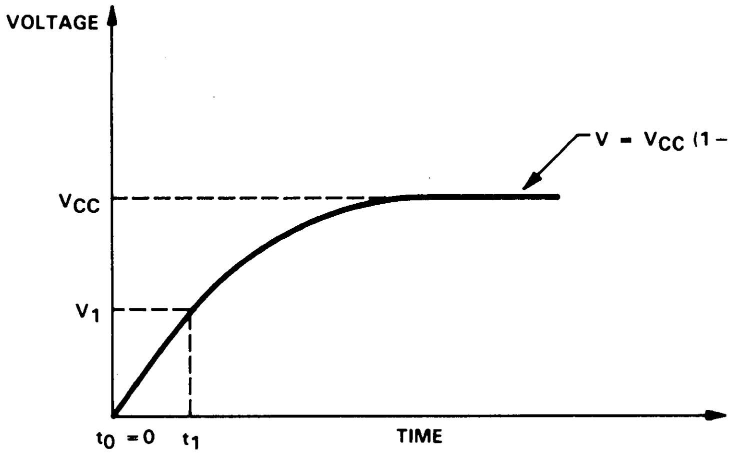


Figure 6-2. Voltage on TMS320C14/E14 Reset Pin

The duration of the low pulse on the reset pin is approximately t_1, which is the time it takes for capacitor C_1 to be charged to 1.5 volts. This is approximately the voltage at which the rest input switches from a logic level 0 to a logic level 1. The capacitor voltage is given by following formula:

$$V = V_{CC}\left[1 - e - \frac{t}{\tau}\right]$$

where $\tau = R_1C_1$ is the reset circuit time constant. Solving (1) for t is given in the formula:

$$t = -R_1\,C_1 \ln\left[1 - \frac{V}{V_{CC}}\right]$$

For example, the following values of:

$R_1 = 1\ \text{M}\Omega$	$V_{CC} = 5\ \text{V}$
$C_1 = 0.47\ \mu\text{F}$	$V = V_1 = 1.5\ \text{V}$

yields $t = t_1 = 167$ ms. In this case, the reset circuit of Figure 6-1 can generate a low pulse of long enough duration (167 ms) to ensure the stabilization of the oscillator upon powerup in most systems.

6.1.2 Crystal Oscillator Circuit

The TMS320C14/E14 requires an external clock source to drive the CLKIN pin. Either a prepackaged oscillator can be used or a simple oscillator circuit using TTL gates can be built. Prepackaged oscillators provide a wide operating range and better stability. A well designed crystal oscillator, however, will provide good performance with TTL gates. Two types of crystal oscillator circuits can be used; one with series resonance, or one with parallel resonance.

Figure 6-3 shows implementation of a parallel resonant oscillator circuit. The circuit is designed to use the fundamental frequency of the crystal. The 74AS04 inverter provides the 180-degree phase shift that a parallel oscillator requires. The 4.7 KΩ resistor provides the negative feedback for stability, i.e., the poles of the system are constrained in a narrow region about the Jω axis of the s-plane (analog domain). The 10 KΩ potentiometer is used to bias the 74AS04 in the linear region.

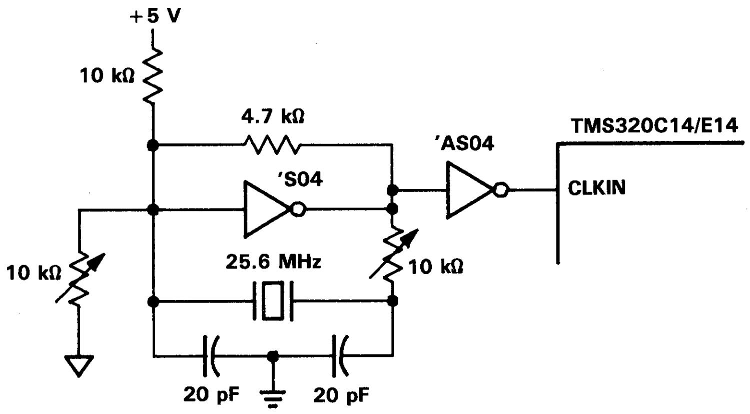


Figure 6-3. Parallel Resonant Crystal Oscillator Circuit

Figure 6-4 shows a series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter provides a 180-degree phase shift in a series resonant oscillator circuit. The 330 Ω resistors provide the negative feedback for stability.

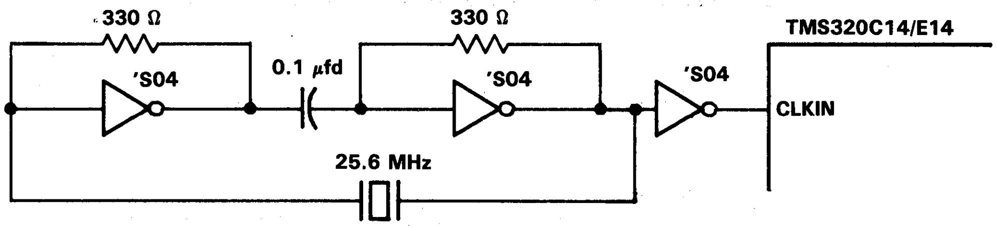
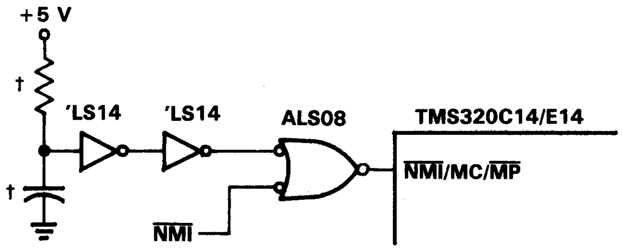


Figure 6-4. Series Resonant Crystal Oscillator Circuit

6.1.3 MC/MP Mode Configurations

The TMS320C14/E14 can be configured (by software or hardware) to address internal or external program memory. The software technique uses the MC/MP bit in the SYSCON register to switch between internal and external memory. The hardware technique uses the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin which is sensed during reset. If the pin is low, the device is put in the microprocessor (external memory) mode. If the pin is high, the device is put in the microcomputer (internal memory).

Figure 6-5 shows a circuit for configuring the device into the microprocessor mode using a hardware technique. The $\overline{\text{RS}}$ and $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pins cannot be tied together since the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin must be held low for 1.25 clock cycles (200 µs in the circuit below) after $\overline{\text{RS}}$ goes high. If the NMI function is not being used, however, the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin may be tied to ground. This will not produce an interrupt since the NMI function is edge-triggered.



†Values depend on reset timing. To operate with this circuit, values of 1 MΩ and 1.33 µF would be used.

Figure 6-5. Mode Control Circuit

6.2 External Memory Interfacing

The TMS320C14/E14 can be interfaced with PROMs, EPROMs, and RAMs. The TMS320C14/E14 has no provisions to insert wait states, all memory devices should therefore meet the full speed access requirements of the device.

The TMS320C14/E14 implements two kinds of memory spaces, program memory (4K words) and data memory (256 words). Data memory accesses are always from internal RAM and no off-chip accesses are available. Program memory accesses can be configured from either on-chip ROM/EPROM or off-chip memory. $\overline{\text{REN}}$ (read enable) and $\overline{\text{WE}}$ (write enable) strobes are active whether accessing on-chip or off-chip program memory.

The following discussion describes the TMS320C14/E14 read and write cycles. A program memory write cycle occurs when a TBLW instruction is executed. For timing diagrams, refer to the TMS320C14/E14 data sheet. Memory interfaces assume that the TMS320C14/E14 is running at 25.6 MHz and Q is 39.1 ns. Q is used to indicate one quarter cycle of CLKOUT frequency.

In the read cycle, the following sequence occurs (refer to Figure 6-6):

1) CLKOUT goes low, terminating the previous bus cycle and starting the next cycle.

2) Address becomes valid after a delay of no more than t_{d1}. This is followed by $\overline{\text{REN}}$ going low after a delay of t_{d2} to indicate the start of a read cycle.

3) External device is selected with a minimum address setup time of t_{su}(A-REN) and puts it's data on the bus.

4) CLKOUT goes low to terminate the cycle and data is sampled at the falling edge of CLKOUT. Data has to be valid for at least $t_{su(D)}$ prior to CLKOUT going low, and must be held valid for at least t_h(D).

5) $\overline{\text{REN}}$ goes high after a delay of no more than t_{d3} and the address bus becomes invalid after a delay of no more than $t_{h(A\text{-}WR)}$.

For read only devices, the maximum access time from address valid until data valid is:

$$\begin{aligned} t_c(C) - t_d1 - t_{su:}(D) &= 156.25 - 40 - 40 \\ &= 76.25 \text{ ns} \end{aligned}$$

From chip enable until data valid is:

$$\begin{aligned} t_c(C) - t_d2 - t_{su}(D) &= 156.25 - (1/4\, t_c(C) + 12) - 40 \\ &= 156.25 - 39.1 - 12 - 40 \\ &= 65.15 \text{ ns} \end{aligned}$$

The minimum address setup time prior to chip enable will be:

$$\begin{aligned} t_{su}(\text{A-REN}) &= 1/4t_c(C) - 35 \\ &= 4.1 \text{ ns} \end{aligned}$$

Therefore for read access, memories with a maximum access time of 65 ns from chip enable are required.

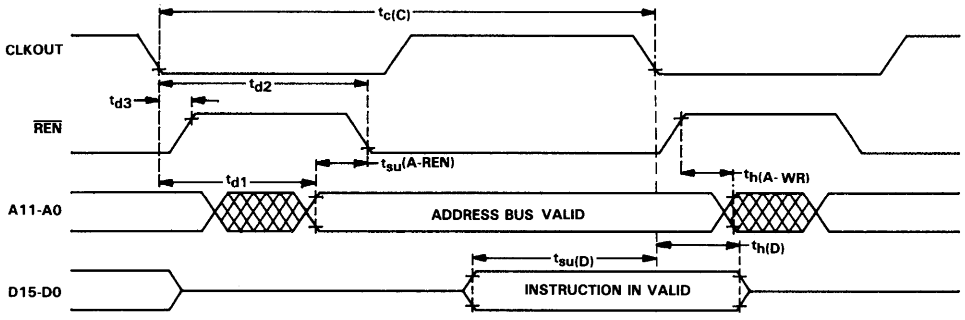


Figure 6-6. Memory Read Timing

In the write cycle (executing a TBLW), the following sequence occurs (refer to Figure 6-7):

1) CLKOUT goes low, terminating the previous cycle and starting the write cycle for the TBLW instruction.

2) Address becomes valid after a delay of t_{d1}.

3) Data bus is driven after a delay of t_d9, and data becomes valid after a delay of no more than t_{d8}.

4) $\overline{\text{WE}}$ goes low in the second half of the cycle after a delay of t_{d6}.

5) CLKOUT goes low to terminate the cycle.

6) $\overline{\text{WE}}$ goes high after a delay of t_{d7} and address bus becomes invalid after a delay of t_v.

7) Data bus is driven invalid after a delay of no more than t_{d10}.

For RAM devices, the maximum access time from address valid until data valid is:

$$\begin{aligned} t_d8 - t_d1 &= 1/4t_c(C) + 52 - 40 \\ &= 39.1 + 12 \\ &= 51.1 \text{ ns} \end{aligned}$$

From chip enable until data valid is:

$$\begin{aligned} t_d8 - t_d6 &= 1/4t_c(C) + 52 - 1/2t_c(C) - 12 \\ &= 1/4t_c(C) + 40 \\ &= 39.1 - 40 \\ &= 0.9 \text{ ns} \end{aligned}$$

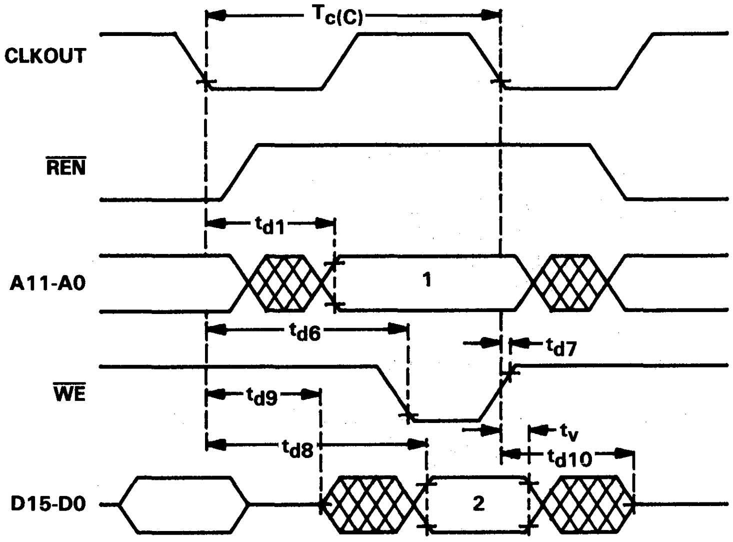
The write enable access time is:

$$\begin{aligned} t_c(C) - t_d6 - t_d7 &= 156.25 - 1/2t_c(C) - 12 - 12 \\ &= 78.1 - 2.4 \\ &= 54.1 \text{ ns} \end{aligned}$$

The minimum data hold time provided is:

$$\begin{aligned} t_v - t_d7 &= 1/4t_c(C) - 10 - 12 \\ &= 39.1 - 22 \\ &= 17.1 \text{ ns} \end{aligned}$$

Therefore for write access, memories with a maximum access time of 50 ns from chip enable are required.



1 = TBLW ADDRESS
2 = TBLW DATA

Figure 6-7. Memory Write Timing (TBLW Instruction)

6.2.1 Program ROM Expansion

Twelve output pins (A11-A0) are available for addressing external memory. They contain either the buffered outputs of the program counter or the I/O port address.

Read operations are performed on external memory either during opcode or operand fetches or during the execution of a TBLR (table read) instruction. Write operations have no effect on the circuit. When a read operation occurs, an address is placed on the address bus, and the $\overline{\text{REN}}$ (read enable) strobe is generated by driving $\overline{\text{REN}}$ low to enable external memory. The instruction word is then transferred to the TMS320C14/E14 via the 16-bit data bus.

A memory address being placed on the bus becomes valid following a maximum delay (t_{d1}) from the falling edge of CLKOUT. The combined delay of:

$$t_{d1} + t_{a(A)} + t_{su(D)} = \text{minimum cycle time } t_{c(C)}$$

where $t_{a(A)}$ = memory access time of EPROM from address valid
$t_{su(D)}$ = setup time from data bus valid prior to CLKOUT↓

serve as the timing constraints used when calculating $t_{c(C)}$.

When only external program ROM is required, a minimum system can consist of a TMS320C14/E14 and up to 4K words of external program memory (TMS27C292), as shown in Figure 6-8. The $\overline{\text{REN}}$ signal and the address (A11-A0) and data (D15-D0) lines on the TMS320C14/E14 are connected directly to the TMS27C292 memories, and no address decoding is required. The memories used are a pair of Texas Instruments TMS27C292 4K x 8 ROMs, configured in parallel for a direct 16-bit interface to the TMS320C14/E14.

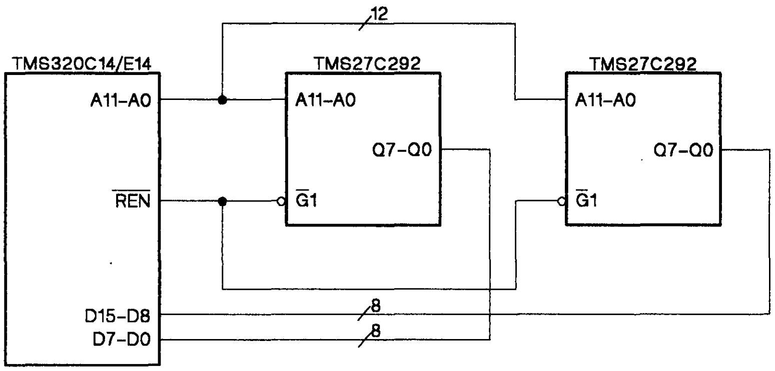


Figure 6-8. Minimum Program ROM Expansion

6.2.2 Data RAM Expansion

No direct memory expansion is provided on the TMS320C14/E14. However, if RAM is used for external program memory, this memory can be used to store data information, accessed using the TBLR and TBLW instructions. These instructions, however, take three cycles to execute.

If larger memory or faster memory accesses are required, an alternative memory expansion scheme using I/O ports can be implemented for a TMS320C14/E14 device. In this case, additional RAM can be used to supplement internal data memory, and can be accessed in only two cycles using the IN and OUT instructions. If RAM is to be used for program memory, additional logic must be included to distinguish between an I/O write (OUT) and a program memory write (TBLW).

Figure 6-9 provides an example of external data memory expansion. The design consists of up to 16K words of static RAM (IMS1420), addressed by the lower 14 bits of a 16-bit counter (74ALS193). In the case of the IMS1420s, the address of the data to be accessed is loaded into the counter by implementing an OUT instruction to port 0. This loads the data bus into the counters. Memory can then be read from or written to sequentially by doing an IN or OUT instruction to port 1. The MSB in the counters determines whether the memory address is incremented (MSB = 0) or decremented (MSB = 1) after a read or write of data memory. Memory continues to be addressed sequentially until new data is loaded into the counters.

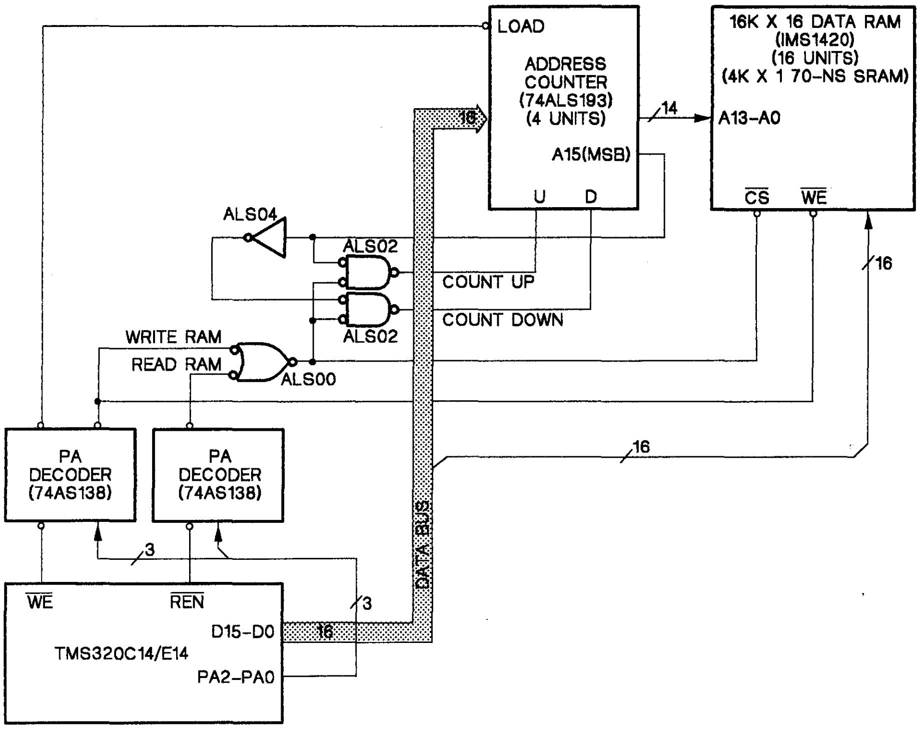


Figure 6-9. Data RAM Expansion

Dynamic memories may also be used; however, these devices may impose additional constraints on the system designer. For example, some memory cycle times may not allow consecutive IN/OUT/IN instruction sequences. Memory refresh must also be considered. Since the TMS320C14/E14 does not implement "wait" states, memory refresh must be generated transparent to the processor.

6.3 External Peripheral Interfacing

The TMS320C14/E14 is flexible enough to be used in a wide range of applications requiring different types of peripheral interfaces. The following sections describe A/D-D/A interfacing, an AIC interface, a Codec interface, and optical encoder application, a serial communication interface, and XDS interface considerations.

6.3.1 A/D Interface

The TMS320C14/E14 can be interfaced to an A/D (analog-to-digital) converter to perform the necessary conversions. A minimum of external circuitry is required.

Figure 6-10 shows an interface of the TLC0820 8-bit A/D converter to the TMS320C14/E14. Since the control circuitry of the TLC0820 operates much more slowly than the TMS320C14/E14, it cannot be directly interfaced. All of the logic functions are implemented with one each of the following devices from the 74ALS family of Advanced Low-power Schottky Logic:

74ALS679	12-bit address comparator
74LS74	Dual positive edge-triggered D-type flip-flops
74ALS465	Octal buffer with three-state output

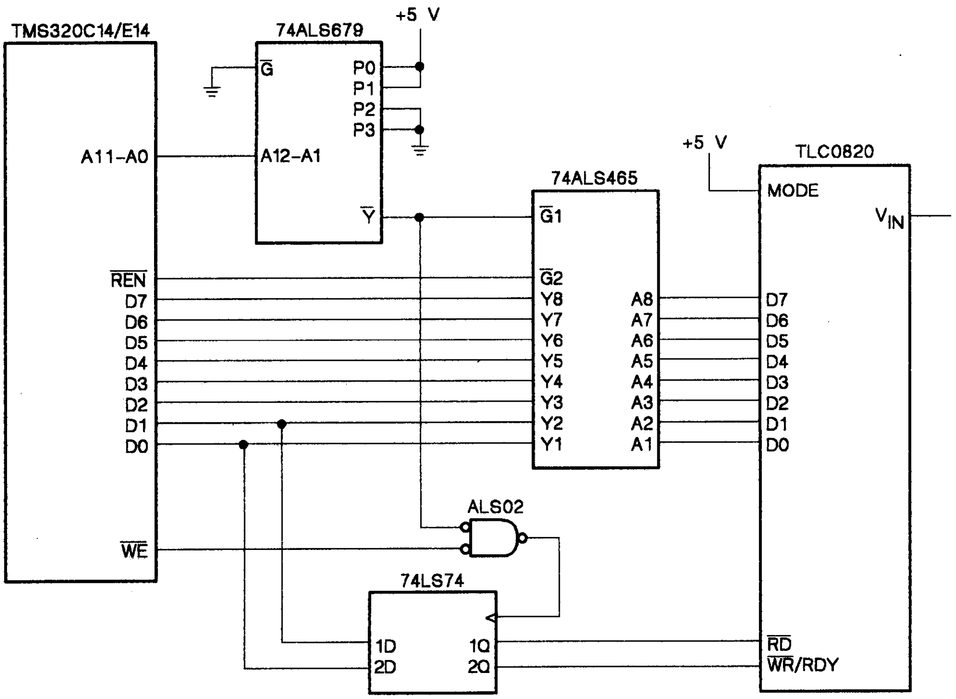


Figure 6-10. A/D Converter to TMS320C14/E14 Interface

6.3.2 D/A Interface

An interface of the TLC7524 8-bit D/A converter to the TMS320C14/E14 is shown in Figure 6-11. Due to the high-speed operation of the internal logic circuitry of the TLC7524, the interface to the TMS320C14/E14 requires external logic circuitry to decode the address of the peripheral. Here a 74ALS679 12-bit address comparator is used.

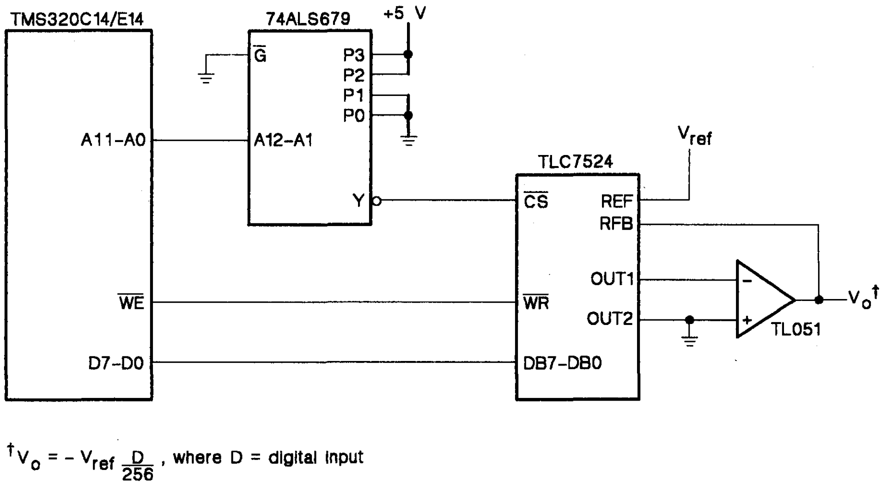


Figure 6-11. D/A Converter to TMS320C14/E14 Interface

For further information about the A/D and D/A converters shown in the figures, refer to the *Linear Circuits Data Book (SLYD001)*.

6.3.3 Codec Interface

Some areas of speech, telecommunications, and many other applications require low-cost analog-to-digital (A/D) and digital-to-analog (D/A) converters. Combo-codecs are most effective in serving DSP system data-conversion requirements. Combo-codecs are single-chip pulse-code-modulated encoders and decoders (PCM codecs), designed to perform the encoding (A/D conversion) and decoding (D/A conversion), as well as the antialiasing and smoothing filtering functions. Since combo-codecs perform these functions in a single 300-mil DIP package at low cost, they are extremely economical for providing system data-conversion functions.

Combo-codecs interface directly to the TMS320C14/E14 by means of the serial port and provide a companded, PCM-coded digital representation of analog input samples. This PCM code is easily translated into linear form by the TMS320C14/E14 for use in processing. The design discussed here and shown in Figure 6-12 uses a Texas Instruments TCM29C13 codec, interfaced using the serial port of the TMS320C14/E14.

The TMS320C14/E14 serial port provides direct synchronous communication with serial devices. The interface signals are compatible with codecs and other serial components so that minimal external hardware is required. Externally, the serial port interface is implemented using the following pins on the TMS320C14/E14:

- TXD/CLK (transmitted serial data)
- TCLK2/CLKX (transmit clock)
- CMP5/CAP3/FSX (transmit framing synchronization signal)
- RXD/DATA (received serial data)
- TCLK1/CLKR (receive clock)
- CMP4/CAP2/FSR (receive framing synchronization signal)

Data on TXD and RXD are clocked by CLKX and CLKR, respectively. These clocks are only required during serial transfers on the TMS320C14/E14.

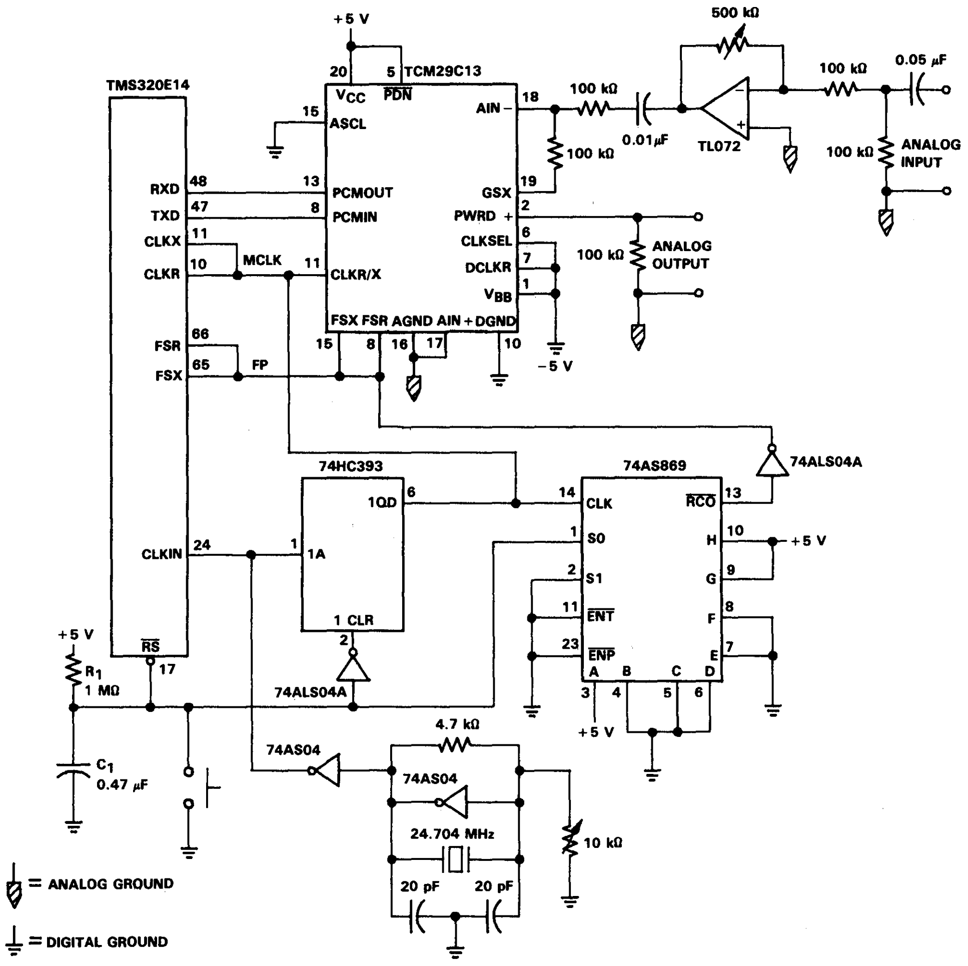


Figure 6-12. Interface of TMS320C14/E14 to TCM29C13 Codec

Serial port transfers are initiated by framing pulses on the FSX and FSR pins for transmit and receive operations, respectively. For transmit operations, the FSX pin can be configured as an input or an output. This option is selected by the SCON register bit 13 (SPC1). In this design, FSX is assumed to be configured as an input; therefore, transmit operations are initiated by a framing pulse on the FSX pin. Upon completion of receive and transmit operations, an RINT (serial port receive interrupt) and an XINT (serial port transmit interrupt) are generated, respectively. The serial port is doubled buffered and allows continuous reception and transmission.

The TMS320C14/E14 interfaces directly to the codec, as shown in Figure 6-12, with no additional logic required. The PCM μ-law data generated by the codec at the PCMOUT pin is read by the TMS320C14/E14 from the data receive (RXD) pin, which is internally connected to the receive serial register (RSR). The data transmitted from the data transmit (TXD) pin of the TMS320C14/E14 is received by the PCMIN input of the codec. During the digital-to-analog conversion, this μ-law companded data must be converted back to a linear representation for use in the TMS320C14/E14. The resulting analog waveform is lowpass-filtered by the codec's internal smoothing filter. Therefore, no additional filtering is required at the codec output (PWRO+). Software companding routines appropriate for use on the TMS320C14/E14 are provided in the book, *Digital Signal Processing Applications with the TMS320 Family*.

A combo-codec configured in the fixed-data-rate mode requires the following external clock signals:

- A 1.544 MHz clock to be used as the master clock, and
- 8-kHz framing pulses to initialize the data transfers.

To generate the 1.544 MHz master clock for the combo-codec, a division by 16 of the 24.704 MHz system clock is required. The 74HC393 contains two divide-by-16 counters.

The 74AS869 is configured to generate the 8-kHz clock pulse (the ripple carry output is 1.544 MHz/193 = 8 kHz). This pulse is used by the TMS320C14/E14 and codec as a framing pulse to initiate data transfers.

The level of the analog input signal is controlled using the TL072 opamp connected in the inverting configuration (see Figure 6-12). Using the 500-kΩ potentiometer, the gain of this circuit can be varied from 0 to 5. The output of the 0.01-μF coupling capacitor drives the TCM29C13's internal op amp. This op amp is connected in the inverting configuration with unity gain (feedback and input impedances having the same value of 100 kΩ).

6.3.4 RS-232 Interface

The TMS320C14/E14 allows for implementation of an RS-232 interface for connection to communication equipment, terminals, and PC's. The only devices needed are line drivers/receivers for the TTL/RS-232 level conversions. Figure 6-13 shows a typical interface with the TMS320C/E14 as a data terminal equipment (DTE) device connected to a data communication equipment (DCE) device (such as a modem). The serial port is configured for the asynchronous mode with the appropriate parameters selected. The bit I/O pins IOP15-IOP0 are used to provide any necessary handshaking signals (i.e., RTS, CTS, DSR, and DTR).

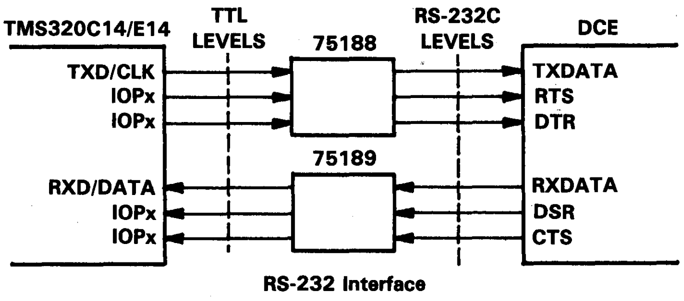


Figure 6-13. RS-232 Interface

6.3.5 Optical Encoder Interface

Optical encoders are commonly used as sensors in measuring both speed and position. The capture inputs of the TMS320C14/E14 allow direct interfacing with optical encoders. A very cost-effective noise filtering system can be built with a couple of gates, D latches, and RC filters. The one-shot constructed of XOR generates pulses on each edge of each signal. Pulses generated by phase A are used to clock the signal from phase B, and vice versa. Figure 6-14 shows a typical interface with an optical encoder.

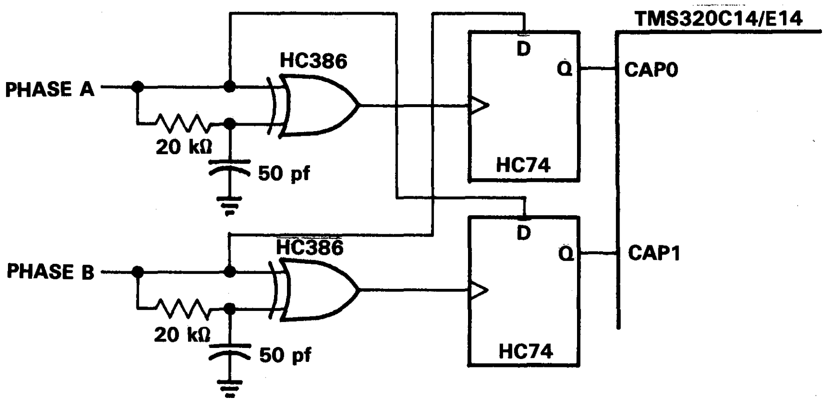


Figure 6-14. Optical Encoder Interface

6.3.6 XDS Design Considerations

The TMS320C14 XDS Emulator is implemented with a dual processor system incorporating a TMS320C14 and a TMS9996. The TMS9996 acts as controller in the system, while the TMS320C14 performs the emulation. The design of the XDS maximizes performance and allows full speed in-circuit emulation. This discussion covers general design considerations as well as timing and loading.

6.3.6.1 Bus Control

When the emulator is halted from the keyboard or by a breakpoint condition, the current state of the TMS320C14 is extracted by the TMS9996 implemented to look like a co-processor. The TMS9996 communicates with the TMS320C14 over the internal data bus (of the emulated device) which is not seen by the user. Additional communication between the two processors is generated with commands entered from the keyboard. The TMS9996 shares the data bus only when the $\overline{\text{REN}}$ and $\overline{\text{WE}}$ signals are high. The target system should drive the data bus **ONLY** when devices on the target system are addressed and $\overline{\text{REN}}$ or $\overline{\text{WE}}$ is low. If these rules are violated, the XDS gives a "PROCESSOR SYNC LOST" error message #1185. This error may also be caused by signal-to-signal shorts in the target system, misalignment of the target connector, or wiring errors in the target system.

6.3.6.2 XDS Timing

The TMS320C14/E14 emulator has additional logic in series with the inputs $\overline{\text{RS}}$, $\overline{\text{INT}}$, $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$, and outputs $\overline{\text{REN}}$ and $\overline{\text{WE}}$. Propagation delays assume 3 ns for cable delay and capacitive loading. The delays outlined below are in addition to device specifications.

Table 6-1. XDS/Target Device Timing Delays

Signal	Delay
$\overline{\text{INT}}$,$\overline{\text{NMI}}$,$\overline{\text{RS}}$	Asynchronous
$\overline{\text{WE}}$, $\overline{\text{REN}}$, PA2-PA0, CLKOUT	Delay from part to pin = 7 ns.
CLKIN	PLCC target cable delay from pin to part = 20 ns
Other Signals	Delay from part to pin or pin to part = 3 ns

6.3.6.3 Reset

The Reset ($\overline{\mathrm{RS}}$) signal is synchronized on the rising edge of CLKOUT after being sent from the pin through a 16L8 PAL. The reset signal is applied to the TMS320C14/E14 on the rising edge of CLKOUT. This is done to implement a run on application of target reset function in the emulator. It is not necessary for the user to synchronize $\overline{\mathrm{RS}}$ in their design.

6.3.6.4 Emulator Loading

Additional loading on the outputs is induced by the XDS. The differences in the DC loading between the emulator and the device are defined below.

Table 6-2. XDS/Device DC Loading

Signal	Load
$\overline{\mathrm{RS}}$	0.3 ma
$\overline{\mathrm{NMI}}$/MC/$\overline{\mathrm{MP}}$	0.6 ma
D15-D0	1.0 ma
$\overline{\mathrm{REN}}$, $\overline{\mathrm{WE}}$	0.3 ma

6.3.6.5 Miscellaneous Considerations

The emulator coprocessor initially sets up the device being emulated to run and then does not attempt to communicate with the emulated device until the user communicates with the emulator via the keyboard. If the target system is continuously asserting $\overline{\mathrm{RS}}$, the coprocessor does not gain control of the device and will report a "PROCESSOR SYNC LOST" error message 1185. This condition can be caused by a powered-up emulator plugged into a powered-down target system. Even though $\overline{\mathrm{RS}}$ is pulled up through a resistor on the emulator, the impedance of the target system powered off can be low enough to assert a low on $\overline{\mathrm{RS}}$ or load the data bus so as to keep the emulator from functioning.

Caution:

The conductive foam on the XDS target connector and logic show cable must be removed before power-up. Failure to do so will result in a malfunction and may cause damage.

6.3.6.6 *Transmission Line Phenomena*

Since the XDS target cable is approximately 20 inches long, use of advanced CMOS or Fast/Advanced Schottky TTL may cause line reflections (ringing above input thresholds) on input lines to the XDS. Series termination resistors (22 to 68 ohms) can help eliminate this problem. Generally, the user should use the cables as supplied, and keep connections as short as possible.

6.3.6.7 *Clock Source*

The XDS Emulator only supports the use of either an internal oscillator or a TTL clock source provided by the user's target system. The emulator clock is selectable from three sources:

1) The target clock
2) A socketed, changeable crystal (Y2) on the Processor Module (PM) board
3) A socketed, changeable canned TTL oscillator (U1) on the PM board

6.4 System Applications

The TMS320C14/E14 can be used in a wide variety of applications. The fast CPU and associated circuitry allow the device to used in computation intensive applications requiring processing of signals in control systems, speech systems, telecommunications, FFT's, and filtering systems. The TMS320C14/E14 goes a step further than previous DSP's by including a large amount of peripheral circuitry. These integrated modules reduce (and may even eliminate) the amount of "glue" circuitry normally required with other DSP's, thus providing a cost effective solution to designers in many applications. Included in this section are discussions of some example applications that the TMS320C14/E14 is suited for.

6.4.1 Disk Drive Control

The implementation of the TMS320C14/E14 for disk drive control is shown in Figure 6-15. In this example, both the read/write actuator and the spindle motor are controlled by the DSP. This illustrates an application where two sets of linear functions are provided.

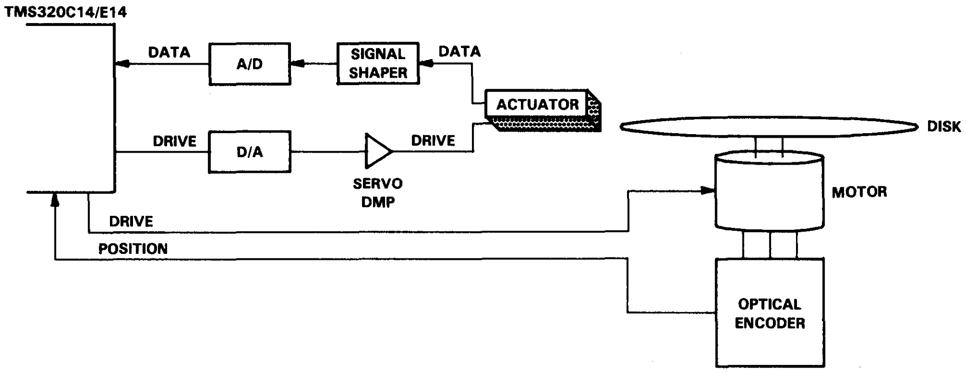


Figure 6-15. Disk Drive Co.،.rol

Using sophisticated algorithms, the TMS320C14/E14 may perform the following functions:

- Precisely control the read/write actuator for fast access time.
- Cancel mechanical resonance.
- Estimate and control actuator speed.

Position information is read from the disk surface and converted into digital form by an A/D. A D/A provides conversion for the position control signal to the actuator. The PWM outputs directly control the speed of the spindle motor, while the capture inputs accept the optical encoder signals indicating velocity.

6.4.2 Plotter Control

Figure 6-16 shows the TMS320C14/E14 used as a pen position controller for a plotter. The capture inputs receive position signals from the optical encoders of the the position rotors, and the PWM outputs provide the controlling signals to the X-Y drive motors.

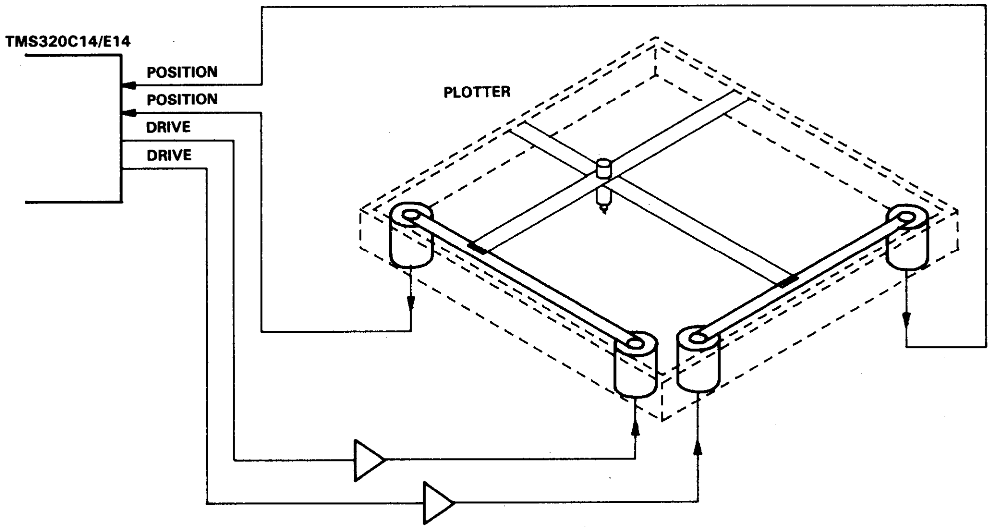


Figure 6-16. Plotter Control

6.4.3 Tape Drive Control

The TMS320C14/E14 is illustrated in Figure 6-17 as providing servo control of a magnetic tape drive mechanism. The PWM outputs generate the signals for controlling the tension and takeup motors, while the optical encoder sensors provide input to the capture inputs. The DSP is responsible for keeping the speed of the tape constant, as well as maintaining proper tape tension. Tape speed may also be detected by reading embedded information off the tape. Tape tension is detected by the position of lever arms attached to servo motors.

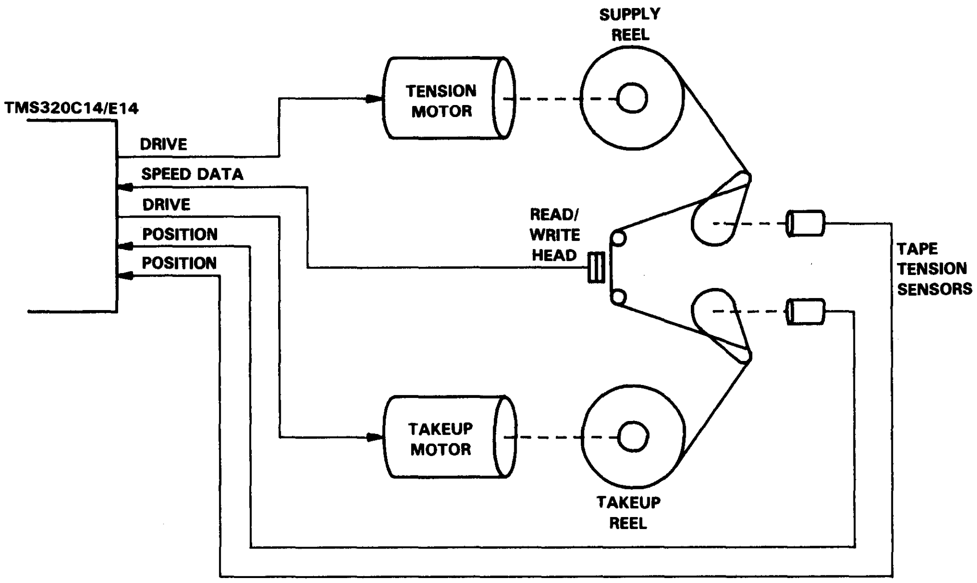


Figure 6-17. Tape Drive Control

6.4.4 AC Motor Control

With the TMS320C14/E14, it is possible to provide cost effective servo control of either AC induction and synchronous motors or DC brushless motors. AC motors are cheaper to manufacture and easier to maintain than DC servo motors. Their control structure, however, is much more complex than DC servo motors. Vector rotation techniques are used to transform the coordinates and simplify the control structure of an AC motor to field controlled DC motor. The TMS320C14/E14 can be used to perform the necessary computations that will permit vector control of AC motors. The PWM outputs can be used to control the multiple phases. Speed information can be obtained from optical encoders. Figure 6-18 shows implementation of control of an AC motor.

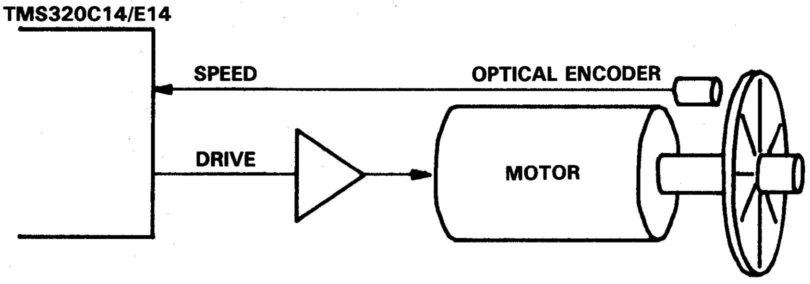


Figure 6-18. AC Motor Control

Appendix A

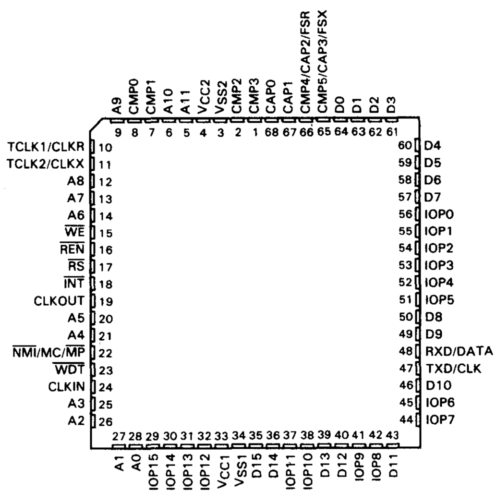
TMS320C14/E14 Data Sheet

TMS320C14/TMS320E14 DIGITAL SIGNAL PROCESSORS

DECEMBER 1988

- 160-ns Instruction Cycle
- 256-Word On-Chip Data RAM
- 4K-Word On-Chip Program ROM (TMS320C14)
- 4K-Word On-Chip Program EPROM (TMS320E14)
- EPROM Code Protection for Copyright Security
- 4K-Word Total External Memory at Full Speed (Microprocessor Mode)
- 32-Bit ALU/Accumulator
- 16 × 16-Bit Multiplier with a 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- Seven Input and Seven Output External Ports
- 16-Bit Bidirectional Data Bus with Greater Than 50-Mbps Transfer Rate
- Bit-Selectable I/O Port (16 Pins)
- Serial Port with Programmable Protocols
- Event Manager with Capture Inputs and Compare Outputs
- Four Independent Timers (Watchdog, General Purpose [2], Serial Port)

TMS320C14, TMS320E14
FN AND FZ PACKAGES
(TOP VIEW)



- Single 5-V Supply
- Packaging: 68-Pin PLCC or CLCC
- 15 Internal/External Interrupts

ADVANCE INFORMATION

introduction

This data sheet provides complete design documentation for the TMS320C14 and TMS320E14 devices, which are a part of the First Generation TMS320 family. The TMS32010, the first digital signal processor of the TMS320 family, was introduced in 1982. Its powerful instruction set, inherent flexibility, high speed number-crunching capabilities, and innovative architecture have made this high performance, cost-effective processor the ideal solution for many commercial, industrial and military applications. Since that time, three generations of the TMS320 family have evolved, each with its own group of related devices. All TMS320 devices combine the flexibility of a high speed controller with the numerical capability of an array processor. This offers an inexpensive alternative to multichip bit-slice processors.

The TMS320C14/E14 devices are 16/32-bit single-chip digital signal processors that are object-code compatible with the TMS32010 device. This allows hardware upgrading without the expense of software re-development. The highly paralleled architecture and efficient instruction set provide the speed and flexibility to execute 6.4 million instructions per second (MIPS). The TMS320C14/E14 devices contain several on-chip peripherals that can reduce and even eliminate interface components and "glue" circuitry, allowing use in space-critical applications.

The TMS320C14/E14 is offered in a 68-pin plastic leaded chip carrier package (FN suffix) rated for operation from 0°C to 70°C (L suffix). It is also offered in a 68-pin ceramic leaded chip carrier package (FZ suffix) carrier rated for operation from 0°C to 70°C (L suffix).

This data sheet is divided into the following major sections: introduction, functional block diagram, architecture, instruction set, development, support products, documentation support, electrical and timing specifications, timing diagrams, and EPROM programming. An index is provided for quick reference to specific information.

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 ● HOUSTON, TEXAS 77001

pin descriptions

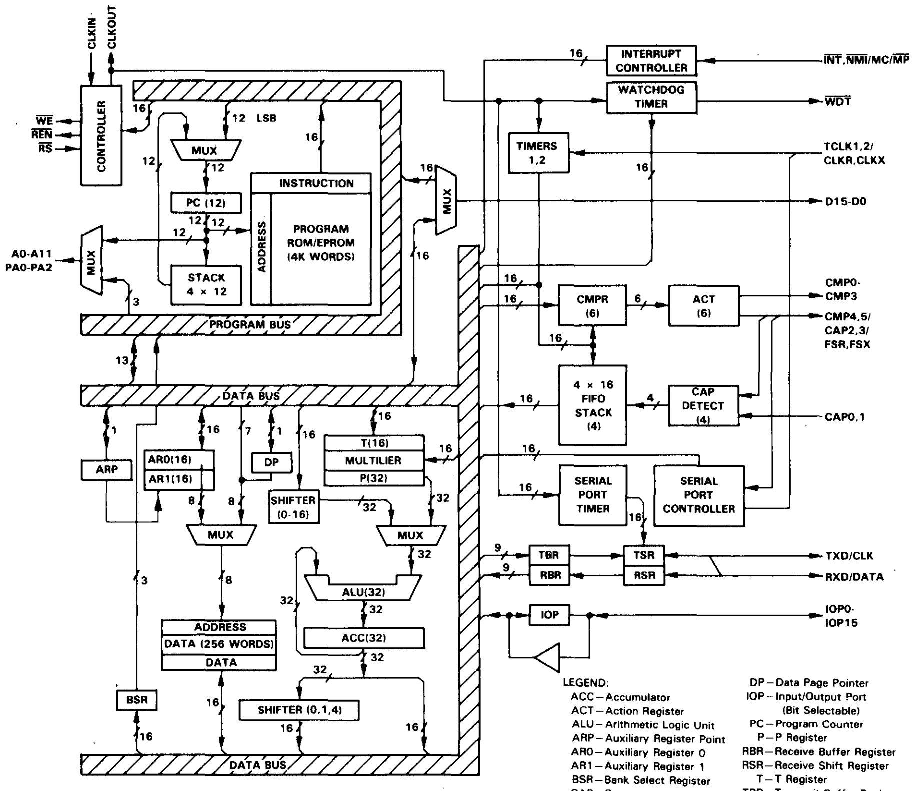
PIN		I/O/Z[†]	DESCRIPTION
NAME	NO.		ADDRESS/DATA BUSES
A11 A10 A9 A8 A7 A6 A5 A4 A3 A2/PA2 A1/PA1 A0/PA0	5 6 9 12 13 14 20 21 25 26 27 28	O/Z	Program memory address bus A11 (MSB) through A0 (LSB) and port addresses PA2 (MSB) through PA0 (LSB). Addresses A11 through A0 are always active and never go to high impedance except during reset. During execution of the IN and OUT instructions, pins 26, 27, and 28 carry the port addresses. Pins A3 through A11 are held high when port accesses are made on pins PA0 through PA2.
D15 MSB D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 LSB	35 36 39 40 43 46 49 50 57 58 59 60 61 62 63 64	I/O/Z	Parallel data bus D15 (MSB) through D0 (LSB). The data bus is always in the high-impedance state except when $\overline{\text{WE}}$ is active (low). The data bus is also active when internal peripherals are written to.
			INTERRUPT AND MISCELLANEOUS SIGNALS
$\overline{\text{INT}}$	18	I	External interrupt input. The interrupt signal is generated by a low signal on this pin.
$\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$	22	I	Non-maskable interrupt. When this pin is brought low, the device is interrupted irrespective of the state of the INTM bit in status register ST. Microcomputer/microprocessor select. This pin is also sampled when $\overline{\text{RS}}$ is low. If high during reset, internal program memory is selected. If low during reset, external memory will be selected.
$\overline{\text{WE}}$	15	O	Write enable. When active low, $\overline{\text{WE}}$ indicates that device will output data on the bus.
$\overline{\text{REN}}$	16	O	Read enable. When active low, $\overline{\text{REN}}$ indicates that device will accept data from the bus.
$\overline{\text{RS}}$	17	I	Reset. When this pin is low, the device is reset and PC is set to zero.
			SUPPLY/OSCILLATOR SIGNALS
CLKOUT	19	O	System clock output (one fourth CLKIN frequency).
V_{CC}	4,33	I	5-V supply pins.
V_{SS}	3,34	I	Ground pins.
CLKIN	24	I	Master clock input from external clock source.

[†]Input/Output/High-impedance state.

pin descriptions (concluded)

PIN		I/O	DESCRIPTION
NAME	NO.		SERIAL PORT AND TIMER SIGNALS
RXD/DATA	48	I/O	In the asynchronous and codec modes, this pin is the receive input. In the synchronous mode, this pin is data in while receiving data, and data out while transmitting data.
TXD/CLK	47	I/O	In the asynchronous and codec modes, this pin is the transmit output. In the synchronous mode, this pin is clock input with external clock, and clock output with internal clock.
TCLK1/CLKR	10	I	Timer 1 clock. If external clock is selected, it serves as clock input to Timer 1. Can also be configured as serial port receive clock in codec mode.
TCLK2/CLKX	11	I	Timer 2 clock. If external clock is selected, it serves as clock input to Timer 2. Can also be configured as serial port transmit clock in codec mode.
$\overline{\text{WDT}}$	23	O	Watchdog timer output. An active low is generated on this pin when the watchdog timer times out.
			BIT I/O PINS
IOP15 MSB	29	I/O	16 bit I/O lines that can be individually configured as inputs or outputs and also individually set or reset when configured as outputs.
IOP14	30		
IOP13	31		
IOP12	32		
IOP11	37		
IOP10	38		
IOP9	41		
IOP8	42		
IOP7	44		
IOP6	45		
IOP5	51		
IOP4	52		
IOP3	53		
IOP2	54		
IOP1	55		
IOP0 LSB	56		
			COMPARE AND CAPTURE SIGNALS
CMP0	8	O	Compare outputs. The states of these pins are determined by the combination of compare and action registers.
CMP1	7		
CMP2	2		
CMP3	1		
CAP0	68	I	Capture inputs. A transition on these pins causes the timer register to be captured in FIFO stack.
CAP1	67		
CMP4/CAP2/ FSR	66	I/O	This pin can be configured as compare output, capture input, or as external framing input/output for the receiver section of the serial port in codec mode.
CMP5/CAP3 FSX	65	I/O	This pin can be configured as compare output, capture input, or as external framing input/output for transmit section of the serial port in codec mode.

functional block diagram



LEGEND:

ACC—Accumulator	DP—Data Page Pointer
ACT—Action Register	IOP—Input/Output Port (Bit Selectable)
ALU—Arithmetic Logic Unit	PC—Program Counter
ARP—Auxiliary Register Point	P—P Register
AR0—Auxiliary Register 0	RBR—Receive Buffer Register
AR1—Auxiliary Register 1	RSR—Receive Shift Register
BSR—Bank Select Register	T—T Register
CAP—Capture	TBR—Transmit Buffer Register
CMPR—Compare Register	TSR—Transmit Shift Register

architecture

The TMS320 family utilizes a modified Harvard architecture for speed and flexibility. In a strict Harvard architecture, program and data memory lie in two separate spaces, permitting a full overlap of instruction fetch and execution. The TMS320 family's modification of the Harvard architecture allows transfers between program and data spaces, thereby increasing the flexibility of the device. This modification permits coefficients stored in program memory to be read into the RAM, eliminating the need for a separate coefficient ROM. It also makes available immediate instructions and subroutines based on computed values.

32-bit ALU/accumulator

The TMS320C14/E14 devices contain a 32-bit ALU and accumulator for support of double-precision, two's-complement arithmetic. The ALU is a general-purpose arithmetic unit that operates on 16-bit words taken from the data RAM or derived from immediate instructions. In addition to the usual arithmetic instructions, the ALU can perform Boolean operations, providing the bit manipulation ability required of a high-speed controller. The accumulator stores the output from the ALU and is often an input to the ALU. It operates with a 32-bit wordlength. The accumulator is divided into a high-order word (bits 31 through 16) and a low-order word (bits 15 through 0). Instructions are provided for storing the high- and low-order accumulator words in memory.

shifters

Two shifters are available for manipulating data. The ALU barrel shifter performs a left-shift of 0 to 16 places on data memory words loaded into the ALU. This shifter extends the high-order bit of the data word and zero-fills the low-order bits for two's-complement arithmetic. The accumulator parallel shifter performs a left-shift of 0, 1, or 4 places on the entire accumulator and places the resulting high-order accumulator bits into data RAM. Both shifters are useful for scaling and bit extraction.

16 x 16-bit parallel multiplier

The multiplier performs a 16 x 16-bit two's-complement multiplication with a 32-bit result in a single instruction cycle. The multiplier consists of three units: the T Register, P Register, and multiplier array. The 16-bit T Register temporarily stores the multiplicand; the P Register stores the 32-bit product. Multiplier values either come from the data memory or are derived immediately from the MPYK (multiply immediate) instruction word. The fast on-chip multiplier allows the device to perform fundamental operations such as convolution, correlation, and filtering.

data and program memory

Since the TMS320C14/E14 devices use a Harvard architecture, data and program memory reside in two separate spaces. These devices have 256 words of on-chip data RAM and 4K words of on-chip program ROM (TMS320C14) or EPROM (TMS320E14). The EPROM cell utilizes standard PROM programmers and is programed identically to a 64K CMOS EPROM (TMS27C64).

program memory expansion

The first-generation devices are capable of executing up to 4K words of external memory at full speed for those applications requiring external program memory space. This allows for external RAM-based systems to provide multiple functionality.

microcomputer/microprocessor operating modes

The TMS320C14/E14 devices offer two modes of operation defined by the state of the $\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ pin during reset: the microcomputer mode ($\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ = 1) or the microprocessor mode ($\overline{\text{NMI}}$/MC/$\overline{\text{MP}}$ = 0). In the microcomputer mode, on-chip ROM is mapped into the memory space with up to 4K words of internal memory available. In the microprocessor mode, all 4K words of memory are external.

interrupts and subroutines

The TMS320C14/E14 devices contain a four-level hardware stack for saving the contents of the program counter during interrupts and subroutine calls. Instructions are available for saving the complete context of the device. PUSH and POP instructions permit a level of nesting restricted only by the amount of available RAM. The TMS320C14/E14 has a total of 16 internal/external interrupts. Fifteen of these are maskable; $\overline{\text{NMI}}$ is the sixteenth.

input/output

The 16-bit parallel data bus can be utilized to access external peripherals. Only the lower three address lines are active, however. The upper nine address lines are driven high.

bit I/O

The TMS320C14/E14 has 16 pins of bit I/O that can be individually configured as inputs or outputs. Each of the pins can be set or cleared without affecting the others. The input pins can also detect and match patterns and generate a maskable interrupt signal to the CPU.

serial port

The TMS320C14/E14 includes an I/O mapped serial port that can operate in one of three modes: asynchronous, synchronous, and codec. Two types of inter-processor communication protocols are supported in all modes. An associated timer provides baud rate/clock generation if required. Depending on the mode, internal/external clock (master/slave) options are available. All communication parameters are software-controlled through a serial control register.

event manager

An event manager is included that provides up to four capture inputs and up to six compare outputs. This peripheral operates with the timers to provide a form of programmable event logging/detection. The six compare outputs can also be configured to produce six channels of high precision PWM.

timers 1 and 2

Two identical 16-bit timers are provided for general purpose applications. Both timers include a 16-bit period register and buffer latch, and can generate a maskable interrupt.

serial port timer

The serial port timer is a 16-bit timer primarily intended for baud rate generation for the serial port. Its architecture is the same as timers 1 and 2, therefore it can serve as a general purpose timer if not needed for serial communication.

watchdog timer

The TMS320C14/E14 contains a 16-bit watchdog timer that can produce a timeout ($\overline{\text{WDT}}$) signal for various applications such as software development and event monitoring. The watchdog timer also generates, at the point of the timeout, a maskable interrupt signal to the CPU.

instruction set

A comprehensive instruction set supports both numeric-intensive operations, such as signal processing, and general-purpose operations, such as high-speed control. All of the first-generation devices are object-code compatible and use the same 60 instructions. The instruction set consists primarily of single-cycle single-word instructions, permitting execution rates of more than six million instructions per second. Only infrequently used branch and I/O instructions are multicycle. Instructions that shift data as part of an arithmetic operation execute in a single cycle and are useful for scaling data in parallel with other operations.

NOTE

The $\overline{\text{BIO}}$ pin on other TMS320C1x devices are not available for use in the TMS320C14/E14. An attempt to execute the BIOZ (Branch on $\overline{\text{BIO}}$ low) instruction will result in a two cycle NOP action.

Three main addressing modes are available with the instruction set: direct, indirect, and immediate addressing.

direct addressing

In direct addressing, seven bits of the instruction word concatenated with the 1-bit data page pointer form the data memory address. This implements a paging scheme in which each page contains 128 words.

indirect addressing

Indirect addressing forms the data memory address from the least-significant eight bits of one of the two auxiliary registers, AR0 and AR1. The Auxiliary Register Pointer (ARP) selects the current auxiliary register. The auxiliary registers can be automatically incremented or decremented and the ARP changed in parallel with the execution of any indirect instruction to permit single-cycle manipulation of data tables. Indirect addressing can be used with all instructions requiring data operands, except for the immediate operand instructions.

immediate addressing

Immediate instructions derive data from part of the instruction word rather than from the data RAM. Some useful immediate instructions are multiply immediate (MPYK), load accumulator immediate (LACK), and load auxiliary register immediate (LARK).

instruction set summary

Table 1 lists the symbols and abbreviations used in Table 2, the instruction set summary. Table 2 contains a short description and the opcode for each TMS320 first-generation instruction. The summary is arranged according to function and alphabetized within each functional group.

TABLE 1. INSTRUCTION SYMBOLS

SYMBOL	MEANING
ACC	Accumulator
D	Data memory address field
I	Addressing mode bit
K	Immediate operand field
PA	3-bit port address field
R	1-bit operand field specifying auxiliary register
S	4-bit left-shift code
X	3-bit accumulator left-shift field

TABLE 2. TMS320 FIRST-GENERATION INSTRUCTION SET SUMMARY

ACCUMULATOR INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABS	Absolute value of accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0
ADD	Add to accumulator with shift	1	1	0	0	0	0	←	S		→	I	←			D			→
ADDH	Add to high-order accumulator bits	1	1	0	1	1	0	0	0	0	0	I	←			D			→
ADDS	Add to accumulator with no sign extension	1	1	0	1	1	0	0	0	0	1	I	←			D			→
AND	AND with accumulator	1	1	0	1	1	1	1	0	0	1	I	←			D			→
LAC	Load accumulator with shift	1	1	0	0	1	0	←	S		→	I	←			D			→
LACK	Load accumulator immediate	1	1	0	1	1	1	1	1	1	0	←				K			→
OR	OR with accumulator	1	1	0	1	1	1	1	0	1	0	I	←			D			→
SACH	Store high-order accumulator bits with shift	1	1	0	1	0	1	1	←	X	→	I	←			D			→
SACL	Store low-order accumulator bits	1	1	0	1	0	1	0	0	0	0	I	←			D			→
SUB	Subtract from accumulator with shift	1	1	0	0	0	1	←	S		→	I	←			D			→
SUBC	Conditional subtract (for divide)	1	1	0	1	1	0	0	1	0	0	I	←			D			→
SUBH	Subtract from high-order accumulator bits	1	1	0	1	1	0	0	0	1	0	I	←			D			→
SUBS	Subtract from accumulator with no sign extension	1	1	0	1	1	0	0	0	1	1	I	←			D			→
XOR	Exclusive OR with accumulator	1	1	0	1	1	1	1	0	0	0	I	←			D			→
ZAC	Zero accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	1
ZALH	Zero accumulator and load high-order bits	1	1	0	1	1	0	0	1	0	1	I	←			D			→
ZALS	Zero accumulator and load low-order bits with no sign extension	1	1	0	1	1	0	0	1	1	0	I	←			D			→

AUXILIARY REGISTER AND DATA PAGE POINTER INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LAR	Load auxiliary register	1	1	0	0	1	1	1	0	0	R	I	←			D			→
LARK	Load auxiliary register immediate	1	1	0	1	1	1	0	0	0	R	←				K			→
LARP	Load auxiliary register pointer immediate	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	K
LDP	Load data memory page pointer	1	1	0	1	1	0	1	1	1	1	I	←			D			→
LDPK	Load data memory page pointer immediate	1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	K
MAR	Modify auxiliary register and pointer	1	1	0	1	1	0	1	0	0	0	I	←			D			→
SAR	Store auxiliary register	1	1	0	0	1	1	0	0	0	R	I	←			D			→

TABLE 2. TMS320 FIRST-GENERATION INSTRUCTION SET SUMMARY (continued)

BRANCH INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
B	Branch unconditionally	2	2	1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0
				0 0 0 0 ← BRANCH ADDRESS →
BANZ	Branch on auxiliary register not zero	2	2	1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0
				0 0 0 0 ← BRANCH ADDRESS →
BGEZ	Branch if accumulator ≥ 0	2	2	1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0
				0 0 0 0 ← BRANCH ADDRESS →
BGZ	Branch if accumulator > 0	2	2	1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
				0 0 0 0 ← BRANCH ADDRESS →
BLEZ	Branch if accumulator ≤ 0	2	2	1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0
				0 0 0 0 ← BRANCH ADDRESS →
BLZ	Branch if accumulator < 0	2	2	1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0
				0 0 0 0 ← BRANCH ADDRESS →
BNZ	Branch if accumulator ≠ 0	2	2	1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
				0 0 0 0 ← BRANCH ADDRESS →
BV	Branch on overflow	2	2	1 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0
				0 0 0 0 ← BRANCH ADDRESS →
BZ	Branch if accumulator 0	2	2	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
				0 0 0 0 ← BRANCH ADDRESS →
CALA	Call subroutine from accumulator	2	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0
CALL	Call subroutine immediately	2	2	1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
				0 0 0 0 ← BRANCH ADDRESS →
RET	Return from subroutine or interrupt routine	2	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1

T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
APAC	Add P register to accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1
LT	Load T register	1	1	0 1 1 0 1 0 1 0 I ← D →
LTA	LTA combines LT and APAC into one instruction	1	1	0 1 1 0 1 1 0 0 I ← D →
LTD	LTD combines LT, APAC, and DMOV into one instruction	1	1	0 1 1 0 1 0 1 1 I ← D →
MPY	Multiply with T register, store product in P register	1	1	0 1 1 0 1 1 0 1 I ← D →
MPYK	Multiply T register with immediate operand; store product in P register	1	1	1 0 0 ← K →
PAC	Load accumulator from P register	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0
SPAC	Subtract P register from accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 1 0 0 0 0

TABLE 2. TMS320 FIRST-GENERATION INSTRUCTION SET SUMMARY (concluded)

CONTROL INSTRUCTIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINT	Disable interrupt	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1
EINT	Enable interrupt	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0
LST	Load status register	1	1	0	1	1	1	1	0	1	1	I	◄—— D ——►						
NOP	No operation	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
POP	POP stack to accumulator	2	1	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1
PUSH	PUSH stack from accumulator	2	1	0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0
ROVM	Reset overflow mode	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	0
SOVM	Set overflow mode	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1
SST	Store status register	1	1	0	1	1	1	1	1	0	0	I	◄—— D ——►						

I/O AND DATA MEMORY OPERATIONS

MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMOV	Copy contents of data memory location into next higher location	1	1	0	1	1	0	1	0	0	1	I	◄—— D ——►						
IN	Input data from port	2	1	0	1	0	0	0	◄PA►			I	◄—— D ——►						
OUT	Output data to port	2	1	0	1	0	0	1	◄PA►			I	◄—— D ——►						
TBLR	Table read from program memory to data RAM	3	1	0	1	1	0	0	1	1	1	I	◄—— D ——►						
TBLW	Table write from data RAM to program memory	3	1	0	1	1	1	1	1	0	1	I	◄—— D ——►						

development support products

Texas Instruments offers an extensive line of development support products to assist the user in all aspects of TMS320 first-generation-based design and development. These products range from development and application software to complete hardware development and evaluation systems such as the XDS/22. Table 3 lists the software and hardware support products for the first-generation TMS320 devices.

TABLE 3. TMS320C14 SOFTWARE AND HARDWARE SUPPORT

SOFTWARE TOOLS	PART NUMBER
Macro Assembler/Linker	
VAX VMS†	TMDS3242250-08
IBM PC MS-DOS‡§	TMDS3242850-02
VAX ULTRIX†	TMDS3242260-08
SUN-3 UNIX#¶	TMDS3242550-08
CPU Simulator	
VAX VMS†	TMDS3240211-08
IBM PC MS-DOS‡§	TMDS3240811-02
Digital Filter Design Package (DFDP)	
IBM PC MS-DOS‡§	DFDP-IBM002
DSP Software Library	
VAX VMS†	TMDC3240212-18
IBM PC MS-DOS‡§	TMDC3240812-12
HARDWARE TOOLS	**PART NUMBER**
Analog Interface Board (AIB2)	RTC/EVM320C-06
AIB2 Adapter Board	RTC/ADPC14A-06
XDS/22 Emulator	TMDS3262214
EPROM Programmer Adapter Socket	TMDX3270110
TMS320 Design Kit	TMS320DDK

†VAX, VMS, and ULTRIX are trademarks of Digital Equipment Corporation.
‡MS-DOS is a trademark of Microsoft, Incorporated.
§IBM PC is a trademark of IBM Corporation.
¶UNIX is a trademark of AT&T Bell Laboratories.
#SUN is a trademark of Sun Microsystems, Incorporated.

System development begins with the use of the Emulator (XDS). This hardware tool allows the designer to evaluate the processor's performance, benchmark time-critical code, and determine the feasibility of using a TMS320 device to implement a specific algorithm.

Software and hardware can be developed in parallel by using the macro assembler/linker and simulator for software development and the XDS for hardware development. The assembler/linker translates the system's assembly source program into an object module that can be executed by the CPU simulator or XDS. The XDS provides realtime in-circuit emulation and is a powerful tool for debugging and integrating software and hardware modules.

Additional support for the TMS320 products consists of extensive documentation and three-day DSP design workshops offered by the TI Regional Technology Centers (RTCs). The workshops provide hands-on experience with the TMS320 development tools. Refer to the *TMS320 Family Development Support Reference Guide* (SPRU011) for further information about TMS320 development support products and DSP workshops. When technical questions arise regarding the TMS320, contact the Texas Instruments TMS320 DSP Hotline, (713) 274-2320.

documentation support

Extensive documentation supports the first-generation TMS320 devices from product announcement through applications development. The types of documentation include data sheets with design specifications, complete user's guides, and 750 pages of application reports published in the book *Digital Signal Processing Applications with the TMS320 Family* (SPRA012A).

A series of DSP textbooks is being published by both Prentice Hall and John Wiley and Sons to support digital signal processing research and education. Prentice Hall (201) 767-5937 offers among others: *Practical Approaches to Speech Coding*, and *A DSP Laboratory Using the TMS32010*. John Wiley and Sons (800) 526-5368 has published such books as *Digital Filter Design, DFT/FFT and Convolution Algorithms,* and *A Practical Guide to Adaptive Filter Design.* The TMS320 newsletter, *Details on Signal Processing,* is published quarterly and distributed to update TMS320 customers on product information. The TMS320 DSP bulletin board service provides access to large amounts of information pertaining to the TMS320 family.

Refer to the *TMS320 Family Development Support Reference Guide* for further information about TMS320 documentation. To receive copies of first-generation TMS320 literature, call the Customer Response Center at 1-800-232-3200.

electrical specifications

This section contains all the electrical specifications for the TMS320C14/E14 devices, including test parameter measurement information. Parameters with pp subscript apply only to TMS320E14 in EPROM programming mode.

absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{CC}‡	−0.3 V to 7 V
Supply voltage range, V_{PP}‡	−0.6 V to 14 V
Input voltage range	−0.3 V to 14 V
Output voltage range	−0.3 V to 7 V
Continuous power dissipation	0.5 W
Air temperature range above operating device: L version	0 °C to 70 °C
Storage temperature range	−55 °C to +150 °C

†Stresses beyond 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 beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
‡All voltage values are with respect to V_{SS}.

recommended operating conditions

		MIN	NOM	MAX	UNIT
V_{CC} Supply voltage	EPROM devices	4.75	5	5.25	V
	EPROM devices while Fast programming	5.75	6.0	6.25	
	EPROM devices while SNAP! programming	6.25	6.5	6.75	
	All other devices	4.5	5	5.5	
V_{PP} Supply voltage for Fast programming (see Note 1)		12.25	12.5	12.75	V
V_{PP} Supply voltage for SNAP! programming (see Note 1)		12.75	13.0	13.25	V
V_{SS} Supply voltage			0		V
V_{IH} High-level input voltage	CLKIN	3			V
	CLKIN, CAP0, CAP1, CMP4/CAP2/FSR, CMP5/CAP3/FSX, $\overline{RS}$		4		
	All remaining inputs	2			
V_{IL} Low-level input voltage, all inputs except as noted				0.8	V
V_{IL} CAP0, CAP1, CMP4/CAP2/FSR, CMP5/CAP3/FSX, $\overline{RS}$			1		V
I_{OH} High-level output current, all outputs				−300	μA
I_{OL} Low-level output current, all outputs				2	mA
T_A Operating free-air temperature, L version		0		70	°C

NOTE 1: V_{PP} can be connected directly (except in the program mode). V_{CC} supply current in this case would be $I_{CC} + I_{PP}$.

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP[†]	MAX	UNIT
V_{OH}	High-level output voltage		I_{OH} = MAX		2.4	3		V
			I_{OH} = 20 μA (see Note 2)		$V_{CC}-0.4$[‡]			V
V_{OL}	Low-level output voltage		I_{OL} = MAX			0.3	0.5	V
I_{OZ}	Off-state output current		V_{CC} = MAX	V_O = 2.4 V			20	μA
				V_O = 0.4 V			−20	
I_I	Input current		V_I = V_{SS} to V_{CC}	All inputs except CLKIN			±20	μA
				CLKIN			±50	
I_{CC}[§]	Supply current	EPROM	f = 25.6 MHz, V_{CC} = 5.25 V, T_A = 0°C to 70°C			65		mA
		ROM	f = 25.6 MHz, V_{CC} = 5.25 V, T_A = 0°C to 70°C			55		mA
I_{PP1}	V_{PP} supply current		V_{PP} = V_{CC} = 5.5 V				100	μA
I_{PP2}	V_{PP} supply current (during program pulse)		V_{PP} = 13 V			30	50	mA
C_i	Input capacitance	Data bus	f = 1 MHz, All other pins 0 V			25[‡]		pF
		All others				15[‡]		
C_o	Output capacitance	Data bus				25[‡]		pF
		All others				10[‡]		

[†]All typical values are at V_{CC} = 5 V, T_A = 25°C, except I_{CC} at 70°C.
[‡]Values derived from characterization data and not tested.
[§]I_{CC} characteristics are inversely proportional to temperature.
NOTE 2: This voltage specification is included for interface to HC logic. However, note that all of the other timing parameters defined in this data sheet are specified for TTL logic levels and will differ for HC logic levels.

PARAMETER MEASUREMENT INFORMATION

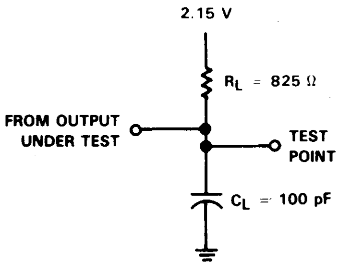


FIGURE 1. TEST LOAD CIRCUIT

EXTERNAL CLOCK REQUIREMENTS

The TMS320C14/E14 uses an external frequency source for a clock. This source is applied to the CLKIN pin, and must conform to the specifications in the table below.

PARAMETER	TEST CONDITIONS	MIN	NOM	MAX	UNIT
CLKIN input clock frequency	T_A = 0°C to 70°C	6.7		25.6	MHz

CLOCK TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLKOUT cycle time‡	$R_L = 825\,\Omega$, $C_L = 100$ pF, See Figure 1.	156.25	160	597	ns
$t_{r(C)}$	CLKOUT rise time			10†		ns
$t_{f(C)}$	CLKOUT fall time			8†		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			72†		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			70†		ns
$t_{d(MCC)}$	Delay time CLKIN↑ to CLKOUT↓			45†		ns

†Values derived from characterization data and not tested.
‡$t_{c(C)}$ is the cycle time of CLKOUT, i.e., 4 × $t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(MC)}$	Master clock cycle time	39.06	40	150	ns
$t_{r(MC)}$	Rise time master clock input		5†	10†	ns
$t_{f(MC)}$	Fall time master clock input		5†	10†	ns
$t_{w(MCP)}$	Pulse duration master clock	$0.45t_{c(MC)}$†		$0.55t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration master clock low		15†		ns
$t_{w(MCH)}$	Pulse duration master clock high		15†		ns

†Values derived from characterization data and not tested.

MEMORY READ AND INSTRUCTION TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d1}	Delay time CLKOUT↓ to address bus valid	$R_L = 825\,\Omega$, $C_L = 100$ pF, See Figure 1.	10†		40	ns
t_{d2}	Delay time CLKOUT↓ to $\overline{\text{REN}}$↓ (memory access)		$0.25t_{c(C)} - 5$†		$0.25t_{c(C)} + 12$	ns
t_{d3}	Delay time CLKOUT↓ to $\overline{\text{REN}}$↑ (memory access)		−10†		12	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{\text{REN}}$↓ (I/O access)		$0.25t_{c(C)} - 5$†		$0.25t_{c(C)} + 12$	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{\text{REN}}$↑ (I/O access)		−10†		12	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{\text{WE}}$↓		$0.5t_{c(C)} - 5$†		$0.5t_{c(C)} + 12$	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{\text{WE}}$↑		−10†		12	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid				$0.25t_{c(C)} + 52$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$0.25t_{c(C)} - 5$†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				$0.25t_{c(C)} + 30$†	ns
t_v	Data bus OUT valid after CLKOUT↓		$0.25t_{c(C)} - 10$			ns
$t_{h(A\text{-}WR)}$	Address hold time after $\overline{\text{WE}}$↑, $\overline{\text{REN}}$↑		0†			ns
$t_{su(A\text{-}REN)}$	Address bus setup time prior to $\overline{\text{REN}}$↓		$0.25t_{c(C)} - 35$			ns

†Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{su(D)}$	Setup time data bus valid prior to CLKOUT↓	$R_L = 825\,\Omega$, $C_L = 100$ pF, See Figure 1.	40			ns
$t_{h(D)}$	Hold time data bus held valid after CLKOUT↓ (see Note 3)		0			ns

NOTE 3: Data may be removed from the data bus upon $\overline{\text{REN}}$↑ preceding CLKOUT↓.

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time $\overline{WE}$↑, and $\overline{REN}$↑ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF, See Figure 1.			$0.5t_{c(C)}+50$†	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$0.25t_{c(C)}+50$†	ns
$t_{dis(A)}$	Address bus disable time after $\overline{RS}$ low				$0.25t_{c(C)}+50$†	ns
$t_{en(A)}$	Address bus enable time after $\overline{RS}$ high				$0.25t_{c(C)}+50$†	ns

†These values were derived from characterization data and not tested.

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 4)	40			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			ns

NOTE 4: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

MICROCOMPUTER/MICROPROCESSOR MODE ($\overline{NMI}$/MC/$\overline{MP}$)

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{h(MC/MP)}$‡	Hold time after RS high	$1.25t_c$			ns

‡Hold time to put device in microprocessor mode.

INTERRUPT ($\overline{INT}$)/NON-MASKABLE INTERRUPT ($\overline{NMI}$)

timing requirements over recommended operating conditions (see Note 5)

		MIN	NOM	MAX	UNIT
$t_{f(INT)}$	Fall time $\overline{INT}$			15†	ns
$t_{f(NMI)}$	Fall time $\overline{NMI}$			15†	ns
$t_{w(INT)}$	Pulse duration $\overline{INT}$	$t_{c(C)}$			ns
$t_{w(NMI)}$	Pulse duration $\overline{NMI}$	$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time $\overline{INT}$ before CLKOUT low	40			ns
$t_{su(NMI)}$	Setup time $\overline{NMI}$ before CLKOUT low	40			ns

†These values were derived from characterization data and not tested.
NOTE 5: $\overline{INT}$ and $\overline{NMI}$ are synchronous inputs and can occur at any time during the cycle. $\overline{NMI}$ and $\overline{INT}$ are edge triggered only.

ADVANCE INFORMATION

BIT I/O TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{rfP}	Rise and fall time outputs	R_L = 825 Ω, C_L = 100 pF, See Figure 1.			20†	ns
$t_{d(IOP)}$	CLKOUT low to data valid outputs				$.25t_{c(C)}+20$	ns

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{rfI(IOP)}$	Rise and fall time inputs	R_L = 825 Ω,			20†	ns
$t_{su(IOP)}$	Data setup time before CLKOUT time	C_L = 100 pF,	20†			ns
$t_{wI(IOP)}$	Input pulse duration	See Figure 1.	$t_{c(C)}+20$			ns

†These values were derived from characterization data and not tested.

GENERAL PURPOSE TIMERS

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{r(TIM)}$	TCLK1, TCLK2 rise time	R_L = 825 Ω, C_L = 100 pF, See Figure 1.			20†	ns
$t_{f(TIM)}$	TCLK1, TCLK2 fall time				20†	ns
$t_{hL(TIM)}$	Hold time TCLK1, TCLK2 low		$t_{c(C)}+20$			ns
$t_{hH(TIM)}$	Hold time TCLK1, TCLK2 high		$t_{c(C)}+20$			ns

†These values were derived from characterization data and not tested.

WATCHDOG TIMER TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{f(WDT)}$	Fall time, $\overline{\text{WDT}}$	R_L = 825 Ω,			20†	ns
$t_{d(WDT)}$	CLKOUT to $\overline{\text{WDT}}$ valid	C_L = 100 pF,	$0.25t_{c(C)}+20$			ns
$t_{w(WDT)}$	$\overline{\text{WDT}}$ output pulse duration	See Figure 1.	$8t_{c(C)}-20$		$8t_c+20$	ns

†These values were derived from characterization data and not tested.

EVENT MANAGER TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{f(CMP)}$	Fall time, CMP0-CMP5	R_L = 825 Ω, C_L = 100 pF, See Figure 1.			20†	ns
$t_{r(CMP)}$	Rise time, CMP0-CMP5				20†	ns

†These values were derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{w(CAP)}$	CAP0-CAP3 input pulse duration	R_L = 825 Ω, C_L = 100 pF, See Figure 1.	$t_{c(C)}+20$			ns
$t_{su(CAP)}$	Capture input setup time before CLKOUT low		20†			ns

†These values were derived from characterization data and not tested.

ADVANCE INFORMATION

SERIAL PORT-SYNCHRONOUS MODE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
$t_{c(CLK-S)}$	Serial port clock cycle time†	$R_L = 825\,\Omega$, $C_L = 100$ pF, See Figure 1.	$t_{c(C)}$	$65{,}536t_{c(C)}$	ns
$t_{f(CLK-S)}$	TXD/CLK fall time†			20	ns
$t_{r(CLK-S)}$	TXD/CLK rise time†			20	ns
$t_{wL(CLK-S)}$	TXD/CLK low time†		$0.5t_{c(CLK-S)} - 20$	$0.5t_{c(CLK-S)} + 20$	ns
$t_{wH(CLK-S)}$	TXD/CLK high time†		$0.5t_{c(CLK-S)} - 20$	$0.5t_{c(CLK-S)} + 20$	ns
$t_{d(TX-S)}$	RXD/DATA output valid before TXD/CLK low†		$t_{wH(CLK-S)} - 20$		ns
$t_{h(TX-S)}$	RXD/DATA hold after TXD/CLK (internal) low†			$t_{d(TX-S)} + 20$	ns
$t_{d(TX-S)}$	RSD/DATA output valid before TXD/CLK low‡		$t_{wH} - 1.75t_{c(C)} + 20$		ns
$t_{h(TX-S)}$	RXD/DATA hold after after TXD/CLK low‡			$t_{wL} + 1.75t_{c(C)} + 20$	ns

†Internal clock
‡External clock

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	MAX	UNIT
$t_{wL(CLK-S)}$	TXD/CLK low time (external)‡	$R_L = 825\,\Omega$, $C_L = 100$ pF, See Figure 1.	$2t_{c(C)}$		ns
$t_{wH(CLK-S)}$	TXD/CLK high time (external)‡		$2t_{c(C)}$		ns
$t_{su(RX-S)}$	RXD/DATA input setup before TXD/CLK low‡		0		ns
$t_{h(RX-S)}$	RXD/DATA input hold after TXD/CLK low‡			$2t_{c(C)} - 20$	ns
$t_{su(RX-S)}$	RXD/DATA input setup before TXD/CLK external low†		20		ns
$t_{h(RX-S)}$	RXD/DATA input hold after TXD/CLK external low†		$0.25t_{c(C)} + 20$		ns

†Internal clock
‡External clock

SERIAL PORT-CODEC MODE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
$t_{d(TXD-C)}$	TXD output valid before CLKX low	$R_L = 825\,\Omega$, $C_L = 100$ pF, See Figure 1.	$0.5t_{c(C)} - 20$		ns
$t_{h(TXD-C)}$	TXD output hold after CLKX low			$t_{wL} + 1.75t_{c(C)} - 20$	ns

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	MAX	UNIT
$t_{c(CLK-C)}$	CLKR, CLKX cycle time	$R_L = 825\,\Omega$, $C_L = 100$ pF, See Figure 1.	$3t_{c(C)}$§		ns
$t_{f(CLK-C)}$	CLKR, CLKX fall time			20§	ns
$t_{r(CLK-C)}$	CLKR/CLKX rise time			20§	ns
$t_{wL(CLK-C)}$	CLKR, CLKX high time		$1.5t_{c(C)} - 20$¶		ns
$t_{wH(CLK-C)}$	CLKR, CLKX low time		$1.5t_{c(C)} - 20$¶		ns
$t_{su(FSX)}$	FSX valid before CLKX low		$0.5t_{c(C)} - 20$		ns
$t_{su(FSR)}$	FSR valid before CLKR low		$0.5t_{c(C)} - 20$		ns
$t_{su(RXD-C)}$	RXD input setup time before CLKR low		0		ns
$t_{h(TXD-C)}$	RXD input hold time after CLKR low			$t_{c(C)} + 20$	ns

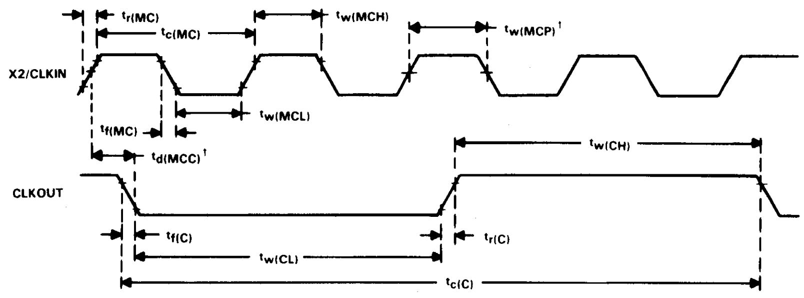
§These values were derived from characterization data and not tested.
¶This cycle time is only possible when CLK(R) and CLK(X) are synchronized with CLKOUT.

timing diagrams

This section contains all the timing diagrams for the TMS320C14/E14 devices.

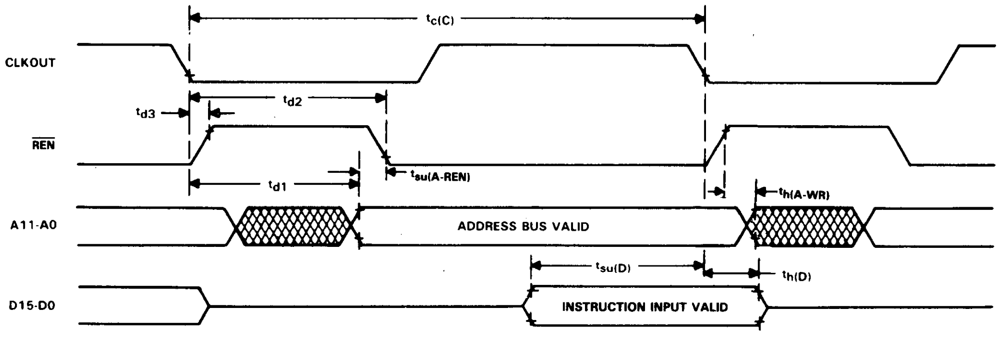
Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

clock timing



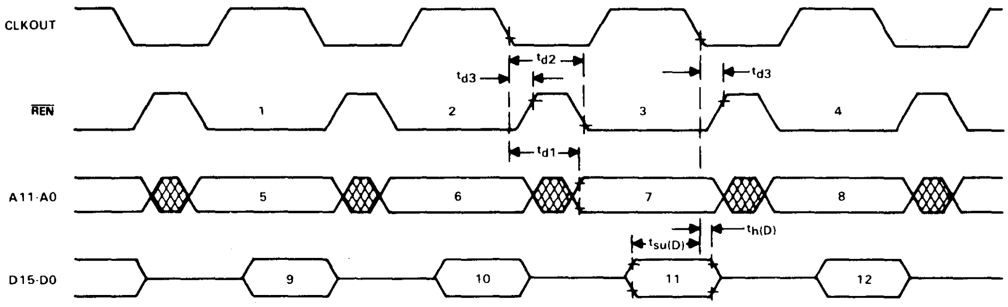
†$t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5 volts on the CLKIN waveform.

memory read timing



ADVANCE INFORMATION

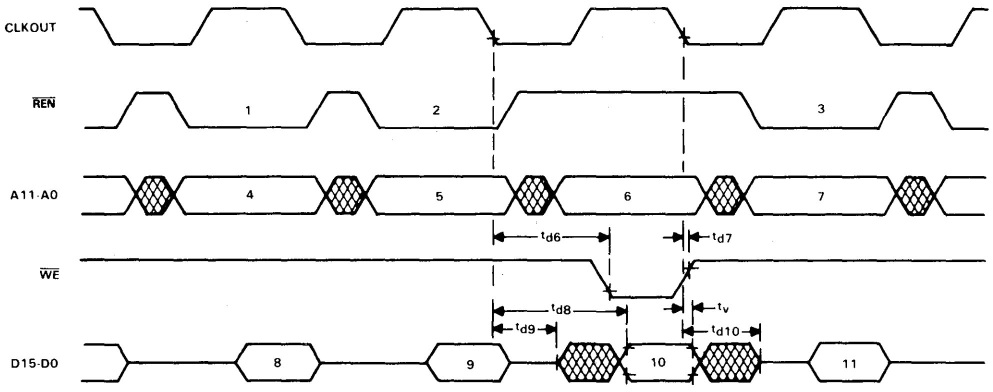
TBLR instruction timing



LEGEND:

1. TBLR INSTRUCTION PREFETCH	7. ADDRESS BUS VALID
2. DUMMY PREFETCH	8. ADDRESS BUS VALID
3. DATA FETCH	9. INSTRUCTION INPUT VALID
4. NEXT INSTRUCTION PREFETCH	10. INSTRUCTION INPUT VALID
5. ADDRESS BUS VALID	11. DATA INPUT VALID
6. ADDRESS BUS VALID	12. INSTRUCTION INPUT VALID

TBLW instruction timing

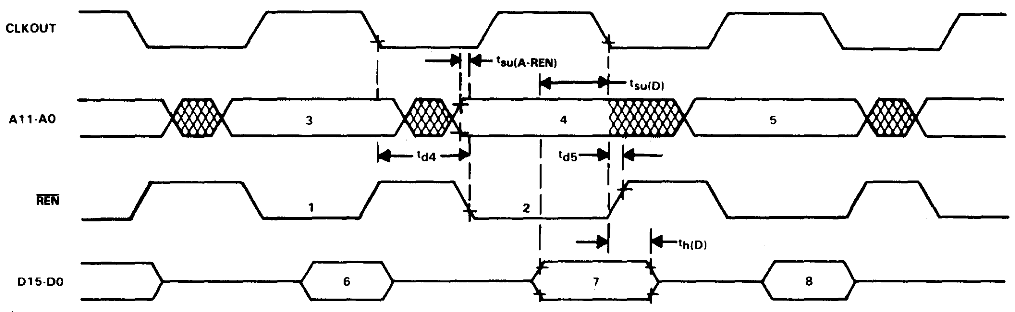


LEGEND:

1. TBLW INSTRUCTION PREFETCH	7. ADDRESS BUS VALID
2. DUMMY PREFETCH	8. INSTRUCTION INPUT VALID
3. NEXT INSTRUCTION PREFETCH	9. INSTRUCTION INPUT VALID
4. ADDRESS BUS VALID	10. DATA OUTPUT VALID
5. ADDRESS BUS VALID	11. INSTRUCTION INPUT VALID
6. ADDRESS BUS VALID	

ADVANCE INFORMATION

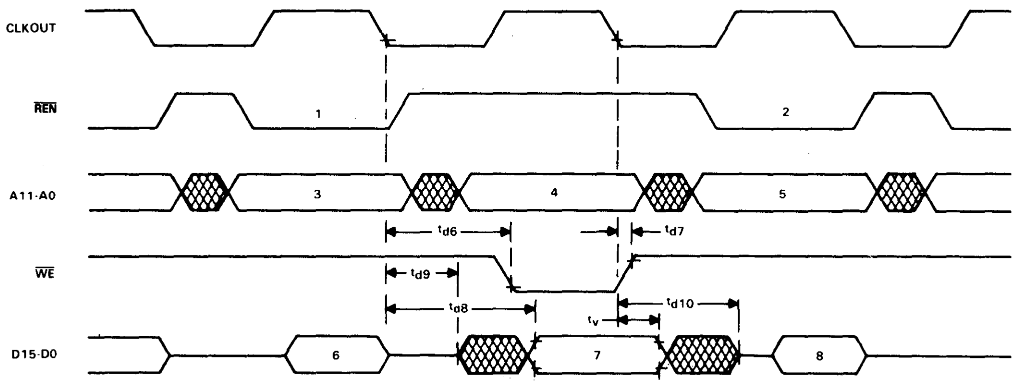
IN instruction timing



LEGEND:

1. IN INSTRUCTION PREFETCH	5. ADDRESS BUS VALID
2. NEXT INSTRUCTION PREFETCH	6. INSTRUCTION INPUT VALID
3. ADDRESS BUS VALID	7. DATA INPUT VALID
4. PERIPHERAL ADDRESS VALID	8. INSTRUCTION INPUT VALID

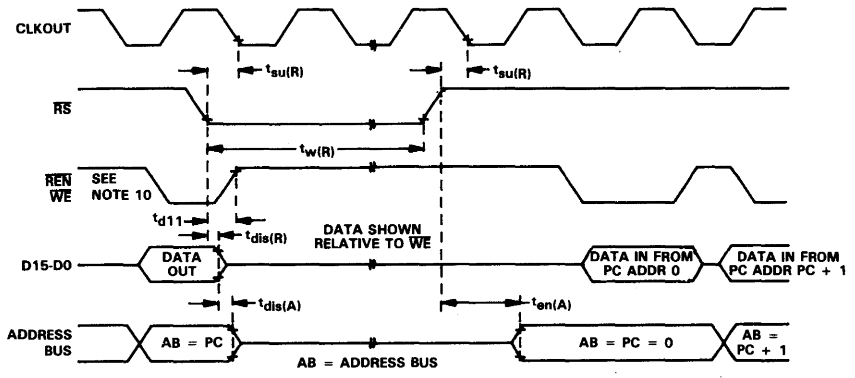
OUT instruction timing



LEGEND:

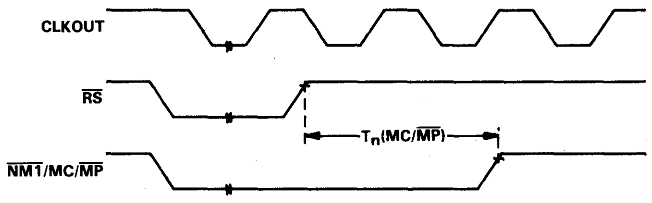
1. OUT INSTRUCTION PREFETCH	5. ADDRESS BUS VALID
2. NEXT INSTRUCTION PREFETCH	6. INSTRUCTION INPUT VALID
3. ADDRESS BUS VALID	7. DATA OUTPUT VALID
4. PERIPHERAL ADDRESS VALID	8. INSTRUCTION INPUT VALID

reset timing



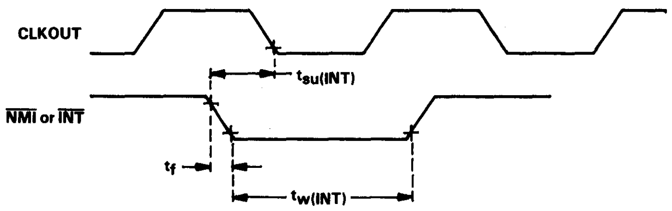
NOTES:
6. $\overline{RS}$ forces $\overline{REN}$ and $\overline{WE}$ high and places data bus D0-D15 and address bus A0-A11 in a high-impedance state. AB outputs (and program counter) are synchronously cleared to zero after the next complete CLK cycle from $\overline{RS}$↑.
7. $\overline{RS}$ must be maintained for a minimum of five clock cycles.
8. Resumption of normal program will commence after one complete CLK cycle from $\overline{RS}$↑.
9. Due to the synchronizing action on $\overline{RS}$, time to execute the function can vary dependent upon when $\overline{RS}$↑ or $\overline{RS}$↓ occur in the CLK cycle.
10. Diagram shown is for definition purpose only. $\overline{WE}$ and $\overline{REN}$ are mutually exclusive.

microcomputer/microprocessor mode timing diagram

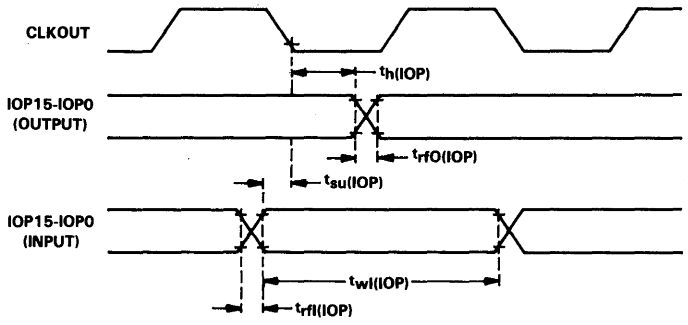


ADVANCE INFORMATION

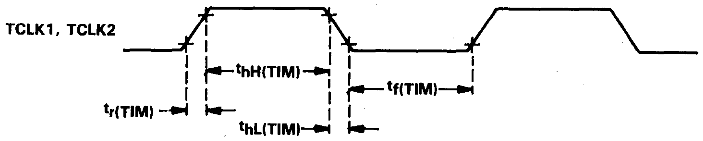
interrupt timing



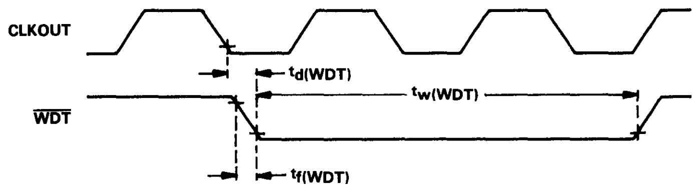
bit I/O timing



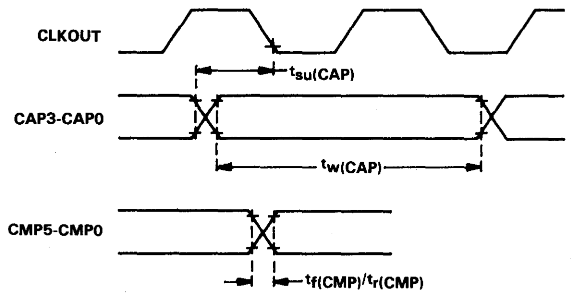
general purpose timers



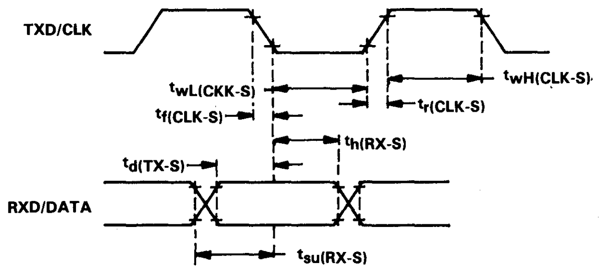
watchdog timer



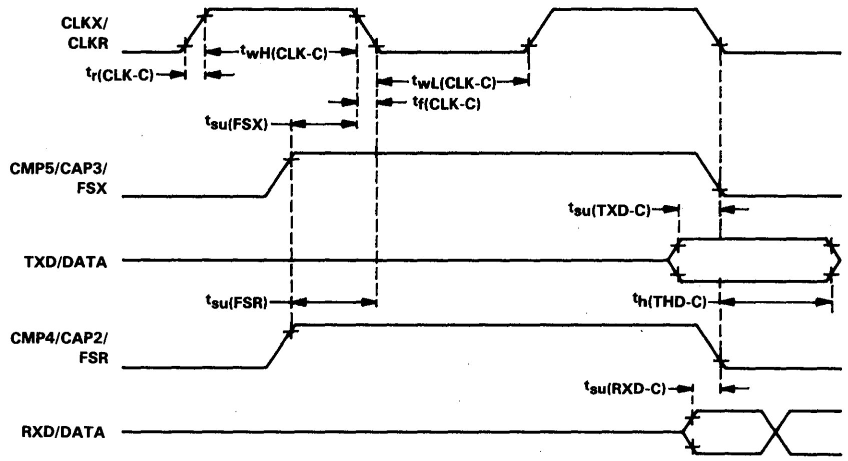
event manager



serial port - synchronous mode timing



serial port - codec mode timing



ADVANCE INFORMATION

EPROM programming

The TMS320E14 includes a 4K x 16-bit industry-standard EPROM cell for prototyping and low-volume production. The TMS320C14 with a 4K-word masked ROM then provides a migration path for cost-effective production. An EPROM adapter socket (part #TMDX3270110), shown in Figure 2, is available to provide 68-pin to 28-pin conversion for programming the TMS320E14.

Key features of the EPROM cell include the normal programming operation as well as verification. The EPROM cell also includes a code protection feature that allows code to be protected against copyright violations.

The TMS320E14 EPROM cell is programmed using the same family and device codes as the TMS27C64 8K × 8-bit EPROM. The TMS27C64 EPROM series are ultraviolet-light erasable, electrically programmable, read-only memories, fabricated using HVCMOS technology. They are pin-compatible with existing 28-pin ROMs and EPROMs. These EPROMs operate from a single 5-V supply in the read mode; however, a 12.5-V supply is needed for programming. All programming signals are TTL level. For programming outside the system, existing EPROM programmers can be used. Locations may be programmed singly, in blocks, or at random.

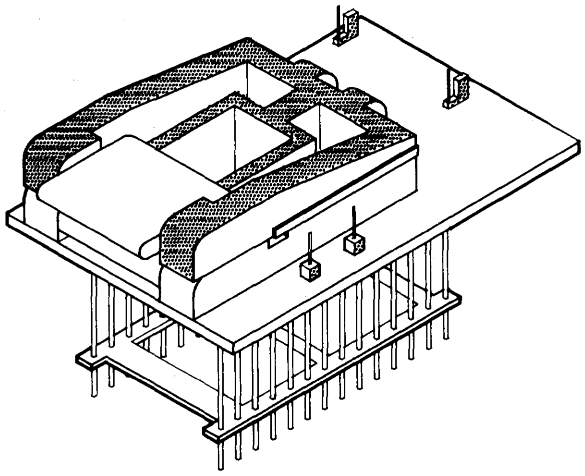


FIGURE 2. EPROM ADAPTER SOCKET

The TMS320E14 uses 12 address lines plus $\overline{\mathrm{WE}}$ to address the 4K-word memory in byte format (8K-byte memory). In word format, the most-significant byte of each word is assigned an even address and the least-significant byte an odd address in the byte format. Programming information should be downloaded to EPROM programmer memory in a high-byte to low-byte order for proper programming of the devices (see Figure 3.)

TMS320C14 On-Chip Program Memory (Word Format)		**TMS320E14 On-Chip Program Memory (Byte Format)**		**EPROM Programmer Memory Byte Format with Adapter Socket**	
0(0000h)	1234h	0(0000h)	34h	0(0000h)	12h
1(0001h)	5678h	1(0001h)	12h	1(0001h)	34h
2(0002h)	9ABCh	2(0002h)	78h	2(0002h)	56h
3(0003h)	DEF0h	3(0003h)	56h	3(0003h)	78h
.	.	4(0004h)	BCh	4(0004h)	9Ah
.	.	5(0005h)	9Ah	5(0005h)	BCh
.	.	6(0006h)	F0h	6(0006h)	DEh
4095(0FFh)		7(0007h)	DEh	7(0007h)	F0h
		.	.	.	.
		.	.	.	.
		.	.	.	.
				8191(1FFFh)	

FIGURE 3. EPROM PROGRAMMING DATA FORMAT

Figure 4 shows the wiring conversion to program the TMS320E14 using the 28-pin pinout of the TMS27C64. The table of pin nomenclature provides a description of the TMS27C64 pins.

CAUTION

The TMS320E14 does not support the signature mode available with some EPROM programmers. The signature mode puts a high voltage (12.5 V DC) on pin A9. The TMS320E14 EPROM cell is not designed for this feature and will be damaged if subjected to it. A 3.9 kΩ resistor is standard on the TI programmer socket between pin A9 and the programmer. This protects the device from unintentional use of the signature mode.

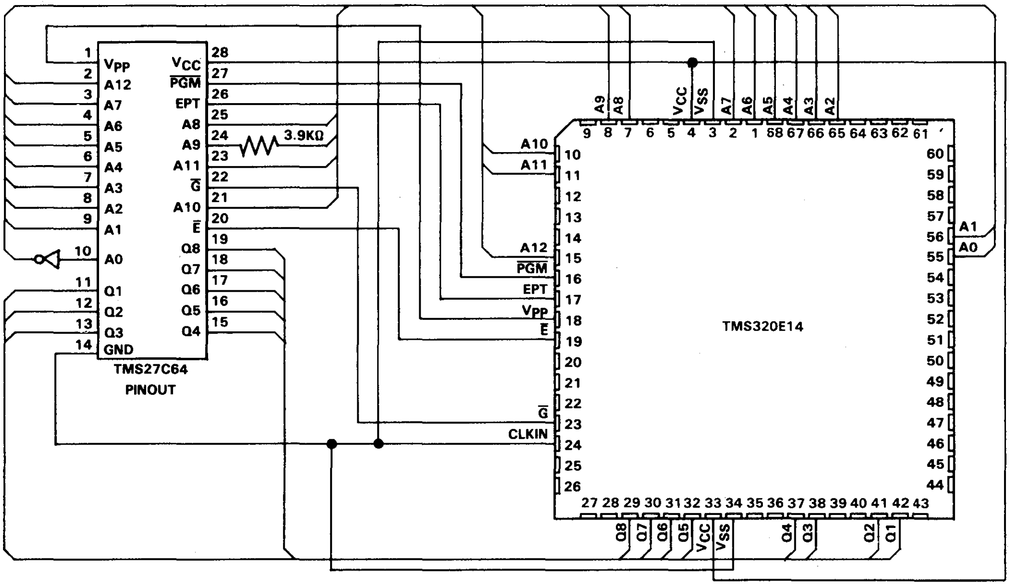


FIGURE 4. TMS320E14 EPROM PROGRAMMING CONVERSION TO TMS27C64 EPROM PINOUT

PIN NOMENCLATURE (TMS320E14)

NAME	I/O	DEFINITION
A12(MSB)-A0(LSB)	I	On-chip EPROM programming address lines
CLKIN	I	Clock oscillator input
$\overline{E}$	I	EPROM chip enable
EPT	I	EPROM test mode select
$\overline{G}$	I	EPROM output enable
GND	I	Ground
$\overline{PGM}$	I	EPROM write/program select
Q8(MSB)-Q1(LSB)	I/O	Data lines for byte-wide programming of on-chip 8K bytes of EPROM
$\overline{RS}$	I	Reset for initializing the device
V_{CC}	I	5-V to 6.5-V power supply
V_{PP}	I	12.5-V to 13-V power supply

ADVANCE INFORMATION

Table 4 shows the programming levels required for programming, verifying, reading, and protecting the EPROM cell.

TABLE 4. TMS320E14 PROGRAMMING MODE LEVELS

SIGNAL NAME[†]	TMS320E14 PIN	TMS27C64 PIN	PROGRAM	PROGRAM VERIFY	READ	EPROM PROTECT	PROTECT VERIFY
$\overline{E}$	19	20	V_{IL}	V_{IL}	V_{IL}	V_{IH}	V_{IL}
$\overline{G}$	23	22	V_{IH}	$\overline{PULSE}$	$\overline{PULSE}$	V_{IH}	V_{IL}
$\overline{PGM}$	16	27	$\overline{PULSE}$	V_{IH}	V_{IH}	V_{IH}	V_{IH}
V_{PP}	18	1	V_{PP}	V_{PP}	V_{CC}	V_{PP}	V_{CCP}
V_{CC}	4,33	28	V_{CCP}	V_{CCP}	V_{CC}	V_{CCP}	V_{CCP}
V_{SS}	3,34	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
CLKIN	24	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
EPT	17	26	V_{SS}	V_{SS}	V_{SS}	V_{PP}	V_{PP}
Q1-Q8	42,41,38,37, 32-29	19-15,13-11	D_{IN}	Q_{OUT}	Q_{OUT}	$Q_8 = \overline{PULSE}$	$Q_8 =$ RBIT
A12-A7	15,11,10, 8,7,2	25,24,23, 21,3,2	ADDR	ADDR	ADDR	X	X
A6	1	4	ADDR	ADDR	ADDR	X	V_{IL}
A5	68	5	ADDR	ADDR	ADDR	X	X
A4	67	6	ADDR	ADDR	ADDR	V_{IH}	X
A3-A0	66,65,56,55	7-10	ADDR	ADDR	ADDR	X	X

[†]Signal names shown for TMS320E14 EPROM programming mode only.

LEGEND:
V_{IH} = TTL high level; V_{IL} = TTL low level; ADDR = byte address bit; V_{PP} = 12.5 V ± 0.25 V (FAST) or 13.0 V ± 0.25 V (SNAP!).
V_{CC} = 5 V ± 0.25 V; X = don't care; $\overline{PULSE}$ = low-going TTL pulse.
D_{IN} = byte to be programmed at ADDR; Q_{OUT} = byte stored at ADDR.
V_{CCP} = 6.0 V ± 0.25 V (FAST) or 6.5 V ± 0.25 V (SNAP!)

programming

Since every memory bit in the cell is a logic 1, the programming operation reprograms certain bits to 0. Once programmed, these bits can only be erased using ultraviolet light. The correct byte is placed on the data bus with Vpp set to the 12.5-V level. The $\overline{PGM}$ pin is then pulsed low to program in the zeroes.

erasure

Before programming, the device must be erased by exposing it to ultraviolet light. The recommended minimum exposure dose (UV-intensity X exposure-time) is 15 watt-seconds per square centimeter. A typical 12 milliwatt-seconds per square centimeter, filterless UV lamp will erase the device in 21 minutes. The lamp should be located about 2.5 centimeters above the chip during erasure. After exposure, all bits are in the high state.

verify/read

To verify correct programming, the EPROM cell can be read using either the verify or read line definitions shown in Table 4, assuming the inhibit bit has not been programmed.

program inhibit

Programming may be inhibited by maintaining a high level input on the $\overline{E}$ pin or $\overline{PGM}$ pin.

standard programming procedure

Before programming, the device must first be completely erased. Then the device can be programmed with the correct code. It is advisable to program unused sections with zeroes as a further security measure. After the programming is complete, the code programmed into the cell should be verified. If the cell passes verification, the next step is to program the ROM protect bit (RBIT). Once the RBIT programming is verified, an opaque label should be placed over the window to protect the EPROM cell from inadvertent erasure by ambient light. At this point, the programming is complete, and the device is ready to be placed into its destination circuit.

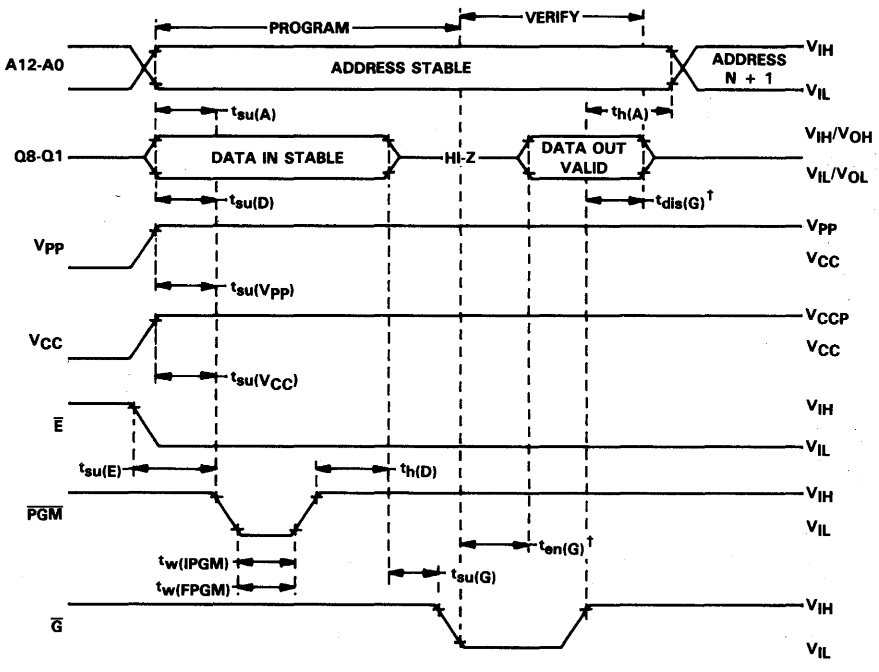
Refer to Appendix F of the TMS320C14/E14 User's Guide for additional information on EPROM programming.

recommended timing requirements for programming: V_{CC} = 6 V and V_{PP} = 12.5 V (Fast) or V_{CC} = 6.5 V and V_{PP} = 13.0 V (SNAP! Pulse), T_A = 25 °C (see Note 6)

			MIN	NOM	MAX	UNIT
$t_{w(IPGM)}$	Initial program pulse duration	Fast programming algorithm	0.95	1	1.05	ms
		SNAP! Pulse programming algorithm	95	100	105	µs
$t_{w(FPGM)}$	Final pulse duration	Fast programming only	2.85		78.75	ms
$t_{su(A)}$	Address setup time		2			µs
$t_{su(E)}$	$\overline{E}$ setup time		2			µs
$t_{su(G)}$	$\overline{G}$ setup time		2			µs
$t_{su(D)}$	Data setup time		2			µs
$t_{su(VPP)}$	V_{PP} setup time		2			µs
$t_{su(VCC)}$	V_{CC} setup time		2			µs
$t_{h(A)}$	Address hold time		0			µs
$t_{h(D)}$	Data hold time		2			µs

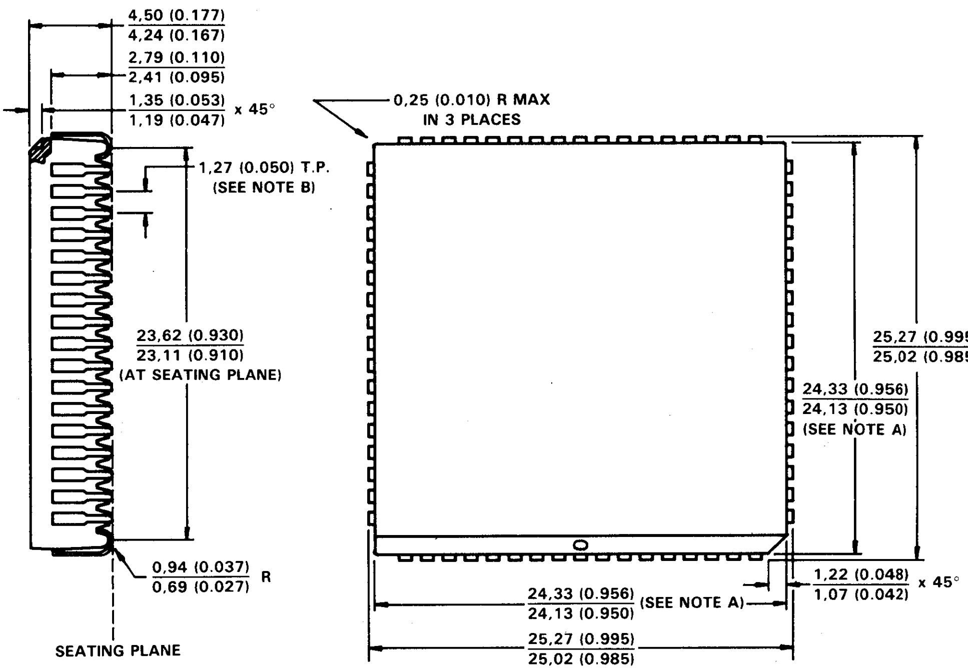
NOTE: 6. For all switching characteristics and timing measurements, input pulse levels are 0.40 V to 2.4 V and V_{PP} = 12.5 V ± 0.5 V during programming.

program cycle timing



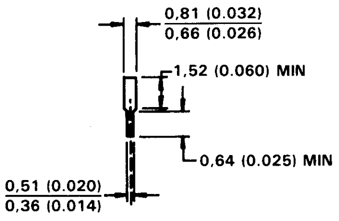
†$t_{dis(G)}$ and $t_{en(G($ are characteristics of the device but must be accommodated by the programmer.

68-lead plastic chip carrier package (FN suffix)



THERMAL RESISTANCE CHARACTERISTICS (SEE NOTE C)

$R_{\theta JA}$ °C/W	$R_{\theta JC}$ °C/W
60	17



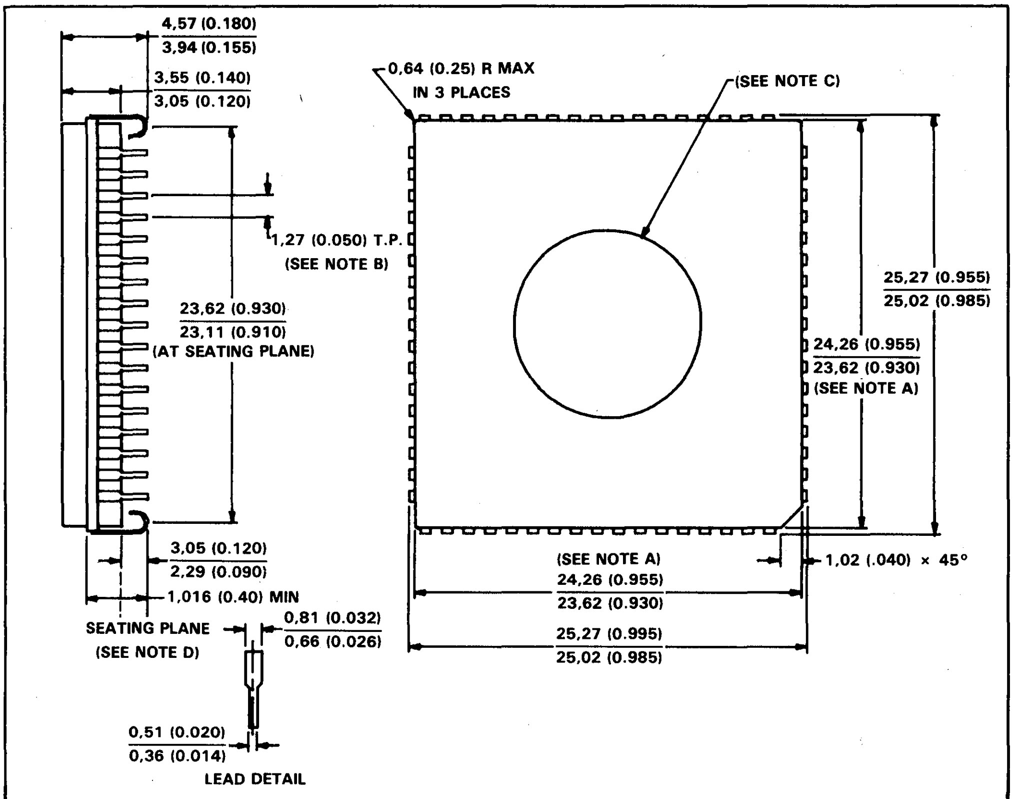
LEAD DETAIL

NOTES: A. Centerline of center pin each side is within 0,10 (0.004) of package centerline as determined by this dimension.
B. Location of each pin is within 0,27 (0.005) of true position with respect to center pin on each side.
C. Thermal resistance calculations based on I_{CC} = 65 mA TYP at T_A = 70°C.

ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES.

ADVANCE INFORMATION

68-lead cerquad chip carrier package (FZ suffix)



ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES.

NOTES: A. Centerline of center pin each side is within 0,10 (0.004) of package centerline as determined by this dimension.
B. Location of each pin is within 0,27 (0.005) of true position with respect to center pin on each side.
C. Glass is optional.
D. The lead contact points are planar within 0,15 (0.006).

INDEX

ADVANCE INFORMATION

Appendix B

ROM Codes

Board space can be a critical concern in many DSP applications. In order to reduce chip count and provide the customer with a single-chip solution, the TMS320C14/E14 provides substantial on-chip memory and peripherals. The on-chip ROM of the TMS320C14 can be masked with the customer's own code, while the EPROM of the TMS320E14 is reprogrammable. This allows the customer to take advantage of the general-purpose features of TI's digital signal processors while at the same time customizing the processor to suit a specific application.

To facilitate design, prototype work may be performed using the TMS320E14 device. TMS320C14/E14 development tools permit a designer to test and refine algorithms for immediate results. When the algorithm has been finalized, the customer can submit the code to Texas Instruments to be masked into the on-chip ROM of the device. The code medium maybe the device itself or a floppy disk.

The MC/$\overline{\text{MP}}$ (microcomputer/microprocessor) mode often shortens design and field upgrade cycle times, thereby reducing expense. This mode permits the customer to use the TMS320C14/E14 as a standard device operating out of external program memory. When TMS320C14/E14 code is altered during design, the delays associated with new silicon processing are avoided. Field upgrade cycle times and the associated expense of inventory obsolescence when the code is altered are also avoided.

An entire algorithm or an often-used routine may be programmed into the on-chip program memory space of the TMS320C14/E14 device. The micro-processer mode adds flexibility by using external program memory. This allows multiple function operation by a single device, enhancing the product's capa-bilities.

Both the TMS320C14 and the TMS320E14 devices provide 4K words of on-chip program memory space, and can address the same amount of external memory in the microprocessor mode.

Note:

The last 96 words of internal program memory on the TMS320C14 ROM are reserved for TI use only, and should not be used by the customer. If the program code requires a full 4K x 16 bit memory, external memory or EPROM version (TMS320E14) must be used.

Figure B-1 illustrates the procedure flow for implementing TMS320C14 masked parts. For the current mask-ROM production charges and minimum order requirements, contact the nearest TI Field Sales Office.

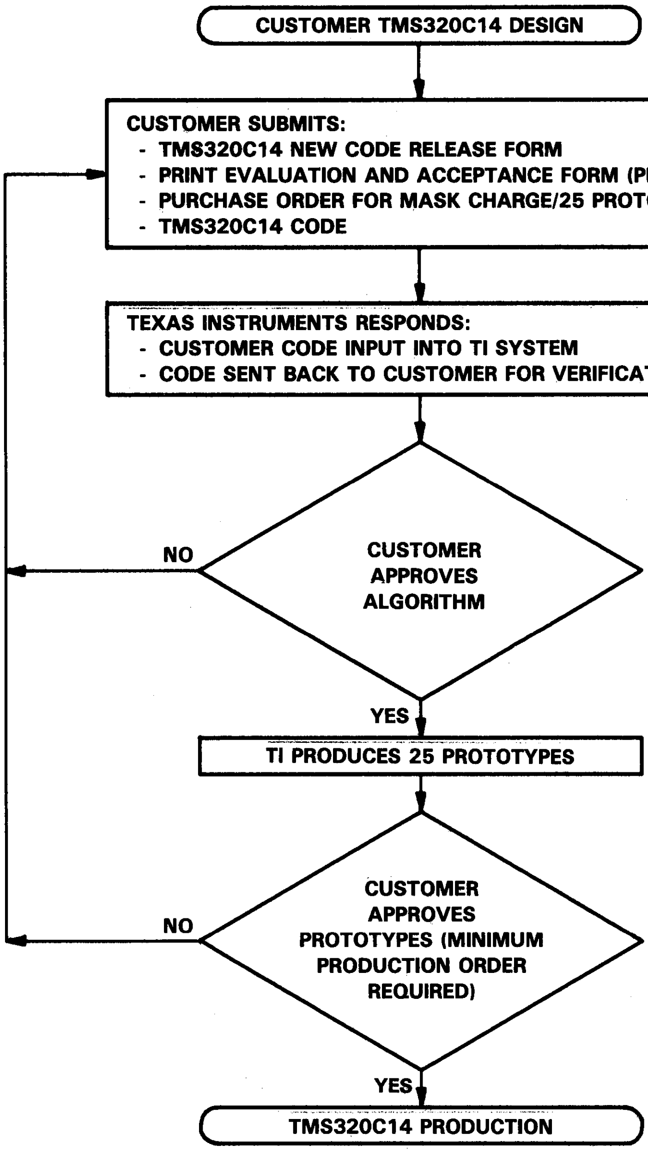

CUSTOMER TMS320C14 DESIGN
CUSTOMER SUBMITS:
- TMS320C14 NEW CODE RELEASE FORM
- PRINT EVALUATION AND ACCEPTANCE FORM (PEAF)
- PURCHASE ORDER FOR MASK CHARGE/25 PROTOTYPES
- TMS320C14 CODE
TEXAS INSTRUMENTS RESPONDS:
- CUSTOMER CODE INPUT INTO TI SYSTEM
- CODE SENT BACK TO CUSTOMER FOR VERIFICATION
NO
CUSTOMER APPROVES ALGORITHM
YES
TI PRODUCES 25 PROTOTYPES
NO
CUSTOMER APPROVES PROTOTYPES (MINIMUM PRODUCTION ORDER REQUIRED)
YES
TMS320C14 PRODUCTION

Figure B-1. TMS320C14 ROM Code Flowchart

Leadtimes for the first 25 prototype units begin when the customer has formally verified that TI has recorded his code correctly. Leadtimes for the first production order begin once the customer formally approves the masked prototypes. The typical leadtime for masked TMS320C14 prototypes is 8 weeks and for masked TMS320C14 production 10 to 12 weeks. Texas Instruments constantly strives to improve these leadtimes and reserves the right to make changes at any time. Please contact the nearest TI Sales Office for current leadtimes, further information on these procedures, and confirmation of the mask/production requirements.

A TMS320C14 ROM code may be submitted in one of the following formats (the preferred media is 5 1/4" floppies):

PROM:	TBP28S166, TBP28S86
EPROM:	TMS27C64
FLOPPY:	TI Cross-Assembler Format

When a code is submitted to Texas Instruments for a masked device, the code is reformatted to accommodate the TI mask generation system. System level verification by the customer is therefore necessary. Although the code has been reformatted, it is important that the changes remain transparent to the user and not affect the execution of the algorithm. The formatting changes made involve deletion of all address tags (unnecessary in a ROM code device) and addition of data in the reserved locations of the ROM for device ROM test. Note that because these changes have been made, a checksum comparison is not a valid means of verification.

ROM code algorithms may also be submitted by secure electronic transfer via a modem. Contact the nearest TI sales office for further information.

With each masked device order, the customer must sign a disclaimer stating:

> "The units to be shipped against this order were assembled, for expediency purposes, on a prototype (i.e., non-production qualified) manufacturing line, the reliability of which is not fully characterized. Therefore, the anticipated inherent reliability of these prototype units cannot be expressly defined."

and a release stating:

> "Any masked ROM device may be resymbolized as TI standard product and resold as though it were an unprogrammed version of the device at the convenience of Texas Instruments."

ROM codes will be deleted from the TI system after one year from the last delivery.

Appendix C

Quality and Reliability

The quality and reliability performance of Texas Instruments Microprocessor and Microcontroller Products, which includes the three generations of TMS320 digital signal processors, relies on feedback from:

- Our customers
- Our total manufacturing operation from front-end wafer fabrication to final shipping inspection
- Product quality and reliability monitoring.

Our customer's perception of quality must be the governing criterion for judging performance. This concept is the basis for Texas Instruments Corporate Quality Policy, which is as follows:

> "For every product or service we offer, we shall define the requirements that solve the customer's problems, and we shall conform to those requirements without exception."

Texas Instruments offers a leadership reliability qualification system, based on years of experience with leading-edge memory technology as well as years of research into customer requirements. Quality and reliability programs at TI are therefore based on customer input and internal information to achieve constant improvement in quality and reliability.

C.1 Reliability Stress Tests

Accelerated stress tests are performed on new semiconductor products and process changes to ensure product reliability excellence. The typical test environments used to qualify new products or major changes in processing are:

- High-temperature operating life
- Storage life
- Temperature cycling
- Biased humidity
- Autoclave
- Electrostatic discharge
- Package integrity
- Electromigration
- Channel-hot electrons (performed on geometries less than 2.0 μm).

Typical events or changes that require internal requalification of product include:

- New die design, shrink, or layout
- Wafer process (baseline/control systems, flow, mask, chemicals, gases, dopants, passivation, or metal systems)
- Packaging assembly (baseline control systems or critical assembly equipment)
- Piece parts (such as lead frame, mold compound, mount material, bond wire, or lead finish)
- Manufacturing site.

TI reliability control systems extend beyond qualification. Total reliability controls and management include product ramp monitor as well as final product release controls. MOS memories, utilizing high-density active elements, serve as the leading indicator in wafer-process integrity at TI MOS fabrication sites, enhancing all MOS logic device yields and reliability. TI places more than 200,000 MOS devices per month on reliability test to ensure and sustain built-in product excellence.

Table C-1 lists the microprocessor and microcontroller reliability tests, the duration of the test, and sample size. The following defines and describes those tests in the table.

AOQ (Average Outgoing Quality)	Amount of defective product in a population, usually expressed in terms of parts per million (PPM).
FIT (Failure In Time)	Estimated field failure rate in number of failures per billion power-on device hours; 1000 FITS equals 0.1 percent fail per 1000 device hours.
Operating lifetest	Device dynamically exercised at a high ambient temperature (usually 125°C) to simulate field usuage that would

	expose the device to a much lower ambient temperature (such as 55°C). Using a derived high temperature, a 55°C ambient failure rate can be calculated.
High-temperature storage	Device exposed to 150°C unbiased condition. Bond integrity is stressed in this environment.
Biased humidity	Moisture and bias used to accelerate corrosion-type failures in plastic packages. Conditions include 85°C ambient temperature with 85-percent relative humidity (RH). Typical bias voltage is +5 V and ground on alternating pins.
Autoclave (pressure cooker)	Plastic-packaged devices exposed to moisture at 121°C using a pressure of one atmosphere above normal pressure. The pressure forces moisture permeation of the package and accelerates corrosion mechanisms (if present) on the device. External package contaminates can also be activated and caused to generate inter-pin current leakage paths.
Temperature cycle	Device exposed to severe temperature extremes in an alternating fashion (-65°C for 15 minutes and 150°C for 15 minutes per cycle) for at least 1000 cycles. Package strength, bond quality, and consistency of assembly process are stressed in this environment.
Thermal shock	Test similar to the temperature cycle test, but involving a liquid-to-liquid transfer, per MIL-STD-883C, Method 1011.
PIND	Particle Impact Noise Detection test. A non-destructive test to detect loose particles inside a device cavity.
Mechanical Sequence:	
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Mechanical shock	Per MIL-STD-883C, Method 2002.3, 1500 g, 0.5 ms, Condition B
PIND (optional)	Per MIL-STD-883C, Method 2020.4
Vibration, variable frequency	Per MIL-STD-883C, Method 2007.1, 20 g, Condition A
Constant acceleration	Per MIL-STD-883C, Method 2001.2, 20 kg, Condition D, Y1 Plane min
Fine and gross leak	Per MIL-STD-883C, Method 1014.5

Electrical test	To data sheet limits
Thermal Sequence:	
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Solder heat (optional)	Per MIL-STD-750C, Method 1014.5
Temperature cycle (10 cycles minimum)	Per MIL-STD-883C, Method 1010.5, -65 to +150°C, Condition C
Thermal shock (10 cycles minimum)	Per MIL-STD-883C, Method 1011.4, -55 to +125°C, Condition B
Moisture resistance	Per MIL-STD-883C, Method 1004.4
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Electrical test	To data sheet limits
Thermal/Mechanical Sequence:	
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Temperature cycle (10 cycles minimum)	Per MIL-STD-883C, Method 1010.5, -65 to +150°C, Condition C
Constant acceleration	Per MIL-STD-883C, Method 2001.2, 30 kg, Y1 Plane
Fine and gross leak	Per MIL-STD-883C, Method 1014.5
Electrical test	To data sheet limits
Electrostatic discharge	Per MIL-STD-883C, Method 3015
Solderability	Per MIL-STD-883C, Method 2003.3
Solder heat	Per MIL-STD-750C, Method 2031, 10 sec
Salt atmosphere	Per MIL-STD-883C, Method 1009.4, Condition A, 24 hrs min
Lead pull	Per MIL-STD-883C, Method 2004.4, Condition A
Lead integrity	Per MIL-STD-883C, Method 2004.4, Condition B1
Electromigration	Accelerated stress testing of conductor patterns to ensure acceptable lifetime of power-on operation
Resistance to solvents	Per MIL-STD-883C, Method 2015.4

Table C-1. Microprocessor and Microcontroller Tests

TEST	DURATION	SAMPLE SIZE PLASTIC	SAMPLE SIZE CERAMIC
Operating life, 125°C, 5.0 V	1000 hrs	129	129
Operating life, 150°C, 5.0 V	1000 hrs	77 *	77
Storage life, 150°C	1000 hrs	77	77
Biased 85°C/85 percent RH, 5.0 V	1000 hrs	129	-
Autoclave, 121°C, 1 ATM	240 hrs	77	-
Temperature cycle, -65 to 150°C	1000 cyc†	129	129
Temperature cycle, 0 to 125°C	3000 cyc	129	129
Thermal shock, -65 to 150°C	200 cyc	129	129
Electrostatic discharge, ±2 kV		12	12
Latch-up (CMOS devices only)		5	5
Mechanical sequence		-	38
Thermal sequence		-	38
Thermal/mechanical sequence		-	38
PIND		-	45
Internal water vapor		-	3
Solderability		22	22
Solder heat		22	22
Resistance to solvents		15	15
Lead integrity		15	15
Lead pull		22	-
Lead finish adhesion		15	15
Salt atmosphere		15	15
Flammability (UL94-V0)		3	-
Thermal impedance		5	5

* If junction temperature does not exceed plasticity of package.
† For severe environments; reduced cycles for office environments.

Table C-2 provides a list of the TMS320C14/E14 devices, the approximate number of transistors, and the equivalent gates. The numbers have been determined from design verification runs.

Table C-2. Transistor Complement

DEVICE	# TRANSISTORS	# GATES
TMS320C14	129K	33K
TMS320E14	135K	54K

TI Qualification test updates are available upon request at no charge. TI will consider performing any additional reliability test(s), if requested. For more information on TI quality and reliability programs, contact the nearest TI field sales office.

Note:

Texas Instruments reserves the right to make changes in MOS Semiconductor test limits, procedures, or processing without notice. Unless prior arrangements for notification have been made, TI advises all customers to reverify current test and manufacturing conditions prior to relying on published data.

Appendix D

Development Support/Part Order Information

This section provides development support information, device part numbers, and support tool ordering information for the TMS320C14/E14 products. Extensive documentation, including application reports, user's guides, and textbooks, is available to support DSP design, research, and education. To order TMS320 literature, contact the TI Customer Response Center (CRC), 1-800-232-3200. For more information about support products and documentation, refer to the *TMS320 Family Development Support Reference Guide* (SPRA012A).

The nearest TI field sales office can be contacted for support tool availability or further details (see list of sales offices and distributors at end of book). For technical support, contact the TMS320 DSP hotline, (713) 274-2320.

The major topics discussed in this section are listed below.

- Development Support (Section D.1 on page D-2)
  - TMS320C1x/TMS320C2x Assembly Language Tools
  - TMS320C1x CPU Simulator
  - TMS320C14 Emulator (XDS/22)
  - TMS320 Analog Interface Board 2(AIB2)
  - TMS320C14/E14 AIB2 Adapter
  - Digital Filter Design Package (DFDP)
  - DSP Software Library
  - TMS320 DSP Hotline/Bulletin Board Service

- Part Order Information (Section D.2 on page D-11)
  - Device part numbers
  - Software and hardware support tools part numbers
  - Device and support tool prefix designators
  - Device and support tool nomenclature

D.1 TMS320C14/E14 Development Support

Texas Instruments offers extensive development support and complete documentation with the TMS320C14/E14 digital signal processors. Tools are provided to evaluate the performance of the processors, develop algorithm implementations, and fully integrate the design's software and hardware modules. Development operations are performed with the TMS320C1x/TMS320C2x Assembly Language Tools, Simulator, Emulator (XDS),and other support products.

A description and key features for each TMS320C14/E14 development support tool is provided in the following subsections. For more information about support products, refer to the *TMS320 Family Development Support Reference Guide* (SPRA012A). For ordering information, see Section D.2.

D.1.1 TMS320C1x/TMS320C2x Assembly Language Tools

The TMS320C1x/TMS320C2x Assembly Language Tools provide program code generation for first and second generation TMS320 devices, including the TMS320C14/E14 devices. The assembly language tools package consists of the following:

- An **Assembler** that translates assembly language source files into machine language object code in a common object file format (COFF).
- An **Archiver** that allows the programmer to collect a group of files into a single file, or produce a "library" of macros.
- A **Linker** that combines object files into a single executable module.
- A **Format Conversion Utility** that converts files into TI-tagged, Intel, or Tektronix object format.

Figure D-1 shows the assembly language tools development flow. The shaded section in the center of the diagram represents the basic routine for software development. Other portions are optional.

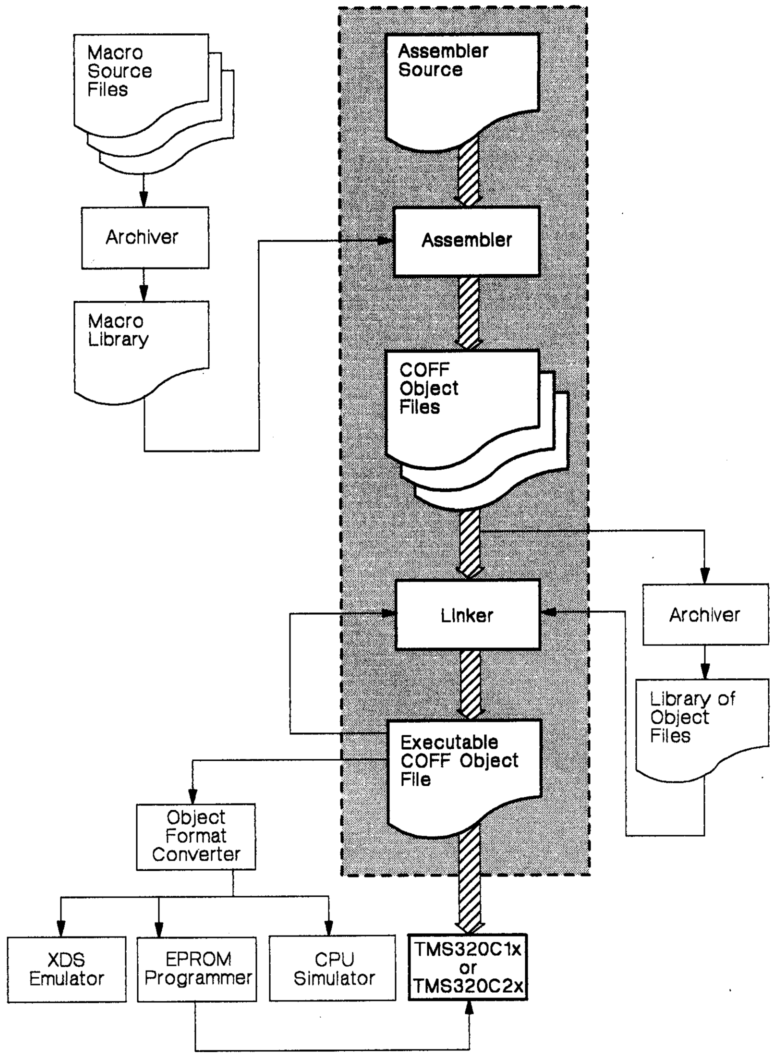


Figure D-1. TMS320C14/E14 Software Development Flow

The TMS320C1x/TMS320C2x Assembly Language Tools create and use object files that are in common object file format (COFF). This format is an improvement over object code developed with earlier macro assemblers. The COFF files provide more efficient programming of the TMS320C14/E14 by allowing the programmer to divide and sub- divide program code into sections for modular manipulation/relocation.

The advantages of COFF are:

- Faster execution of code,
- Easier programming,
- Supports high level languages *and*
- Allows symbolic debugging.

The assembly language tools are available for the following systems:

- **PCs:**
  - IBM PC with MS-DOS
- **VAX:**
  - VMS
  - ULTRIX
- **SUN**
  - SUN-3 UNIX

Note:

The COFF files generated by the assembly language tools are not compatible with TI-tagged, Intel, or Tektronix object files. The code conversion utility *does* convert COFF files into a standard format for use with most EPROM programmers.

D.1.2 TMS320C1x CPU Simulator

The TMS320C1x CPU Simulator is a software program that simulates operation of the TMS320C1x CPU to allow program verification. The debug mode enables the user to monitor the state of the simulated TMS320C1x while the program is executing. The simulator uses the object code produced by the TMS320C1x Assembly Language Tools. During program execution, the internal registers and memory of the simulated device are modified as each instruction is interpreted by the host computer. Once program execution is suspended, the internal registers and both program and data memories can be inspected and/or modified. In addition, files can be associated with the I/O ports.

Note:

The TMS320C1x CPU Simulator only simulates the operation of the CPU. The operation of peripherals unique to a particular TMS320 family device (such as the TMS320C14/E14) are **not** simulated.

The following features highlight simulator capability for effective TMS320C1x software development:

- Program debug/verification
- Single-step option
- Trace/breakpoint capabilities
- Full access to simulated registers and memories

The simulator is currently available for the VAX/VMS and IBM PC MS-DOS operating systems.

D.1.3 TMS320 Third-Party Support

The TMS320 family of digital signal processors is supported by product and service offerings from many independent vendors and consultants, known as third parties. These support products take many forms (both software and hardware) from cross assemblers, simulators, and DSP utility packages to logic analyzers and emulators. The expertise of those involved in support services range from speech encoding and vector quantization to software/hardware design and system analysis.

Section 11 of the *TMS320 Family Development Support Reference Guide* (SPRU011) describes a number of tools and services that augment the TMS320 support provided by Texas Instruments. An update of this third-party support is given in the *TMS320 Development Support Update* (SPRZ003). Inclusion of a product in either of these publications does not constitue product endorsement on the part of Texas Instruments, but merely an attempt at product awareness.

D.1.4 TMS320C14/E14 Emulator (XDS)

The TMS320C14/E14 Emulator (XDS/22) is a user-friendly system that has all the features necessary for realtime in-circuit emulation of TMS320C1x family devices. Both hardware and software can be integrated in the debugging process. By setting breakpoints based on internal conditions or external events, execution of the program can be suspended and control given to the debug mode. In the debug mode, all registers and memory locations can be inspected and modified. Single-step execution is available. Full-trace capabilities at full speed and a reverse assembler that translates machine code back into assembly instructions also increase debugging productivity. Using a standard RS-232-C port, the object file produced by the TMS320C1x/C2x Assembly Language Tools can be downloaded into the emulator, which then can be controlled through a terminal.

The XDS/22 provides 4K x 16 words of high-speed static RAM (zero wait states) for program memory. It also has the capability of executing out of target memory to utilize the full TMS320C14/E14 program/data address range. For multiprocessing configurations, up to nine emulators can be daisy-chained together.

The XDS/22 emulator is a completely self-contained system with power supply. With three RS-232-C ports, the XDS/22 Emulator can be interfaced to a terminal, host computer for source or object downloading/uploading capabilities, and printer or PROM programmer.

The key features of the TMS320C14/E14 XDS/22 Emulator are as follows:

- Full-speed in-circuit emulation
- 4K words of program memory for user code
- Hardware breakpoint on program, data, or I/O conditions
- 2K words of full-speed hardware trace
- Use of target system crystal or internal crystal
- Up to ten software breakpoints
- Single-step option
- Assembler/reverse assembler
- Host-independent upload/download capabilities to/from program or data memory
- Ability to inspect and modify registers and program/data memory
- Multiprocessor system development.

Figure D-2 shows a block diagram of a typical system configuration using the TMS320C14/E14 XDS/22 Emulator.

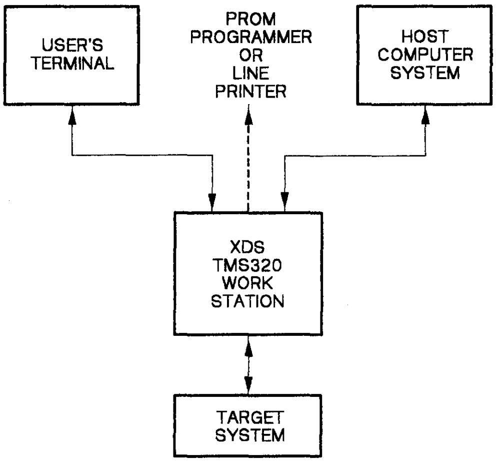


Figure D-2. TMS320C14/E14 XDS/22 System Configuration

D.1.5 TMS320 Analog Interface Board (AIB2)

The TMS320 Analog Interface Board (AIB2) is an analog-to-digital and digital-to-analog conversion board used as a preliminary target system with the TMS320C1x XDS, or another emulator (see Figure D-3). The AIB2 is an educational tool that provides a simple, inexpensive way to become familiar with digital signal processing (DSP) techniques.

The AIB2 allows testing of application programs with analog I/O by providing an interface to the TMS320C14 XDS/22 through an adapter board.

Key features of the AIB2 are as follows:

- Sockets for TMS320C10/C15/C17/C25 devices
- Socket for TMS32020 device
- 16-bit analog-to-digital converter with sample and hold
- 16-bit digital-to-analog converter
- Supports TLC3204x Analog Interface chips and TCM2918 codec chip
- Stand-alone operation (dual 27xxx EPROM sockets and socketed oscillator
- On-board noise and function generator

The sample rate clock on the AIB is derived from the onboard oscillator and may be programmed to provide periodic analog input, output, or both. There are two analog lowpass filters on the AIB. One filter on the A/D input bandlimits the input to minimize aliasing effects. The other filter smooths the output of the D/A. The frequency response of the filters is controlled by varying the external components in the filter stages. The cutoff of these filters is set to 4.7 kHz, but may be (plug) programmed. A pre-amp with a microphone input and an audio amplifier that will drive an 8-ohm speaker are provided. Sockets for 8K words of expansion memory are also provided. This memory is addressed through I/O and can support direct or autoincrement/decrement addressing. Up to 64K words of memory may be addressed through the memory expansion connector via this I/O interface.

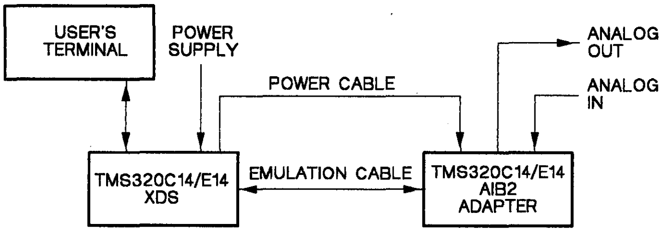


Figure D-3. TMS320C14/E14 AIB2 System Configuration

D.1.6 TMS320C14/E14 AIB2 Adapter

The TMS320C14/E14 Adapter allows the AIB2 board previously discussed to support the TMS320C14 XDS/22. The Adapter plugs into the TMS320C15/C17 socket to support the signals generated by the TMS320C14/E14. For instance, the adapter produces the $\overline{\text{MEN}}$ and $\overline{\text{DEN}}$ signals from the $\overline{\text{REN}}$ signal. The adapter includes a wire-wrap area with access to compare, capture, and bit I/O pins.

D.1.7 Digital Filter Design Package (DFDP)

The Digital Filter Design Package (DFDP) from Atlanta Signal Processors, Inc. (ASPI) is a user-friendly, menu-driven software package intended to speed the design of digital filters with floating-point accuracy or fixed-point economy in a variety of filter structures. The package consists of four interactive filter design modules capable of performing the following functions:

1) Designing FIR filters (Kaiser window)
2) Designing FIR filters (Parks-McClellan)
3) Designing IIR filters (Butterworth, Chebychev I and II, and elliptic)
4) Generating TMS320C1x assembly code by converting the ASCII file containing the filter coefficients into fully commented assembly language code for TMS320C1x devices.

Cascade and parallel structures as well as higher-performance lattice, normalized lattice, and orthogonal forms are included in the modules.

The DFDP can design filters to meet any piecewise linear response specification, evaluate filter characteristics before and after coefficient quantization, and design special-purpose FIR filters, such as multiband filters, differentiators, Hilbert transformers, and raised-cosine filters. The DFDP can also generate coefficients for filter implementations on any general-purpose processor or signal processing chip, as well as fully commented assembly language code for a variety of DSP chips. Magnitude, log magnitude, and impulse responses can be plotted for printer or screen display; in addition, the phase, group delay, and pole-zero map can be plotted for IIR filters. After the filter is designed, the user can generate code associated with the filter using the CGEN design module.

The DFDP runs on the IBM PS/2, IBM PC/XT/AT, and compatible systems. Operating systems must have 192 kbytes of memory available. For more information, contact Atlanta Signal Processors, Inc. (404-892-7265) or the nearest TI field sales office.

D.1.8 DSP Software Library

The Digital Signal Processing Software Library contains the major DSP routines (FFT, FIR/IIR filtering, and floating-point operations) and application algorithms (echo cancellation, ADPCM, and DTMF coding/decoding) presented in the book, *Digital Signal Processing Applications with the TMS320 Family* (SPRA012A). These routines and algorithms are written in either TMS320C1x and/or TMS320C2x source code. In addition, macros for the TMS320C1x are included in the library.

The software package consists of four diskettes for use with the IBM PC MS-DOS (version 1.1 or later) or a 1600 BPI magnetic tape for the VAX/VMS version. All the directories on the PC MS-DOS version are contained on the magnetic tape for the VMS version. Each directory contains a README.LIS file briefly describing the contents of the files in the directory and the reference to the code. The book, *Digital Signal Processing Applications with the TMS320 Family* (SPRA012A), is the major reference for the theory and algorithms, and also provides printed code in the appendices of each application report. The library can be ordered through TI (see Table D-2 for ordering information). The programs in the software library can also be downloaded from the TMS320 Bulletin Board Service (see D.1.10 for further information).

All the software in the library is copyrighted by Texas Instruments. The library is continually being updated; therefore, check the TMS320 DSP Bulletin Board (713-274-2323) for update information.

D.1.9 TMS320 DSP Hotline/Bulletin Board Service

The TMS320 group at Texas Instruments provides a DSP Hotline to answer TMS320 technical questions such as device problems, development tools, documentation, upgrades, and new TMS320 products. The hotline is open five days a week from 8:00 AM to 6:00 PM Central Time. The phone number is (713) 274-2320. To order literature, call the Customer Response Center (CRC) at (800) 232-3200. For pricing and availability of TMS320 devices or development tools, contact the nearest TI sales office.

The TMS320 DSP Bulletin Board Service is a telephone-line computer bulletin board that provides access to information pertaining to TMS320 devices. Specification updates for current or new TMS320 devices and development tools are communicated via the bulletin board as the information becomes available. The Bulletin Board Service can be accessed with a 300, 1200, or 2400-bps modem by dialing (713) 274-2323.

The bulletin board contains TMS320 source code from the application reports included in the book, *Digital Signal Processing Applications with the TMS320*

Family (SPRA012A) The bulletin board also provides new DSP applications software as it becomes available. See the *TMS320 Family Development Support Reference Guide* (SPRA012A) for information on how to access the bulletin board.

The TMS320 DSP hotline also has a FAX number which can be used for technical questions or other information. The hotline FAX number is (713) 274-2324.

D.2 Part Order Information

This section provides the device and support tool part numbers. Table D-1 lists the part numbers for all the first-generation members of the TMS320 family. Table D-2 gives ordering information for TMS320C14/E14 hardware and software support tools. A discussion of the TMS320 family device and development support tool prefix and suffix designators is included to assist in understanding the TMS320 product numbering system.

Table D-1. TMS320C14/E14 Digital Signal Processor Part Numbers

DEVICE NAME	TECHNOLOGY	OPERATING FREQUENCY	PACKAGE TYPE
TMS320C14FNL	CMOS	25.6 MHz	68-pin PLCC
TMS320E14FZL	CMOS	25.6 MHz	68-pin CLCC

Table D-2. TMS320C14/E14 Support Tool Part Numbers

TOOL DESCRIPTION	OPERATING SYSTEM	PART NUMBER
SOFTWARE		
Assembly Language Tools (Assembler/Linker)	VAX VMS IBM PC MS-DOS VAX ULTRIX SUN-3 UNIX	TMDS3242250-08 TMDS3242850-02 TMDS3242260-08 TMDS3242550-08
Simulator	VAX VMS IBM PC MS-DOS	TMDS3240211-08 TMDS3240811-02
Digital Filter Design Package	IBM PC MS-DOS	DFDP-/IBM002
DSP Software Library	VAX VMS IBM PC MS-DOS	TMDC3240212-18 TMDC3240812-12
HARDWARE		
XDS/22 Emulator		TMDX3262214
Analog Interface Board 2(AIB2)		RTC/EVM320C-06
TMS320C14 Adapter Board		RTC/ADPC14A-06
EPROM Programmer Adapter Socket		TMDX3270110

D.2.1 Device and Development Support Tool Prefix Designators

To assist the user in understanding the stages in the product development cycle, Texas Instruments assigns prefix designators in the part number nomenclature. A device prefix designator has three options: TMX, TMP, and TMS, and a development support tool prefix designator has two options: TMDX and TMDS. These prefixes are representative of the evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices (TMS/TMDS). This development flow is defined below.

Device Development Evolutionary Flow:

TMX Experimental device that is not representative of the final device's electrical specifications.

TMP Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification.

TMS Fully qualified production device.

Support Tool Development Evolutionary Flow:

TMDX Development support product that has not yet completed Texas Instruments internal qualification testing.

TMDS Fully qualified development support product.

TMX and TMP devices and TMDX development support tools are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

> **Note:**
>
> Texas Instruments recommends that prototype devices (TMX or TMP) not be used in production systems since their expected end-use failure rate is undefined but predicted to be greater than standard qualified production devices.

TMS devices and TMDS development support tools have been fully characterized and the quality and reliability of the device has been fully demonstrated. Texas Instruments standard warranty applies.

D.2.2 Device and Development Support Tool Nomenclature

In addition to the prefix, the device nomenclature includes a suffix that follows the device family name. This suffix indicates the package type (e.g., N, FN, or GB) and temperature range (e.g., L). Figure D-4 provides a legend for reading the complete device name for any TMS320 family member.

TMS 320 E 15 JD L

PREFIX
SMJ = MIL-STD-883C
TMP = prototype device
TMS = qualified device
TMX = experimental device

DEVICE FAMILY
320 = TMS320 family

TECHNOLOGY
C = CMOS
E = CMOS EPROM
No letter = NMOS

DEVICE
1st-gen. DSP:
10
14
15
17
2nd-gen. DSP:
20
25
3rd-gen. DSP:
30

TEMPERATURE RANGE
A = -40 to 85°C
H = 0 to 50°C
L = 0 to 70°C
M = -55 to 125°C
S = -55 to 100°C

PACKAGE TYPE
FD = leadless ceramic CC
FN = plastic leaded CC
FZ = ceramic leaded CC
GB = ceramic PGA
JD = Ceramic DIP side-brazed
N = plastic DIP

Figure D-4. TMS320 Device Nomenclature

External Peripherals, Sockets and Crystals

This appendix provides product information regarding memories, analog converters, and sockets, which are manufactured by Texas Instruments and compatible with the TMS320C14/E14.

The contents of the major areas in this appendix are listed below.

E.1 TI Peripherals

This section provides pages of product information taken from data sheets for EPROM memories, codecs, analog interface circuits, and D/A and D/A converters.

All of these devices can be interfaced with TMS320C14/E14 processors (see Section 6 for hardware interface designs). Refer to *Digital Signal Processing Applications with the TMS320 Family* (SPRA012A) for additional information on interfaces using memories and analog conversion devices.

The following paragraphs give the name of each device and where the data sheet for that device is located in order to obtain further specification information if desired.

Data sheets for EPROM memories are located in the *MOS Memory Data Book* (SMYD006). The name of the device and the page number in the book on which the device is introduced are listed.

TMS27C64	(page 6-21)
TMS27C128	(page 6-29)
TMS27C256	(page 6-37)
TMX27C512	(page 6-45)

Another EPROM memory, TMS27C291/292, is described in a data sheet (SMLS291A).

The TCM29C13/14/16/17 codecs and filters are described in the data sheet beginning on page 2-111 of the *Telecommunications Circuits Data Book* (SCT001). An analog interface for the DSP using a codec and filter is provided by the TCM29C18/19 (data sheet number SCT021).

The data sheet for the TLC32040 analog interface circuit is provided in the *Interface Circuits Data Book* (SLYD002), beginning on page 2-271.

In the same book are data sheets for A/D and D/A converters. The name of the device and the page on which it is introduced are as follows:

TLC0820	(page 2-113)
TLC1205/1225	(page 2-181)
TLC7524	(page 2-243)

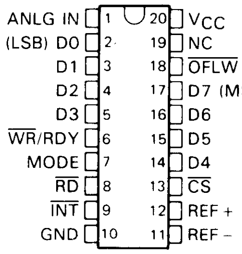
The following pages contain data sheets of the periphals discussed above.

ADC0820B, ADC0820C, TLC0820A, TLC0820B Advanced LinCMOS™ HIGH-SPEED 8-BIT ANALOG-TO-DIGITAL CONVERTERS USING MODIFIED "FLASH" TECHNIQUES

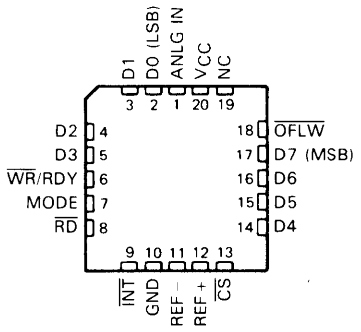
D2873, SEPTEMBER 1986 REVISED MARCH 1988

- Advanced LinCMOS™ Silicon-Gate Technology
- 8-Bit Resolution
- Differential Reference Inputs
- Parallel Microprocessor Interface
- Conversion and Access Time Over Temperature Range
  Write-Read Mode . . . 1.18 μs and 1.92 μs
  Read Mode . . . 2.5 μs Max
- No External Clock or Oscillator Components Required
- On-Chip Track-and-Hold
- Low Power Consumption . . . 50 mW Typ
- Single 5-V Supply
- TLC0820B is Direct Replacement for National Semiconductor ADC0820B/BC and Analog Devices AD7820L/C/U;
  TLC0820A is Direct Replacement for National Semiconductor ADC0820C/CC and Analog Devices AD7820K/B/T

ALL TYPES . . . DW OR N PACKAGE
TLC0820_M . . . J PACKAGE
(TOP VIEW)



TLC0820_M . . . FK PACKAGE
ADC0820_CI, ADC0820_C . . . FN PACKAGE
TLC0820_I, TLC0820_C . . . FN PACKAGE
(TOP VIEW)



NC – No internal connection

description

The ADC0820B, ADC0820C, TLC0820A, and TLC0820B are Advanced LinCMOS™ 8-bit analog-to-digital converters each consisting of two 4-bit "flash" converters, a 4-bit digital-to-analog converter, a summing (error) amplifier, control logic, and a result latch circuit. The modified "flash" technique allows low-power integrated circuitry to complete an 8-bit conversion in 1.18 μs over temperature. The on-chip track-and-hold circuit has a 100-ns sample window and allows these devices to convert continuous analog signals having slew rates of up to 100 mV/μs without external sampling components. TTL-compatible three-state output drivers and two modes of operation allow interfacing to a variety of microprocessors. Detailed information on interfacing to most popular microprocessors is readily available from the factory.

The M-suffix devices are characterized for operation over the full military temperature range of −55 °C to 125 °C. The I-suffix devices are characterized for operation from −40 °C to 85 °C. The C-suffix devices are characterized for operation from 0 °C to 70 °C. See Available Options.

ADVANCE INFORMATION

Advanced LinCMOS is a trademark of Texas Instruments Incorporated.

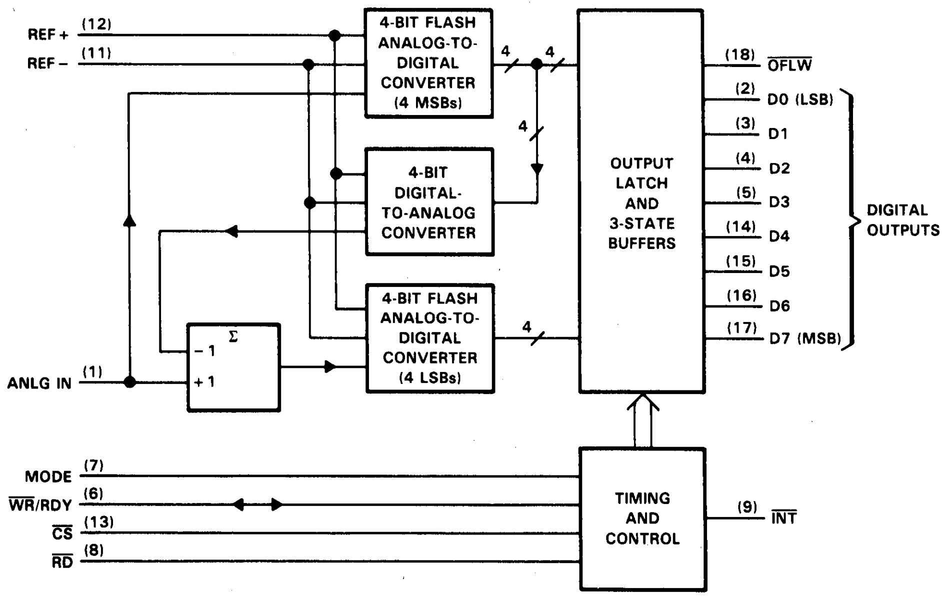
TEXAS INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

AVAILABLE OPTIONS

SYMBOLIZATION†		OPERATING TEMPERATURE RANGE	TOTAL UNADJUSTED ERROR
DEVICE	PACKAGE SUFFIX		
ADC0820BC	DW, FN, N	0°C to 70°C	±0.5 LSB
ADC0820BCI	DW, FN, N	−40°C to 85°C	±0.5 LSB
ADC0820CC	DW, FN, N	0°C to 70°C	±1 LSB
ADC0820CCI	DW, FN, N	−40°C to 85°C	±1 LSB
TLC0820AC	DW, FN, N	0°C to 70°C	±1 LSB
TLC0820AI	DW, FN, N	−40°C to 85°C	±1 LSB
TLC0820AM	DW, FK, J, N	−55°C to 125°C	±1 LSB
TLC0820BC	DW, FN, N	0°C to 70°C	±0.5 LSB
TLC0820BI	DW, FN, N	−40°C to 85°C	±0.5 LSB
TLC0820BM	DW, FK, J, N	−55°C to 125°C	±0.5 LSB

†In many instances, these ICs may have both ADC0820 and TLC0820 labeling on the package.

functional block diagram



PIN		DESCRIPTION
NAME	NUMBER	
ANLG IN	1	Analog input
$\overline{\text{CS}}$	13	This input must be low in order for $\overline{\text{RD}}$ or $\overline{\text{WR}}$ to be recognized by the ADC.
D0	2	Three-state data output, bit 1 (LSB)
D1	3	Three-state data output, bit 2
D2	4	Three-state data output, bit 3
D3	5	Three-state data output, bit 4
D4	14	Three-state data output, bit 5
D5	15	Three-state data output, bit 6
D6	16	Three-state data output, bit 7
D7	17	Three-state data output, bit 8 (MSB)
GND	10	Ground
$\overline{\text{INT}}$	9	In the WRITE-READ mode, the interrupt output, $\overline{\text{INT}}$, going low indicates that the internal count-down delay time, $t_{d(int)}$, is complete and the data result is in the output latch. $t_{d(int)}$ is typically 800 ns starting after the rising edge of the $\overline{\text{WR}}$ input (see operating characteristics and Figure 3). If $\overline{\text{RD}}$ goes low prior to the end of $t_{d(int)}$, $\overline{\text{INT}}$ goes low at the end of t_{dRIL} and the conversion results are available sooner (see Figure 2). $\overline{\text{INT}}$ is reset by the rising edge of either $\overline{\text{RD}}$ or $\overline{\text{CS}}$.
MODE	7	Mode-selection input. It is internally tied to GND through a 50-μA current source, which acts like a pull-down resistor. READ mode: Occurs when this input is low. WRITE-READ mode: Occurs when this input is high.
NC	19	No internal connection
$\overline{\text{OFLW}}$	18	Normally the $\overline{\text{OFLW}}$ output is a logical high. However, if the analog input is higher than the V_{REF+}, $\overline{\text{OFLW}}$ will be low at the end of conversion. It can be used to cascade 2 or more devices to improve resolution (9 or 10-bits).
$\overline{\text{RD}}$	8	In the WRITE-READ mode with $\overline{\text{CS}}$ low, the 3-state data outputs D0 through D7 are activated when $\overline{\text{RD}}$ goes low. $\overline{\text{RD}}$ can also be used to increase the conversion speed by reading data prior to the end of the internal count-down delay time. As a result, the data transferred to the output latch is latched after the falling edge of $\overline{\text{RD}}$. In the READ mode with $\overline{\text{CS}}$ low, the conversion starts with $\overline{\text{RD}}$ going low. $\overline{\text{RD}}$ also enables the three-state data outputs upon completion of the conversion. The RDY output going into the high-impedance state and $\overline{\text{INT}}$ going low indicates completion of the conversion.
REF −	11	This input voltage is placed on the bottom of the resistor ladder.
REF +	12	This input voltage is placed on the top of the resistor ladder.
V_{CC}	20	Power supply voltage
$\overline{\text{WR}}$/RDY	6	In the WRITE-READ mode with $\overline{\text{CS}}$ low, the conversion is started on the falling edge of the $\overline{\text{WR}}$ input signal. The result of the conversion is strobed into the output latch after the internal count-down delay time, $t_{d(int)}$, provided that the $\overline{\text{RD}}$ input does not go low prior to this time. $t_{d(int)}$ is approximately 800 ns. In the READ mode, RDY (an open-drain output) will go low after the falling edge of $\overline{\text{CS}}$, and will go into the high-impedance state when the conversion is strobed into the output latch. It is used to simplify the interface to a microprocessor system.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

	TLC0820_M	ADC0820_CI TLC0820_I	ADC0820_C TLC0820_C	UNIT
Supply voltage, V_{CC} (see Note 1)	10	10	10	V
Input voltage range, all inputs (see Note 1)	−0.2 to $V_{CC}+0.2$	−0.2 to $V_{CC}+0.2$	−0.2 to $V_{CC}+0.2$	V
Output voltage range, all outputs (see Note 1)	−0.2 to $V_{CC}+0.2$	−0.2 to $V_{CC}+0.2$	−0.2 to $V_{CC}+0.2$	V
Operating free-air temperature range	−55 to 125	−40 to 85	0 to 70	°C
Storage temperature range	−65 to 150	−65 to 150	−65 to 150	°C
Case temperature for 60 seconds: FK package	260			°C
Case temperature for 10 seconds: FN package		260	260	°C
Lead temperature 1,6 mm (1/16 inch) from case for 60 seconds: J package	300			°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds: DW or N package	260	260	260	°C

NOTE 1: All voltages are with respect to network ground terminal, pin 10.

recommended operating conditions

			TLC0820_M			ADC0820_CI TLC0820_I			ADC0820_C TLC0820_C			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}			4.5	5	8	4.5	5	8	4.5	5	8	V
Analog input voltage			−0.1		$V_{CC}+0.1$	−0.1		$V_{CC}+0.1$	−0.1		$V_{CC}+0.1$	V
Positive reference voltage, V_{REF+}			V_{REF-}		V_{CC}	V_{REF-}		V_{CC}	V_{REF-}		V_{CC}	V
Negative reference voltage, V_{REF-}			GND		V_{REF+}	GND		V_{REF+}	GND		V_{REF+}	V
High-level input voltage, V_{IH}	V_{CC} = 4.75 V to 5.25 V	$\overline{CS}$, $\overline{WR}$/RDY, $\overline{RD}$	2			2			2			V
		MODE	3.5			3.5			3.5			
Low-level input voltage, V_{IL}	V_{CC} = 4.75 V to 5.25 V	$\overline{CS}$, $\overline{WR}$/RDY, $\overline{RD}$			0.8			0.8			0.8	V
		MODE			1.5			1.5			1.5	
Delay to next conversion, $t_{d(NC)}$ (see Figures 1, 2, 3, and 4)			500			500			500			ns
Delay time from $\overline{WR}$ to $\overline{RD}$ in write-read mode, t_{dWR} (see Figure 2)			0.4			0.4			0.4			µs
Write-pulse duration in write-read mode, t_{wW} (see Figures 2, 3, and 4)			0.5		50	0.5		50	0.5		50	µs
Operating free-air temperature, T_A			−55		125	−40		85	0		70	°C

electrical characteristics at specified operating free-air temperature, V_{CC} = 5 V (unless otherwise noted)

PARAMETER			TEST CONDITONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	Any D, $\overline{INT}$, or $\overline{OFLW}$	V_{CC} = 4.75 V, I_{OH} = −360 μA	Full range	2.4			V
			V_{CC} = 4.75 V, I_{OH} = −10 μA	Full range	4.5			
				25°C	4.6			
V_{OL}	Low-level output voltage	Any D, $\overline{OFLW}$, $\overline{INT}$, or $\overline{WR}$/RDY	V_{CC} = 5.25 V, I_{OL} = 1.6 mA	Full range			0.4	V
				25°C			0.34	
I_{IH}	High-level input current	$\overline{CS}$ or $\overline{RD}$	V_{IH} = 5 V	Full range		0.005	1	μA
		$\overline{WR}$/RDY		Full range			3	
				25°C		0.1	0.3	
		MODE		Full range			200	
				25°C		50	170	
I_{IL}	Low-level input current	$\overline{CS}$, $\overline{WR}$/RDY, $\overline{RD}$, or MODE	V_{IL} = 0	Full range		−0.005	−1	μA
I_{OZ}	Off-state (high-impedance state) output current	Any D or $\overline{WR}$/RDY	V_O = 5 V	Full range			3	μA
				25°C		0.1	0.3	
			V_O = 0	Full range			−3	
				25°C		−0.1	−0.3	
I_I	Analog input current		$\overline{CS}$ at 5 V, V_I = 5 V	Full range			3	μA
				25°C			0.3	
			$\overline{CS}$ at 5 V, V_I = 0	Full range			−3	
				25°C			−0.3	
I_{OS}	Short-circuit output current	Any D, $\overline{OFLW}$, $\overline{INT}$, or $\overline{WR}$/RDY	V_O = 5 V	Full range	7			mA
				25°C	8.4	14		
		Any D or $\overline{OFLW}$	V_O = 0	Full range	−6			
				25°C	−7.2	−12		
		$\overline{INT}$		Full range	−4.5			
				25°C	−5.3	−9		
R_{ref}	Reference resistance			Full range	1.25		6	kΩ
				25°C	1.4	2.3	5.3	
I_{CC}	Supply current		$\overline{CS}$, $\overline{WR}$/RDY, and $\overline{RD}$ at 0 V	Full range			15	mA
				25°C		7.5	13	
C_i	Input capacitance	Any digital		Full range		5		pF
		ANLG IN				45		
C_o	Output capacitance	Any digital		Full range			5	pF

†All typical values are at T_A = 25°C.

ADVANCE INFORMATION

operating characteristics, V_{CC} = 5 V, V_{REF+} = 5 V, V_{REF-} = 0, t_r = t_f = 20 ns, T_A = 25°C (unless otherwise noted)

PARAMETER		TEST CONDITIONS		ADC0820B TLC0820B			ADC0820C TLC0820A			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
k_{SVS}	Supply voltage sensitivity	V_{CC} = 5 V ± 5%, T_A = MIN to MAX			±1/16	±1/4		±1/16	±1/4	LSB
	Total unadjusted error[†]	MODE pin at 0 V, T_A = MIN to MAX				1/2			1	LSB
t_{convR}	Read mode conversion time	MODE pin at 0 V, See Figure 1			1.6	2.5		1.6	2.5	μs
$t_{d(int)}$	Internal count-down delay time	MODE pin at 5 V, C_L = 50 pF, See Figures 3 and 4			800	1300		800	1300	ns
t_{aR}	Access time from $\overline{RD}$↓	MODE pin at 0 V, See Figure 1			t_{convR} + 20	t_{convR} + 50		t_{convR} + 20	t_{convR} + 50	ns
t_{aR1}	Access time from $\overline{RD}$↓	MODE pin at 5 V, $t_{dWR} < t_{d(int)}$, See Figure 2	C_L = 15 pF		190	280		190	280	ns
			C_L = 100 pF		210	320		210	320	
t_{aR2}	Access time from $\overline{RD}$↓	MODE pin at 5 V, $t_{dWR} > t_{d(int)}$ See Figure 3	C_L = 15 pF		70	120		70	120	ns
			C_L = 100 pF		90	150		90	150	
t_{aINT}	Access time from $\overline{INT}$↓	MODE pin at 5 V, See Figure 4			20	50		20	50	ns
t_{dis}	Disable time from $\overline{RD}$↑	R_L = 1 kΩ, C_L = 10 pF, See Figures 1, 2, 3, and 5			70	95		70	95	ns
t_{dRDY}	Delay time from $\overline{CS}$↓ to RDY↓	MODE pin at 0 V, C_L = 50 pF, See Figure 1			50	100		50	100	ns
t_{dRIH}	Delay time from $\overline{RD}$↑ to $\overline{INT}$↑	C_L = 50 pF, See Figures 1, 2, and 3			125	225		125	225	ns
t_{dRIL}	Delay time from $\overline{RD}$↓ to $\overline{INT}$↓	MODE pin at 5 V, $t_{dWR} < t_{d(int)}$, See Figure 2			200	290		200	290	ns
t_{dWIH}	Delay time from $\overline{WR}$↑ to $\overline{INT}$↑	MODE pin at 5 V, C_L = 50 pF, See Figure 4			175	270		175	270	ns
	Slew rate tracking				0.1			0.1		V/μs

[†] Total unadjusted error includes offset, full-scale, and linearity errors.

ADVANCE INFORMATION

PARAMETER MEASUREMENT INFORMATION

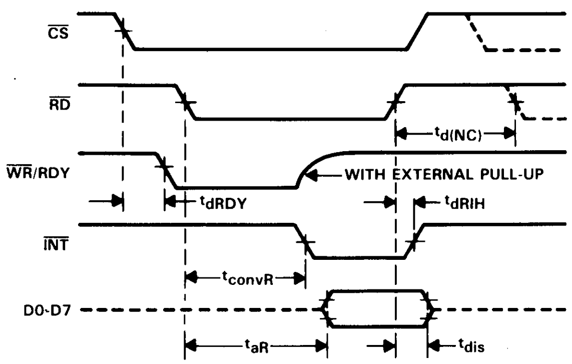


FIGURE 1. READ MODE WAVEFORMS (MODE PIN LOW)

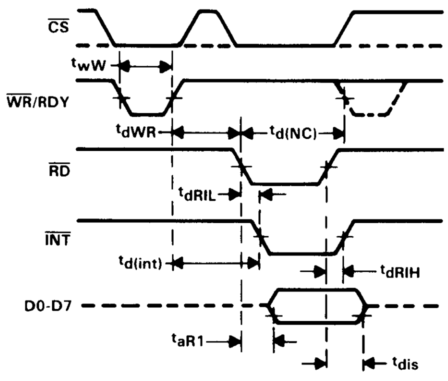


FIGURE 2. WRITE-READ MODE WAVEFORMS [MODE PIN HIGH AND $t_{dWR} < t_{d(int)}$]

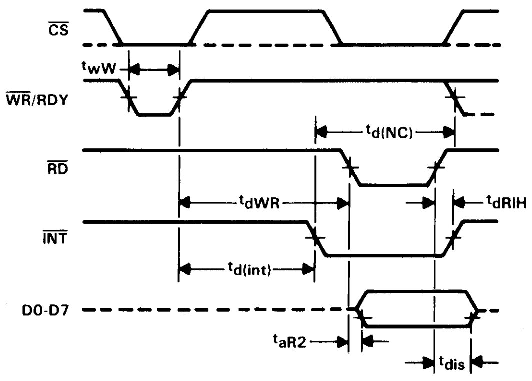


FIGURE 3. WRITE-READ WAVEFORMS [MODE PIN HIGH AND $t_{dWR} > t_{d(int)}$]

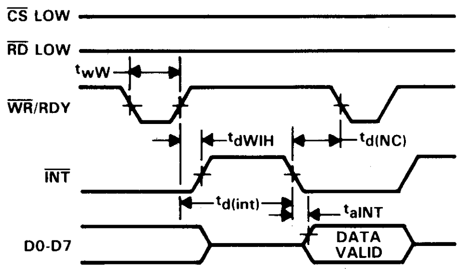
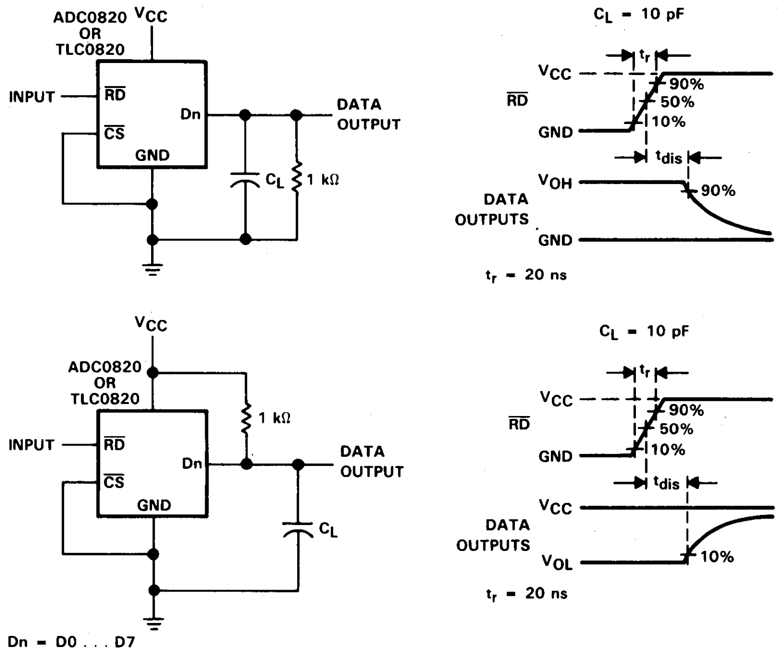


FIGURE 4. WRITE-READ MODE WAVEFORMS (STAND-ALONE OPERATION, MODE PIN HIGH, AND RD LOW)

ADVANCE INFORMATION

PARAMETER MEASUREMENT INFORMATION



TEST CIRCUIT

VOLTAGE WAVEFORMS

FIGURE 5. TEST CIRCUIT AND VOLTAGE WAVEFORMS

PRINCIPLES OF OPERATION

The ADC0820B, ADC0820C, TLC0820A, and TLC0820B each employ a combination of "sampled-data" comparator techniques and "flash" techniques common to many high-speed converters. Two 4-bit "flash" analog-to-digital conversions are used to give a full 8-bit output.

The recommended analog input voltage range for conversion is -0.1 V to $V_{CC}+0.1$ V. Analog input signals that are less than V_{REF-} + ½ LSB or greater than V_{REF+} − ½ LSB convert to 00000000 or 11111111 respectively. The reference inputs are fully differential with common-mode limits defined by the supply rails. The reference input values define the full-scale range of the analog input. This allows the gain of the ADC to be varied for ratiometric conversion by changing the V_{REF+} and V_{REF-} voltages.

The device operates in two modes, read (only) and write-read, which are selected by the MODE pin (pin 7). The converter is set to the read (only) mode when pin 7 is low. In the read mode, the $\overline{\text{WR}}$/RDY pin is used as an output and is referred to as the "ready" pin. In this mode, a low on the "ready" pin while $\overline{\text{CS}}$ is low indicates that the device is busy. Conversion starts on the falling edge of $\overline{\text{RD}}$ and is completed no more than 2.5 μs later when $\overline{\text{INT}}$ falls and the "ready" pin returns to a high-impedance state. Data outputs also change from high-impedance to active states at this time. After the data is read, $\overline{\text{RD}}$ is taken high, $\overline{\text{INT}}$ returns high, and the data outputs return to their high-impedance states.

The converter is set to the write-read mode when pin 7 is high and $\overline{\text{WR}}$/RDY is referred to as the "write" pin. Taking $\overline{\text{CS}}$ and the "write" pin low selects the converter and initiates measurement of the input signal. Approximately 600 ns after the "write" pin returns high, the conversion is completed. Conversion starts on the rising edge of $\overline{\text{WR}}$/RDY in the write-read mode.

The high-order 4-bit "flash" ADC measures the input by means of 16 comparators operating simultaneously. A high precision 4-bit DAC then generates a discrete analog voltage from the result of that conversion. After a time delay, a second bank of comparators does a low-order conversion on the analog difference between the input level and the high-order DAC output. The results from each of these conversions enter an 8-bit latch and are output to the three-state buffers on the falling edge of $\overline{\text{RD}}$.

ADVANCE INFORMATION

TYPICAL APPLICATION DATA

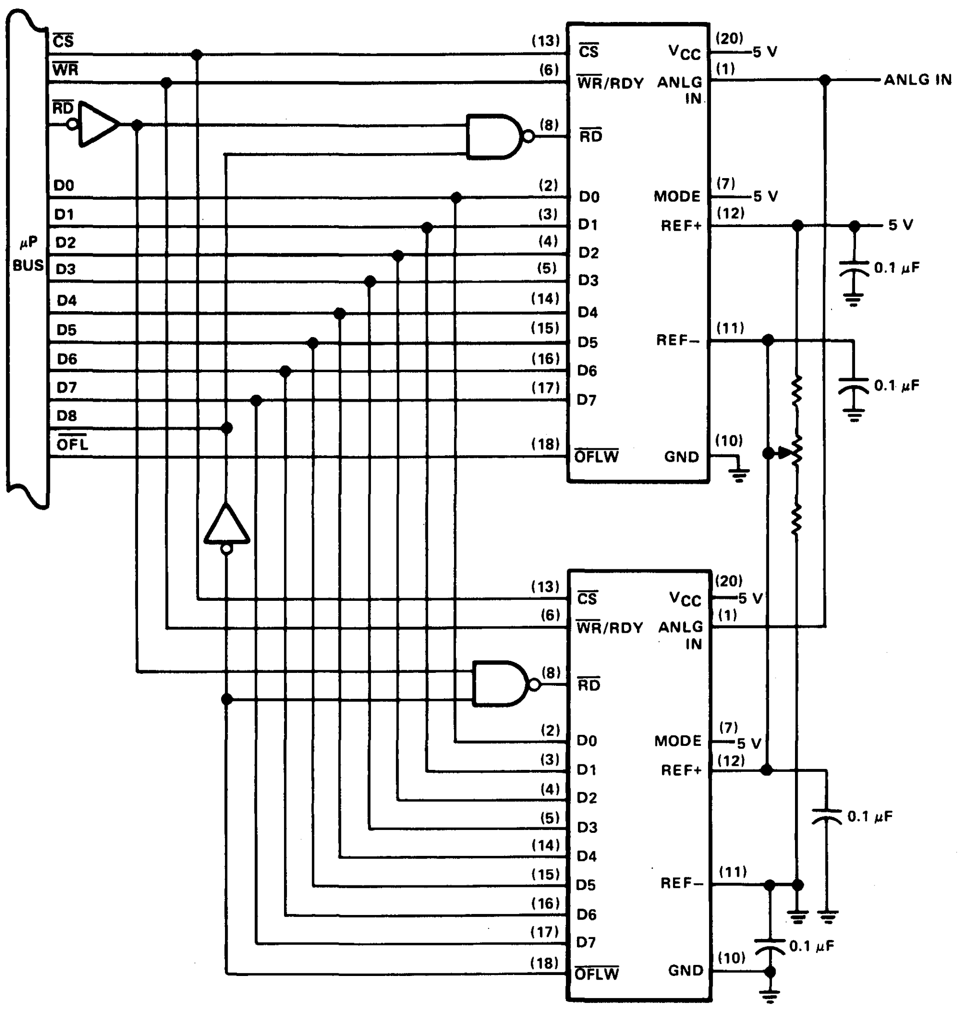


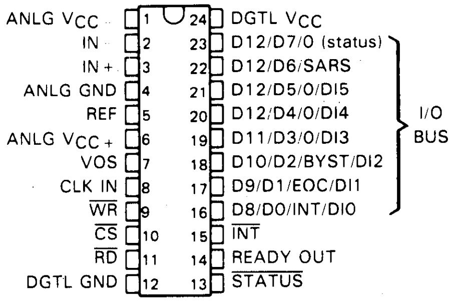
FIGURE 6. CONFIGURATION FOR 9-BIT RESOLUTION

TLC1205A, TLC1205B, TLC1225A, TLC1225B SELF-CALIBRATING 12-BIT-PLUS-SIGN UNIPOLAR OR BIPOLAR ANALOG-TO-DIGITAL CONVERTERS

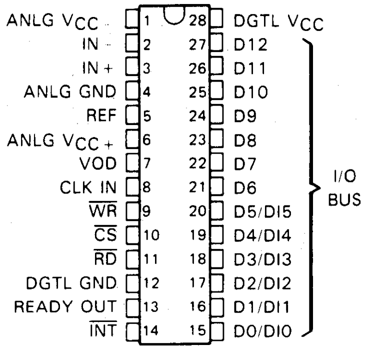
D2982, FEBRUARY 1987 – REVISED AUGUST 1988

- Advanced LinCMOS™ Technology
- Self-Calibration Eliminates Expensive Trimming at Factory and Offset Adjustment in the Field
- 12-Bit Plus Sign Bipolar or 12-Bit Unipolar
- ±1/2 and ±1 LSB Linearity Error in Unipolar Configuration
- 10 µs Conversion Time (Mode 2) (clock = 2.6 MHz)
  20 µs Conversion Time (Mode 1) (clock = 2.6 MHz)
- Compatible with All Microprocessors
- True Differential Analog Voltage Inputs
- 0 to 5 V Analog Voltage Range with Single 5-V Supply (Unipolar Configuration)
- −5 V to 5 V Analog Voltage Range with ±5-V Supplies (Bipolar Configuration)
- Low Power . . . 25 mW Maximum
- Replaces National Semiconductor ADC1205 and ADC1225 in Mode 1 Operation

TLC1205
J OR N DUAL-IN-LINE PACKAGE
(TOP VIEW)



TLC1225
J OR N DUAL-IN-LINE PACKAGE
(TOP VIEW)



description

The TLC1205 and TLC1225 converters are manufactured with Texas Instruments highly efficient Advanced LinCMOS™ technology. Either of the TLC1205 or TLC1225 CMOS analog-to-digital converters can be operated as a unipolar or bipolar converter. A unipolar input (0 to 5 V) can be accommodated with a single 5-V supply; a bipolar input (−5 V to 5 V) requires the addition of a 5-V negative supply. Conversion is performed via the successive-approximation method. The 24-pin TLC1205 outputs the converted data in two 8-bit bytes; the TLC1225 outputs the converted data in a parallel word and interfaces directly to a 16-bit data bus. Negative numbers are given in the two's complement data format. All digital signals are fully TTL and CMOS compatible.

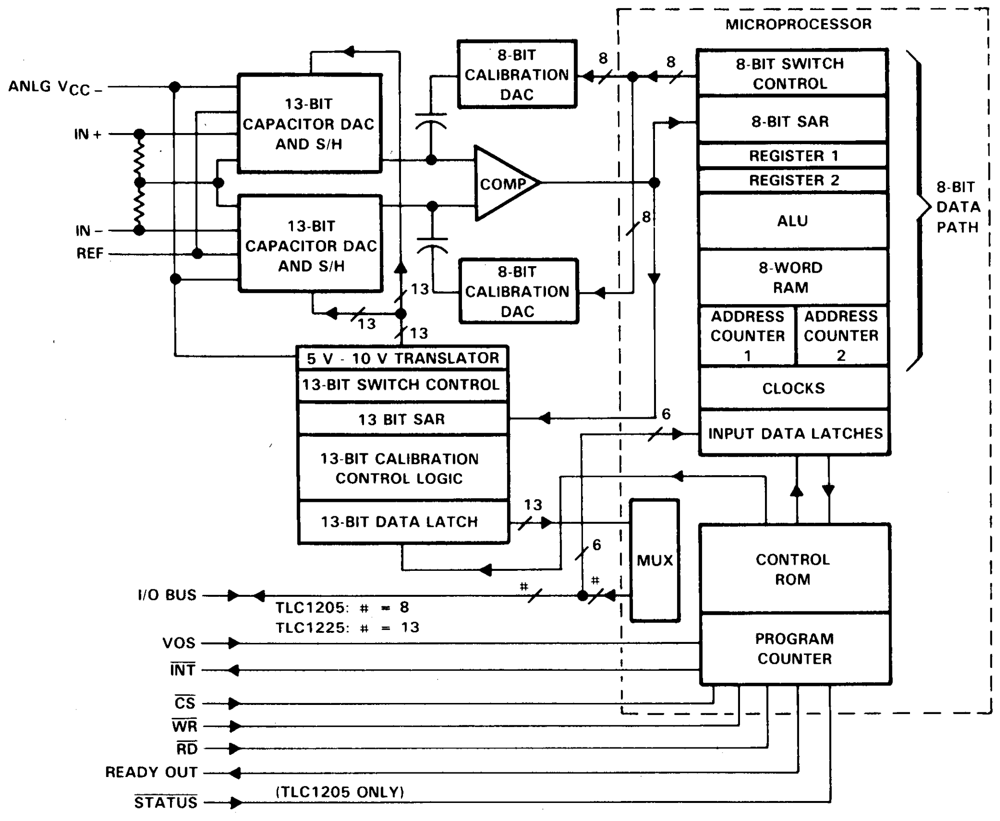
These converters utilize a self-calibration technique by which seven of the internal capacitors in the capacitive ladder of the A/D conversion circuitry can be automatically or manually calibrated. If the converters are operated in Mode 1, one of the seven internal capacitors is calibrated during the first part of the conversion sequence. For example, one capacitor is calibrated during the first conversion. The next capacitor is calibrated during the second conversion. If the converters are operated in Mode 2, the internal capacitors are calibrated during a nonconversion, capacitor-calibrate cycle in which all seven of the internal capacitors are calibrated at the same time. A Mode 2 conversion requires only 10 µs (2.6 MHz clock) after the nonconversion, capacitor-calibrating cycle has been completed. The calibration or conversion cycle may be initiated at any time by issuing the proper address to the data bus. The self-calibrating techniques eliminate the need for expensive trimming of thin-film resistors at the factory and provide excellent performance at low cost.

description (continued)

In Mode 1, these converters are replacements for National Semiconductor ADC1205 and ADC1225 integrated circuits. The Mode 1 conversion time for specified accuracy is 51 clock cycles. In the Mode 2 operation, these devices are no longer true replacements. However, the Mode 2 conversion time for specified accuracy is only 26 clock cycles.

The TLC1205AM, TLC1205BM, TLC1225AM, and TLC1225BM are characterized for operation over the full military temperature range of −55°C to 125°C. The TLC1205AI, TLC1205BI, TLC1225AI, and TLC1225BI are characterized for operation from −40°C to 85°C.

functional block diagram



TEXAS INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

operation description

calibration of comparator offset

The following actions are performed to calibrate the comparator offset:

1. The IN+ and IN− inputs are internally shorted together in order that the comparator input is zero. A course comparator offset calibration is performed by storing the offset voltages of the interconnecting comparator stages on the coupling capacitors that connect the interconnecting stages. Refer to Figure 1. The storage of offset voltages is accomplished by closing all switches and then opening switches A and A', then switches B and B', and then C and C'. This process continues until all interconnecting stages of the comparator are calibrated. After this action, some of the comparator offset still remains uncalibrated.

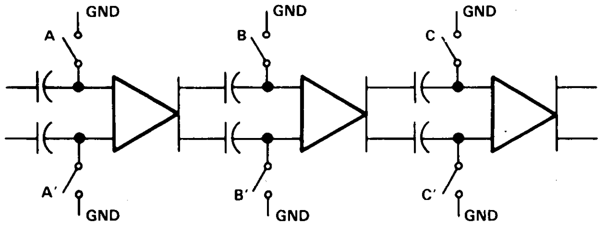


FIGURE 1

2. An A/D conversion is done on the remaining offset with the 8-bit calibration DACs and 8-bit SAR and the result is stored in the RAM.

capacitor calibration of the ADC's capacitive ladder

The following actions are performed to calibrate capacitors in the 13-bit DACs that comprise the ADC's capacitive ladder:

1. The IN+ and IN− inputs are internally disconnected from the 13-bit capacitive DACs.
2. The most significant bit (MSB) capacitor is tied to REF, while the rest of the ladder capacitors are tied to GND. The A/D conversion result for the remaining comparator offset, obtained in Step 2 above, is retrieved from the RAM and is input to the 8-bit DACs.
3. Step 1 of the Calibration of Comparator Offset sequence is performed. The 8-bit DAC input is returned to zero and the remaining comparator offset is then subtracted. Thus, the comparator offset is completely corrected.
4. Now the MSB capacitor is tied to GND, while the rest of the ladder capacitors, C_X, are tied to REF. An MSB capacitor voltage error (see Figure 2) on the comparator output will occur if the MSB capacitor does not equal the sum of the other capacitors in the capacitive ladder. This error voltage is converted to an 8-bit word from which a capacitor error is computed and stored in the RAM.
5. The capacitor voltage error for the next most significant capacitor is calibrated by keeping the MSB capacitor grounded and then performing the above Steps 1-4 while using the next most significant capacitor in lieu of the MSB capacitor. The seven most significant capacitors can be calibrated in this manner.

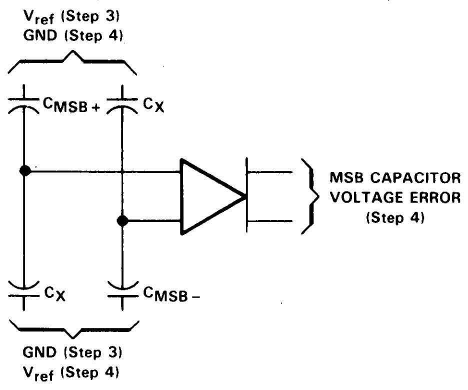


FIGURE 2

analog-to-digital conversion

The following steps are performed in the analog-to-digital conversion process:

1. Step 1 of the Calibration of Comparator Offset Sequence is performed. The A/D conversion result for the remaining comparator offset, which was obtained in Step 2 of the Calibration of Comparator Offset, is retrieved from the RAM and is input to the 8-bit DACs. Thus the comparator offset is completely corrected.
2. IN+ and IN− are sampled onto the 13-bit capacitive ladders.
3. The 13-bit analog-to-digital conversion is performed. As the successive-approximation conversion proceeds successively through the seven most significant capacitors, the error for each of these capacitors is recovered from the RAM and accumulated in a register. This register controls the 8-bit DACs so the total accumulated error for these capacitors is subtracted out during the conversion process.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage (ANLG V_{CC+} and DGTL V_{CC}) (see Note 1)	15 V
Supply voltage, ANLG V_{CC-}	−15 V
Control and Clock input voltage range	−0.3 V to +15 V
Analog input (IN+, IN−) voltage range, V_{I+} and V_{I-}	ANLG V_{CC-} −0.3 V to ANLG V_{CC+} +0.3 V
Reference voltage range, V_{ref}	−0.3 V to ANLG V_{CC+} +0.3 V
Mode select voltage range, V_{OS}	−0.3 V to ANLG V_{CC+} +0.3 V
Output voltage range	−0.3 V to DGTL V_{CC} +0.3 V
Input current (per pin)	±5 mA
Input current (per package)	±20 mA
Operating free-air temperature range: TLC1205AM, TLC1205BM, TLC1225AM, TL1225BM	−55°C to 125°C
TLC1205AI, TLC1205BI, TLC1225AI, TLC1225BI	−40°C to 85°C
Storage temperature range	−65°C to 150°C
Lead temperature 1,6 mm (1/16 inch) from the case for 60 seconds: J package	300°C
Lead temperature 1,6 mm (1/16 inch) from the case for 10 seconds: N package	260°C

NOTE 1: All analog voltages are referred to ANLG GND and all digital voltages are referred to DGTL GND.

recommended operating conditions

		MIN	MAX	UNIT
Supply voltage	ANLG V_{CC+}	4.5	6	V
	ANLG V_{CC-}	−5.5	ANLG GND	
	DGTL V_{CC}	4.5	6	
High-level input voltage, V_{IH}, all digital inputs except CLK IN (V_{CC} = 4.75 V to 5.25 V)		2		V
Low-level input voltage, V_{IL}, all digital inputs except CLK IN (V_{CC} = 4.75 V to 5.25 V)			0.8	V
Analog input voltage, V_{I+}, V_{I-}	Bipolar range	ANLG V_{CC-} − 0.05	ANLG V_{CC+} + 0.05	V
	Unipolar range	ANLG GND − 0.05	ANLG V_{CC+} + 0.05	
Clock input frequency, f_{clock}		0.3	2.6	MHz
Clock duty cycle		40%	60%	
Pulse duration, $\overline{CS}$ and $\overline{WR}$ both low, t_w ($\overline{CS}\cdot\overline{WR}$)		50		ns
Setup time before $\overline{WR}$↑ or $\overline{CS}$↑, t_{su}			50	ns
Hold time after $\overline{WR}$↑ or $\overline{CS}$↑, t_h			50	ns
Operating free-air temperature, T_A	TLC1205AM, TLC1225AM TLC1205BM, TLC1225BM	−55	125	°C
	TLC1205AI, TLC1225AI TLC1205BI, TLC1225BI	−40	85	

electrical characteristics over recommended operating free-air temperature range, ANLG V_{CC+} = DGTL V_{CC} = V_{ref} = 5 V, ANLG V_{CC-} = −5 V (for bipolar input range), ANLG V_{CC-} = ANLG GND (for unipolar input range) (unless otherwise noted) (see Note 1)

PARAMETER		TEST CONDITIONS		MIN	MAX	UNIT
V_{OH}	High-level output voltage	DGTL V_{CC} = 4.75 V	I_O = −1.8 mA	2.4		V
			I_O = −50 µA	4.5		
V_{OL}	Low-level output voltage	DGTL V_{CC} = 4.75 V,	I_O = 8 mA		0.4	V
V_{T+}	Clock positive-going threshold voltage			2.7	3.5	V
V_{T-}	Clock negative-going threshold voltage			1.4	2.1	V
V_{hys}	Clock input hysteresis	V_{T+} min − V_{T-} −max		0.6		V
		V_{T+} max − V_{T-} min			2.1	
r_{ref}	Input resistance, REF terminal			1	10	MΩ
I_{IH}	High-level input current	V_I = 5 V			1	µA
I_{IL}	Low-level input current	V_I = 0			−1	µA
I_{OZ}	High-impedance-state output leakage current	V_O = 0			−3	µA
		V_O = 5 V			3	
I_O	Output current	V_O = 0			−6	mA
		V_O = 5 V			8	
DGTL I_{CC}	Supply current from DGTL V_{CC}	f_{clk} = 2.6 MHz,	$\overline{CS}$ high		3	mA
ANLG I_{CC+}	Supply current from ANLG V_{CC+}	f_{clk} = 2.6 MHz,	$\overline{CS}$ high		3	mA
ANLG I_{CC-}	Supply current from ANLG V_{CC-}	f_{clk} = 2.6 MHz,	$\overline{CS}$ high		−3	mA

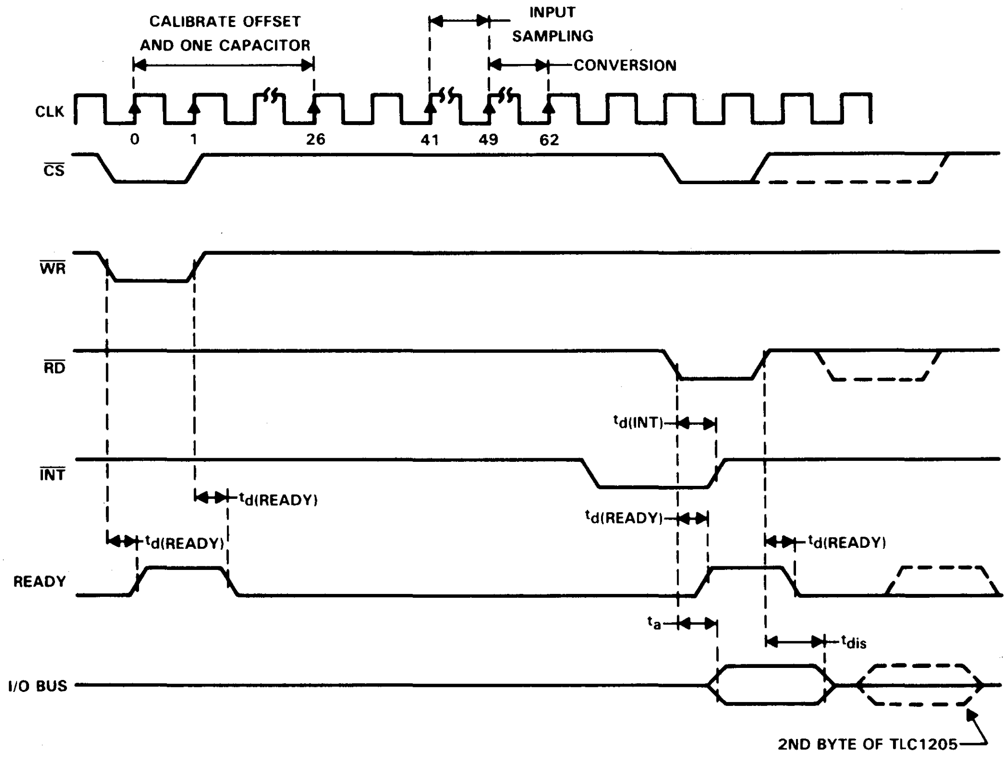
NOTE 1: Bipolar input range is defined as: V_{I+} = −5.05 V to +5.05 V, V_{I-} = −5.05 V to +5.05 V, and $|V_{I+} - V_{I-}| \le 5.05$ V. The unipolar input voltage range is defined as: V_{I+} = −0.05 V to 5.05 V, V_{I-} = −0.05 V to 5.05 V, and $|V_{I+} - V_{I-}| \le 5.05$ V.

operating characteristics over recommended operating free-air temperature range, ANLG V_{CC+} = DGTL V_{CC} = V_{ref} = 5 V, ANLG V_{CC-} = −5 V (for bipolar input range), ANLG V_{CC-} = ANLG GND (for unipolar input range), f_{clock} = 2.6 MHz (unless otherwise noted) (see Note 2)

PARAMETER			TEST CONDITIONS		MIN	MAX	UNIT
	Best-straight-line linearity error (see Note 2)		Unipolar input range	TLC1205A, TLC1225A		±1	LSB
				TLC1205B, TLC1225B		±0.5	
			Bipolar input range	TLC1205A, TLC1225A		±2	
				TLC1205B, TLC1225B		±1.5	
	Zero error					±0.5	LSB
	Adjusted positive and negative full-scale error (see Note 3)		Unipolar input range			±1	LSB
	Adjusted positive and negative full-scale error (see Note 4)		Bipolar input range			±1	LSB
	Temperature coefficient of gain					15	ppm/°C
	Temperature coefficient of offset point					1.5	ppm/°C
k_{SVS}	Supply voltage sensitivity	Zero error	ANLG V_{CC+} = 5 V ± 5%, ANLG V_{CC-} = −5 V ± 5%, DGTL V_{CC} = 5 V ± 5%			±0.75	LSB
		Positive and negative full-scale error				±0.75	
		Linearity error				±0.25	
t_c	Conversion time ($1/f_{clk}$)	Mode 1				62	clock cycles
		Mode 2				24	
t_a	Access time (delay from falling edge of $\overline{CS}\cdot\overline{RD}$ to data output)		C_L = 100 pF			110	ns
t_{dis}	Disable time, output (delay from rising edge of $\overline{RD}$ to high-impedance state		R_L = 10 kΩ,	C_L = 10 pF		60	ns
			R_L = 2 kΩ,	C_L = 100 pF		60	
$t_{d(READY)}$	$\overline{RD}$ or $\overline{WR}$ to READY OUT delay					140	ns
$t_{d(INT)}$	$\overline{RD}$ or $\overline{WR}$ to reset of $\overline{INT}$ delay					400	ns

NOTES: 2. Best-straight-linearity error is the difference between the actual analog value at the transition between any two adjacent steps and its ideal value, after offsset error and gain error have been adjusted to minimize the magnitude of the extreme values of this difference.
3. See section – Positive and Negative Full-Scale Adjustment, Unipolar Inputs.
4. See section – Positive and Negative Full-Scale Adjustment, Bipolar Inputs.

ADVANCE INFORMATION



NOTE: The dashed lines in Figures 3-5 indicate additional control-signalling, which is required for the TLC1205. This additonal signalling is required because of the TLC1205's two-byte operation.

FIGURE 3. MODE 1 TIMING DIAGRAM

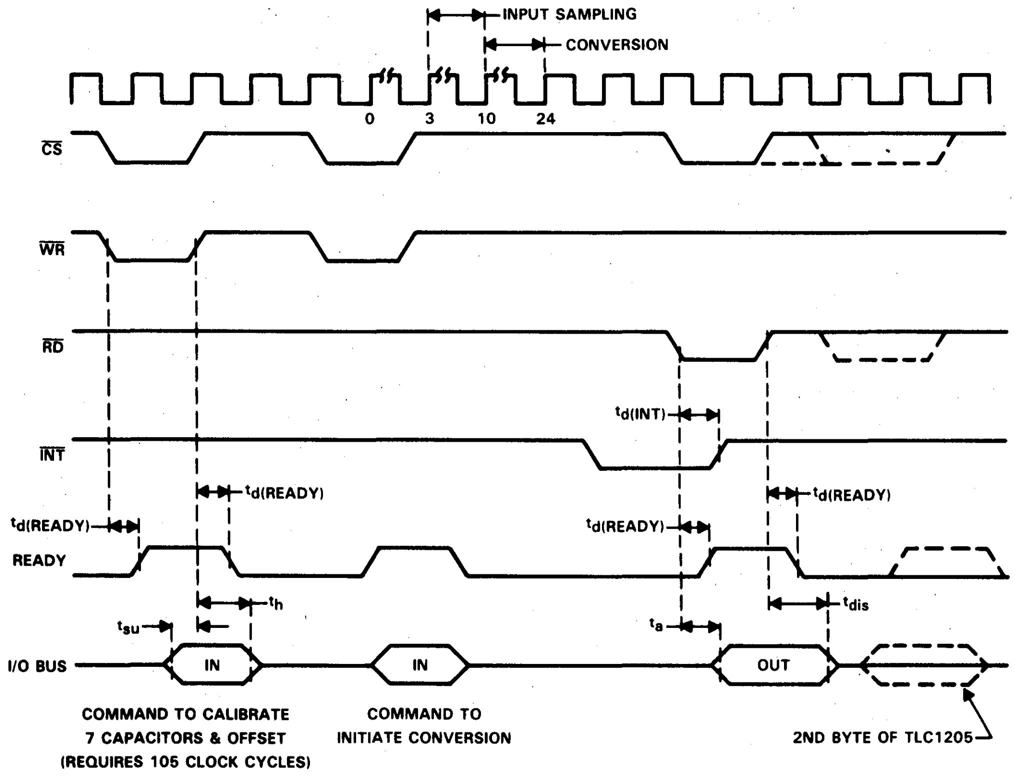


FIGURE 4. MODE 2 TIMING DIAGRAM

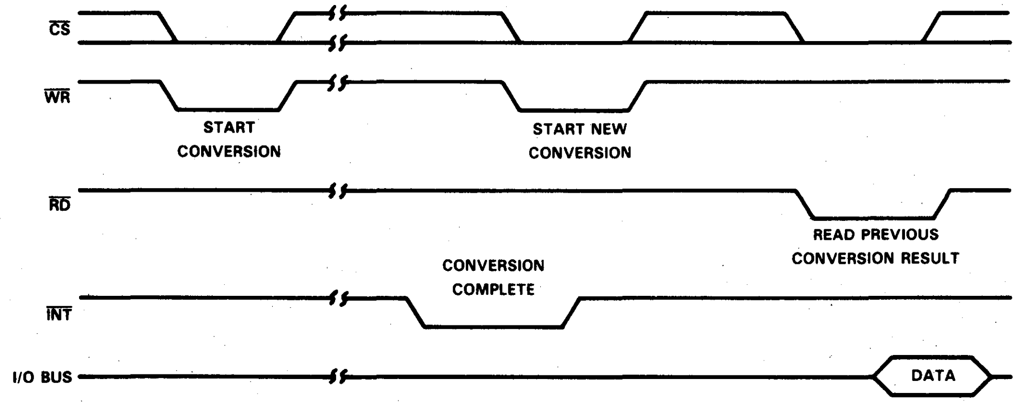


FIGURE 5. MODE 1 – STARTING NEW CONVERSION BEFORE READING PREVIOUS RESULT

PARAMETER MEASUREMENT INFORMATION

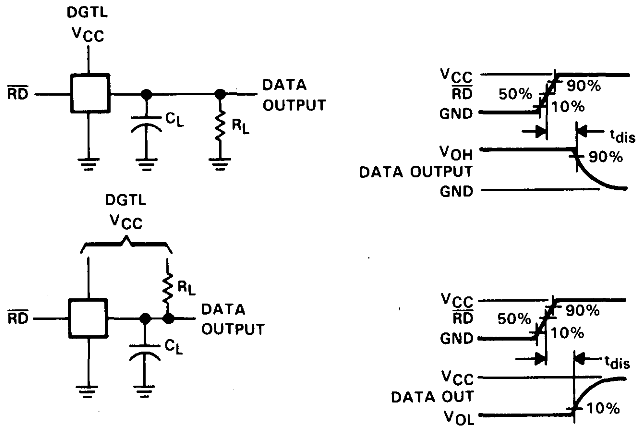


FIGURE 6. LOAD CIRCUITS AND WAVEFORMS

PRINCIPLES OF OPERATION

The following information is categorized into Mode 1 and Mode 2 groupings to allow the designer to concentrate on a particular mode of interest.

power-up calibration sequence

Mode 1

When the chip is powered-up, the internal capacitors are automatically calibrated as part of the power-up sequence. This initial calibration sequence requires 105 clock cycles. The chip will not perform an A/D conversion during this calibration sequence.

Mode 2

Power-Up calibration is not automatic and calibration is initiated by writing control words to the six least significant bits of the data bus. If addressed or initiated, conversion can begin after the first clock cycle. However, full A/D conversion accuracy is not established until after internal capacitor calibration.

conversion start sequence

Mode 1

The conversion sequence is initiated when $\overline{CS}$ and $\overline{WR}$ are both low.

Mode 2

The writing of the conversion command word to the six least significant bits of the data bus, when either $\overline{CS}$ or $\overline{WR}$ goes high, initiates the conversion sequence.

analog sampling sequence

Mode 1

Sampling of the input signal occurs during clock cycles 41 thru 49 of the conversion sequence.

Mode 2

Sampling of the input signal occurs during clock cycles 3 thru 10 of the conversion sequence.

completed A/D conversion

When $\overline{\text{INT}}$ goes low, conversion is complete and the A/D result can be read. A new conversion can begin immediately.

Mode 1

The A/D conversion is complete at the end of clock cycle 62 of the conversion sequence.

Mode 2

The A/D conversion is complete at the end of clock cycle 24 of the conversion sequence.

aborting a conversion in process and beginning a new conversion

Mode 1 and Mode 2

If a conversion is initiated while a conversion sequence is in process, the ongoing conversion will be aborted and a new conversion sequence will begin.

Mode 1

If the new conversion is started before the Analog Sampling begins (see Analog Sampling Sequence section and the Mode 1 Timing Diagram), the particular internal capacitor that was being calibrated during the aborted conversion sequence will be calibrated during the new conversion sequence. Otherwise, the next internal capacitor will be calibrated during the new conversion sequence.

reading the conversion result

TLC1205

Upon activating the required control signals to read the conversion result or status information, the appropriate pins are brought out of a high-impedance state and drive the data bus with the proper information. These pins are D12/D7/0 through D8/D0/INT/DI0.

If $\overline{\text{STATUS}}$, $\overline{\text{CS}}$, and $\overline{\text{RD}}$ are all low, status information can be read. The format of the conversion result and status information and the respective pins for output are presented in Table 1.

TABLE 1

BYTES	$\overline{\text{STATUS}}$	$\overline{\text{CS}}$	$\overline{\text{RD}}$	I/O BUS							
				D12/ D7/ 0	D12/ D6/ SARS	D12/ D5/ 0/ DI5	D12/ D4/ 0/ DI4	D11/ D3/ 0/ DI3	D10/ D2/ BYST/ DI2	D9/ D1/ EOC/ DI1	D8/ D0/ INT/ DI0
MSB	H	L	L	D12	D12	D12	D12	D11	D10	D9	D8
LSB	H	L	↑↓	D7	D6	D5	D4	D3	D2	D1	D0
STATUS	L	L	L	L	SARS	L	L	L	BYST	EOC	INT

The status information is described in Table 2.

TABLE 2

STATUS BIT	BIT DESCRIPTION	TO CLEAR BIT
L	The output has no meaning and is low.	
SARS	A high indicates that conversion is in progress.	
BYST	A low indicates that the next conversion result read will be the most significant conversion byte. A high indicates that the next conversion result read will be the least significant conversion byte. The BYST bit is toggled by reading the conversion result bytes. This bit can be cleared with a "status write" instruction.	By a "status write" or toggled by reading a conversion data byte.
EOC	A high indicates that conversion is complete and the conversion data has been transferred to the output latch.	
INT	A high indicates that conversion is complete and the conversion data has been transferred to the output latch and is ready to read.	By reading a conversion data byte, reading the status byte, or a "status write"

With $\overline{\text{STATUS}}$ high, when $\overline{\text{CS}}$ and $\overline{\text{RD}}$ both go low, the most significant byte (MSB) of the conversion result can be read. Then by taking $\overline{\text{RD}}$ high and back low, the least significant byte (LSB) of the conversion result can be read. Subsequently taking $\overline{\text{RD}}$ high and low causes the alternate reading of the MSB and LSB of the conversion result.

The format of the output is extended sign with 2's complement, right justified data. For both unipolar and bipolar cases, the sign bit D12 is low if $V_{I+} - V_{I-}$ is positive and high if $V_{I+} - V_{I-}$ is negative. The format of the conversion result and the respective output pins are presented in Table 2. The format of the conversion result and the respective pins for output are presented in Table 1.

TLC1225

When both $\overline{\text{CS}}$ and $\overline{\text{RD}}$ go low, all 13 bits of conversion data are output to the I/O bus. The format of the output is extended sign with 2's complement, right justified data. Unlike the TLC1205, the TLC1225 does not have internal status information or a $\overline{\text{STATUS}}$ pin. For both unipolar and bipolar cases, the sign bit D12 is low if $V_{I+} - V_{I-}$ is positive and high if $V_{I+} - V_{I-}$ is negative.

general

reset INT

When reading the conversion data, the falling edge of the first low-going combination of $\overline{\text{CS}}$ and $\overline{\text{RD}}$ will reset $\overline{\text{INT}}$. The falling edge of the low-going combination of $\overline{\text{CS}}$ and $\overline{\text{WR}}$ will also reset $\overline{\text{INT}}$.

ready out

For high-speed microprocessors, READY OUT allows the TLC1205 and the TLC1225 to insert a wait state in the microprocessor's read cycle.

status write (TLC1205)

A status write resets the internal logic and status bits and aborts any conversion in process. A status write occurs when $\overline{\text{CS}}$, $\overline{\text{WR}}$, and $\overline{\text{STATUS}}$ are taken low.

reference voltage (V_{ref})

This voltage defines the range for $|V_{I+} - V_{I-}|$. When $|V_{I+} - V_{I-}|$ equals V_{ref}, the highest conversion data value results. When $|V_{I+} - V_{I-}|$ equals 0, the conversion data value is zero. Thus, for a given input, the conversion data changes ratiometrically with changes in V_{ref}.

VOS

This pin is a digital input and is used to select Mode 1 or Mode 2 operation. A logic low selects Mode 1; a logic high selects Mode 2.

In Mode 1, the ICs are true replacements for National Semiconductor's ADC1205 and ADC1225. The ADC1205 and ADC1225 use the VOS pin to adjust zero error. Since the zero error adjustment voltage is below the TLC1205's and TLC1225's maximum acceptable level for a logic low signal, the TLC1205 and TLC1225 ICs are true replacements. Even in Mode 1, the TLC1205's and TLC1225's converted data can be read earlier than the ADC1205's and ADC1225's.

calibration and conversion considerations

Mode 1

Calibration of the seven internal capacitors is an integral part of the A/D conversion. One of the seven internal capacitors is calibrated during the first part of the conversion sequence. For example, one of the capacitors is calibrated during the first conversion. The next capacitor is calibrated during the second conversion. After seven conversions, the pattern for calibrating the internal capacitors repeats. A conversion sequence requires 62 clock cycles.

A conversion is initiated by the low-going combination of $\overline{\text{CS}}$ and $\overline{\text{WR}}$. The conversion sequence is illustrated in the Mode 1 timing diagram.

Mode 2

Calibration of the internal capacitor and A/D conversion are two separate actions. Each action is independently initiated. Mode 2 conversion is much faster than Mode 1, since Mode 2 conversion is not accompanied by the calibration of internal capacitors. In Mode 2, a calibration command that calibrates all seven internal capacitors is normally issued first. A conversion command then initiates the A/D conversion without calibrating the internal capacitors. Subsequent conversions can be performed by issuing additional conversion commands. The calibration and conversion commands are totally inaependent from one another and can be initiated in any order. Calibration and conversion commands require 105 and 24 clock cycles, respectively.

The calibrate and conversion commands are initiated by writing control words on the six least significant bits of the data bus. These control words are written into the IC when either $\overline{CS}$ or $\overline{WR}$ goes high. The initiation of these commands is illustrated in the Mode 2 Timing Diagram. The bit patterns for the commands are shown in Table 3.

TABLE 3. MODE 2 CONVERSION COMMANDS

COMMAND	$\overline{CS} + \overline{WR}$	I/O BUS						REQUIRED NUMBER OF CLOCK CYCLES
		DI5	DI4	DI3	DI2	DI1	DI0	
Conversion	↑	H	L	X	X	X	L	26
Calibrate†	↑	L	X	L	L	L	L	105

†Calibration is lost when clock is stopped.

analog inputs

differential inputs provide common mode rejection

The differential inputs reduce common-mode noise. Common-mode noise is noise common to both IN+ and IN− inputs, such as 60-Hz noise. There is no time interval between the sampling of the IN+ and IN− so these inputs are truly differential. Thus, no conversion errors result from a time interval between the sampling of the IN+ and IN− inputs.

input bypass capacitors

Input bypass capacitors may be used for noise filtering. However, the charge on these bypass capacitors will be depleted during the input sampling sequence when the internal sampling capacitors are charged. Note that the charging of the bypass capacitors through the differential source resistances must keep pace with the charge depletion of the bypass capacitors during the input sampling sequence. Higher source resistances reduce the amount of charging current for the bypass capacitors. Also, note that fast, successive conversion will have the greatest charge depletion effect on the bypass capacitors. Therefore, the above phenomenon becomes more significant as source resistances and the converssion rate (i.e., higher clock frequency and conversion initiation rate) increase.

In addition, if the above phenomenon prevents the bypass capacitors from fully charging between conversions, voltage drops across the source resistances will result due to the ongoing bypass capacitor charging currents. The voltage drops will cause a conversion error. Also, the voltage drops increase with higher $|V_{I+} - V_{I-}|$ values, higher source resistances, and lower charge on the bypass capacitors (i.e., faster conversion rate).

For low-source-resistance applications ($R_{source} < 100\ \Omega$), a 0.001-µF bypass capacitor at the inputs will prevent pickup due to the series lead inductance of a long wire. A 100-ohm resistor can be placed between the capacitor and the output of an operational amplifier to isolate the capacitor from the operational amplifier.

input leads

The input leads should be kept as short as possible, since the coupling of noise and digital clock signals to the inputs can cause errors.

power supply considerations

Noise spikes on the V_{CC} lines can cause conversion error. Low-inductance tantalum capacitors (> 1 µF) with short leads should be used to bypass ANLG V_{CC} and DGTL V_{CC}. A separate regulator for the TLC1205 or TLC1225 and other analog circuitry will greatly reduce digital noise on the supply line.

positive and negative full-scale adjustment

unipolar inputs

Apply a differential input voltage that is 0.5 LSB below the desired analog full-scale voltage (V_{FS}) and adjust the magnitude of the REF input so that the output code is just changing from 0 1111 1111 1110 to 0 1111 1111 1111. If this transition is desired for a different input voltage, the reference voltage can be adjusted accordingly.

bipolar inputs

First, follow the procedure for the Unipolar case.

Second, apply a differential input voltage so that the digital output code is just changing from 1 0000 0000 0001 to 1 0000 0000 0000. Call this actual differential voltage V_X. The ideal differential voltage for this transition is:

$$-V_{FS} + \frac{V_{FS}}{8192} \tag{1}$$

The difference between the actual and ideal differential voltages is:

$$\text{Delta} = V_X - (-V_{FS} + \frac{V_{FS}}{8192}) \tag{2}$$

Then apply a differential input voltage of:

$$V_X - \frac{\text{Delta}}{2} \tag{3}$$

and adjust V_{ref} so the digital output code is just changing from 1 0000 0000 0001 to 1 0000 0000 0000. This procedure produces positive and negative full-scale transitions with symmetrical minimum error.

TYPICAL APPLICATIONS

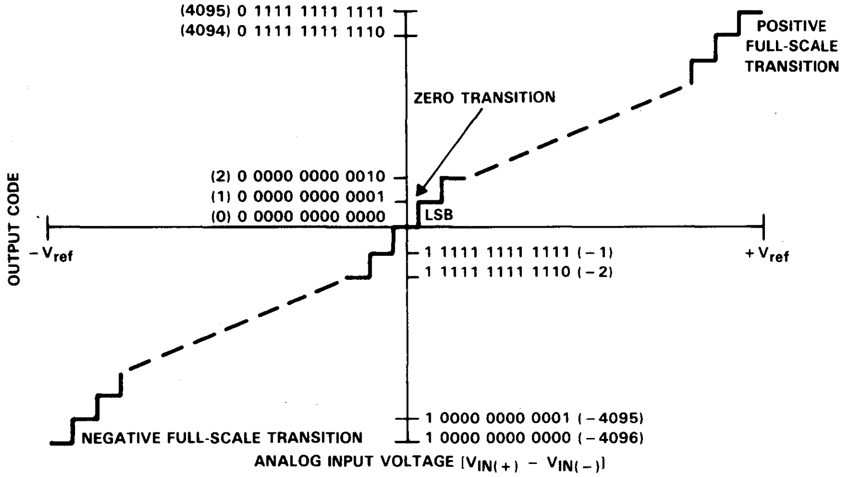
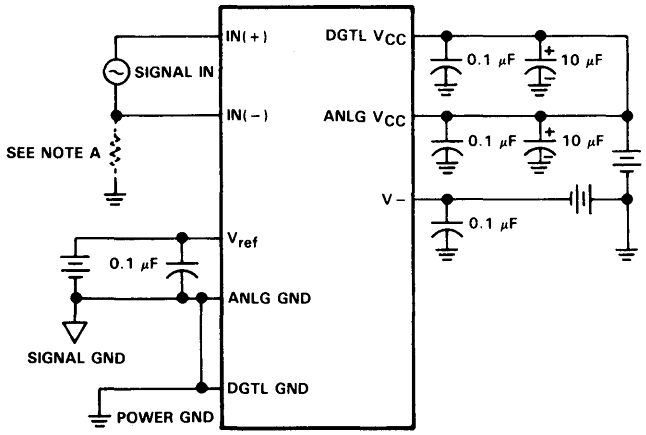


FIGURE 7. TRANSFER CHARACTERISTIC



NOTE: A. The analog input must have some current return path to ANALOG GND.
B. Bypass capacitor leads must be as short as possible.

FIGURE 8. ANALOG CONSIDERATIONS

TYPICAL APPLICATIONS (Continued)

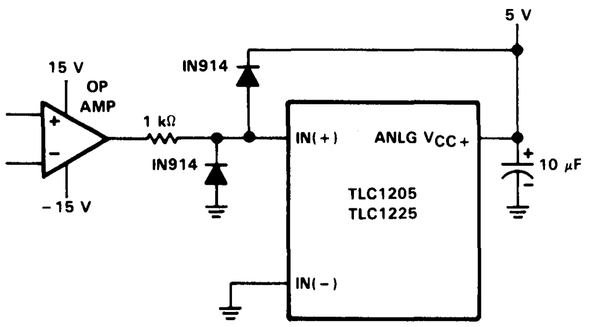
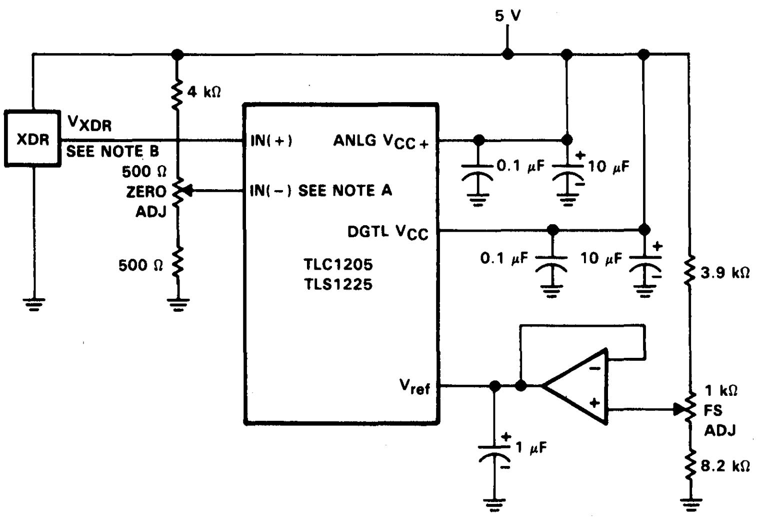


FIGURE 9. INPUT PROTECTION



NOTE: A. $V_{I-} = 0.15 \times$ ANLG V_{CC+}.
B. 15% of ANALOG $V_{CC} \leq V_{XDR} \leq$ 85% of ANALOG V_{CC}.

FIGURE 10. OPERATING WITH RATIOMETRIC TRANSDUCERS

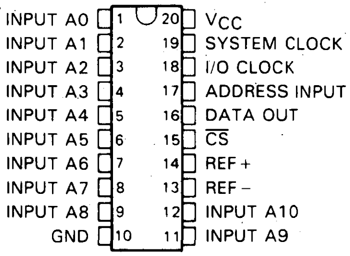
TLC1540M, TLC1540I, TLC1541M, TLC1541I LinCMOS™ 10-BIT ANALOG-TO-DIGITAL PERIPHERALS WITH SERIAL CONTROL AND 11 INPUTS

D2859, DECEMBER 1985—REVISED JANUARY 1988

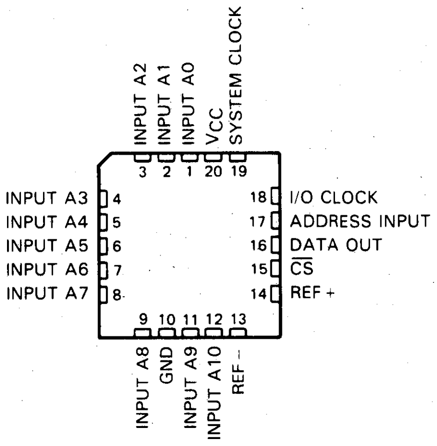
- LinCMOS™ Technology
- 10-Bit Resolution A/D Converter
- Microprocessor Peripheral or Stand-Alone Operation
- On-Chip 12-Channel Analog Multiplexer
- Built-In Self-Test Mode
- Software-Controllable Sample and Hold
- Total Unadjusted Error . . .
  TLC1540: ±0.5 LSB Max
  TLC1541: ±1.0 LSB Max
- Pinout and Control Signals Compatible with TLC540 and TLC549 Families of 8-Bit A/D Converters

TYPICAL PERFORMANCE	
Channel Acquisition Sample Time	5.5 μs
Conversion Time	21 μs
Samples per Second	32×10^3
Power Dissipation	6 mW

J OR N DUAL-IN-LINE PACKAGE
(TOP VIEW)



FK OR FN CHIP CARRIER PACKAGE
(TOP VIEW)



description

The TLC1540 and TLC1541 are LinCMOS™ A/D peripherals built around a 10-bit, switched-capacitor, successive-approximation, A/D converter. They are designed for serial interface to a microprocessor or peripheral via a three-state output with up to four control inputs [including independent System Clock, I/O Clock, Chip Select ($\overline{CS}$), and Address Input]. A 2.1-megahertz system clock for the TLC1540 and TLC1541, with a design that includes simultaneous read/write operation, allows high-speed data transfers and sample rates of up to 32,258 samples per second. In addition to the high-speed converter and versatile control logic, there is an on-chip 12-channel analog multiplexer that can be used to sample any one of 11 inputs or an internal "self-test" voltage, and a sample-and-hold that can operate automatically or under microprocessor control. Detailed information on interfacing to most popular microprocessors is readily available from the factory.

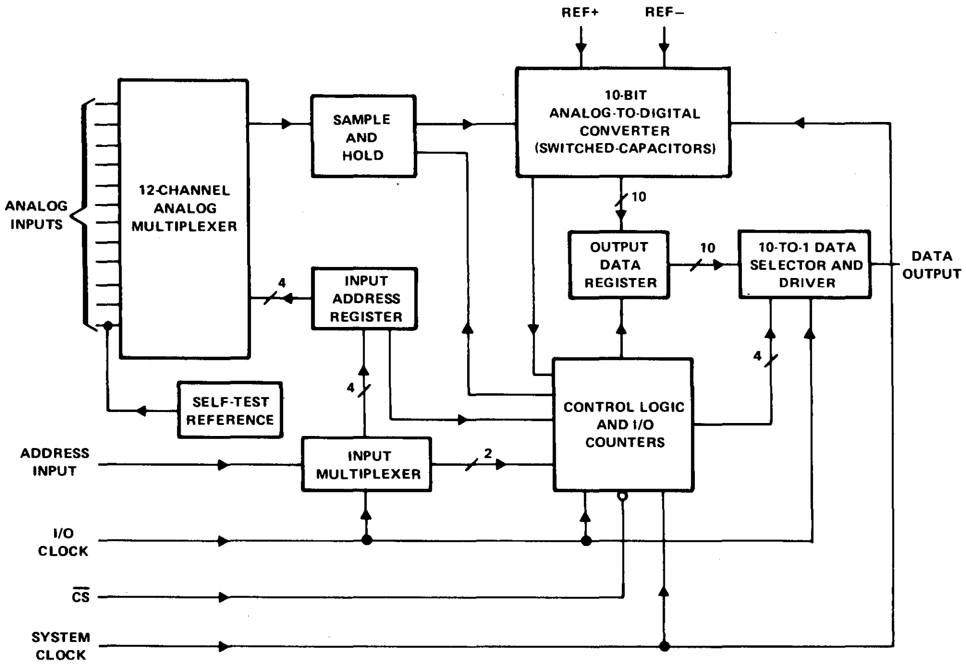
The converters incorporated in the TLC1540 and TLC1541 feature differential high-impedance reference inputs that facilitate ratiometric conversion, scaling, and analog circuitry isolation from logic and supply noises. A totally switched-capacitor design allows guaranteed low-error conversion (±0.5 LSB for the TLC1540, ±1 LSB for the TLC1541) in 21 microseconds over the full operating temperature range.

The TLC1540 and the TLC1541 are available in FK, FN, J, and N packages. The M-suffix versions are characterized for operation from −55°C to 125°C. The I-suffix versions are characterized for operation from −40°C to 85°C.

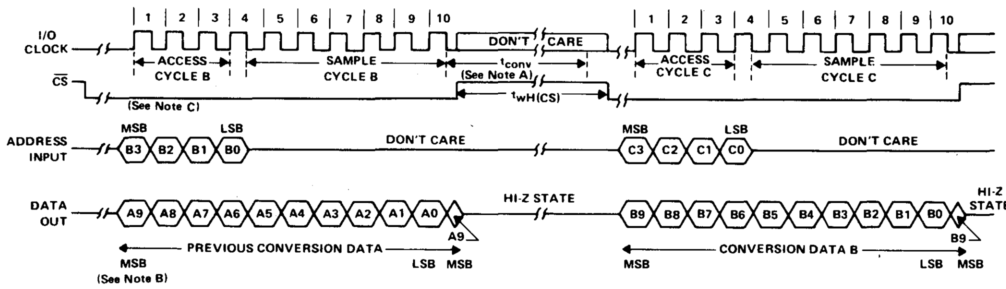
LinCMOS is a trademark of Texas Instruments Incorporated

TEXAS INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

functional block diagram



operating sequence



NOTES: A. The conversion cycle, which requires 44 System Clock periods, is initiated on the 10th falling edge of the I/O Clock↓ after CS↓ goes low for the channel whose address exists in memory at that time. If $\overline{CS}$ is kept low during conversion, the I/O Clock must remain low for at least 44 System Clock cycles to allow conversion to be completed.

B. The most significant bit (MSB) will automatically be placed on the DATA OUT bus after $\overline{CS}$ is brought low. The remaining nine bits (A8-A0) will be clocked out on the first nine I/O Clock falling edges.

C. To minimize errors caused by noise at the CS input, the internal circuitry waits for three System Clock cycles (or less) after a chip-select falling edge is detected before responding to control input signals. Therefore, no attempt should be made to clock-in address data until the minimum chip-select setup time has elapsed.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1)	6.5 V
Input voltage range (any input)	−0.3 V to V_{CC} + 0.3 V
Output voltage range	−0.3 V to V_{CC} + 0.3 V
Peak input current range (any input)	±10 mA
Peak total input current (all inputs)	±30 mA
Operating free-air temperature range: TLC1540I, TLC1541I	−40°C to 85°C
TLC1540M, TLC1541M	−55°C to 125°C
Storage temperature range	−65°C to 150°C
Case temperature for 60 seconds: FK package	260°C
Case temperature for 10 seconds: FN package	260°C
Lead temperature 1,6 mm (1/16 inch) from the case for 60 seconds: J package	300°C
Lead temperature 1,6 mm (1/16 inch) from the case for 10 seconds: N package	260°C

NOTE 1: All voltage values are with respect to digital ground with REF− and GND wired together (unless otherwise noted).

recommended operating conditions

			TLC1540, TLC1541			UNIT
			MIN	NOM	MAX	
Supply voltage, V_{CC}			4.75	5	5.5	V
Positive reference voltage, V_{REF+} (see Note 2)			2.5	V_{CC}	$V_{CC}+0.1$	V
Negative reference voltage, V_{REF-} (see Note 2)			−0.1	0	2.5	V
Differential reference voltage, $V_{REF+} - V_{REF-}$ (see Note 2)			1	V_{CC}	$V_{CC}+0.2$	V
Analog input voltage (see Note 2)			0		V_{CC}	V
High-level control input voltage, V_{IH}			2			V
Low-level control input voltage, V_{IL}					0.8	V
Setup time, address bits before I/O CLK↑, $t_{su(A)}$			400			ns
Hold time, address bits after I/O CLK↑, $t_{h(A)}$			0			ns
Setup time, $\overline{CS}$ low before clocking in first address bit, $t_{su(CS)}$ (see Note 3)			3			System clock cycles
$\overline{CS}$ high during conversion, $t_{wH(CS)}$			44			System clock cycles
Input/Output clock frequency, $f_{CLK(I/O)}$			0		1.1	MHz
System clock frequency, $f_{CLK(SYS)}$			$f_{CLK(I/O)}$		2.1	MHz
System clock high, $t_{wH(SYS)}$			210			ns
System clock low, $t_{wL(SYS)}$			190			ns
Input/Output clock high, $t_{wH(I/O)}$			404			ns
Input/Output clock low, $t_{wL(I/O)}$			404			ns
Clock transition time (see Note 4)	System	$f_{CLK(SYS)} \le 1048$ kHz			30	ns
		$f_{CLK(SYS)} > 1048$ kHz			20	
	I/O	$f_{CLK(I/O)} \le 525$ kHz			100	ns
		$f_{CLK(I/O)} > 525$ kHz			40	
Operating free-air temperature, T_A		TLC1540M, TLC1541M	−55		125	°C
		TLC1540I, TLC1541I	−40		85	

NOTES: 2. Analog input voltages greater than that applied to REF+ convert as all "1"s (11111111), while input voltages less than that applied to REF− convert as all "0"s (00000000). For proper operation, REF+ voltage must be at least 1 volt higher than REF− voltage. Also, the total unadjusted error may increase as this differential reference voltage falls below 4.75 volts.

3. To minimize errors caused by noise at the chip select input, the internal circuitry waits for three System Clock cycles (or less) after a chip select falling edge is detected before responding to control input signals. Therefore, no attempt should be made to clock-in an address until the minimum chip select setup time has elapsed.

4. This is the time required for the clock input signal to fall from V_{IH} min to V_{IL} max or to rise from V_{IL} max to V_{IH} min. In the vicinity of normal room temperature, the devices function with input clock transition time as slow as 2 microseconds for remote data acquisition applications where the sensor and the A/D converter are placed several feet away from the controlling microprocessor.

electrical characteristics over recommended operating temperature range, V_{CC} = V_{REF+} = 4.75 V to 5.5 V (unless otherwise noted), $f_{CLK(I/O)}$ = 1.1 MHz, $f_{CLK(SYS)}$ = 2.1 MHz

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage (pin 16)		V_{CC} = 4.75 V,	I_{OH} = 360 μA	2.4			V
V_{OL}	Low-level output voltage		V_{CC} = 4.75 V,	I_{OL} = 3.2 mA			0.4	V
I_{OZ}	Off-state (high-impedance state) output current		V_O = V_{CC},	$\overline{CS}$ at V_{CC}			10	μA
			V_O = 0,	$\overline{CS}$ at V_{CC}			−10	
I_{IH}	High-level input current		V_I = V_{CC}			0.005	2.5	μA
I_{IL}	Low-level input current		V_I = 0			−0.005	−2.5	μA
I_{CC}	Operating supply current		$\overline{CS}$ at 0 V			1.2	2.5	mA
	Selected channel leakage current		Selected channel at V_{CC}, Unselected channel at 0 V			0.4	1	μA
			Selected channel at 0 V, Unselected channel at V_{CC}			−0.4	−1	
I_{CC} + I_{REF}	Supply and reference current		V_{REF+} = V_{CC},	$\overline{CS}$ at 0 V		1.3	3	mA
C_i	Input capacitance	Analog inputs				7	55	pF
		Control inputs				5	15	

operating characteristics over recommended operating free-air temperature range, $V_{CC} = V_{REF+} = 4.75$ V to 5.5 V, $f_{CLK(I/O)} = 1.1$ MHz, $f_{CLK(SYS)} = 2.1$ MHz

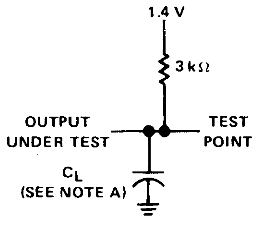
PARAMETER			TEST CONDITIONS	MIN	MAX	UNIT
	Linearity error	TLC1540	See Note 5		±0.5	LSB
		TLC1541			±1	
	Zero error	TLC1540	See Notes 2 and 6		±0.5	LSB
		TLC1541			±1	
	Full-scale error	TLC1540	See Notes 2 and 6		±0.5	LSB
		TLC1541			±1	
	Total unadjusted error	TLC1540	See Note 7		±0.5	LSB
		TLC1541			±1	
	Self-test output code		Input A11 address = 1011 (See Note 8)	0111110100 (500)	1000001100 (524)	
t_{conv}	Conversion time		See Operating Sequence		21	µs
	Total access and conversion time		See Operating Sequence		31	µs
t_{acq}	Channel acquisition time (sample cycle)		See Operating Sequence		6	I/O clock cycles
t_v	Time output data remains valid after I/O clock↓			10		ns
t_d	Delay time, I/O clock↓ to data output valid		See Parameter Measurement Information		400	ns
t_{en}	Output enable time				150	ns
t_{dis}	Output disable time				150	ns
$t_{r(bus)}$	Data bus rise time				300	ns
$t_{f(bus)}$	Data bus fall time				300	ns

† All typical values are at V_{CC} = 5 V, T_A = 25°C.

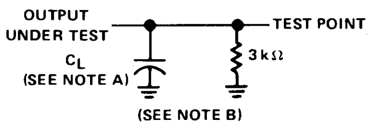
NOTES: 2. Analog input voltages greater than that applied to REF+ convert to all "1"s (11111111), while input voltages less than that applied to REF− convert to all "0"s (00000000). For proper operation, REF+ voltage must be at least 1 volt higher than REF− voltage. Also, the total unadjusted error may increase as this differential reference voltage falls below 4.75 volts.

5. Linearity error is the maximum deviation from the best straight line through the A/D transfer characteristics.
6. Zero error is the difference between 00000000 and the converted output for zero input voltage; full-scale error is the difference between 11111111 and the converted output for full-scale input voltage.
7. Total unadjusted error comprises linearity, zero, and full-scale errors.
8. Both the input address and the output codes are expressed in positive logic. The A11 analog input signal is internally generated and is used for test purposes.

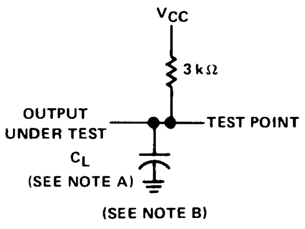
PARAMETER MEASUREMENT INFORMATION



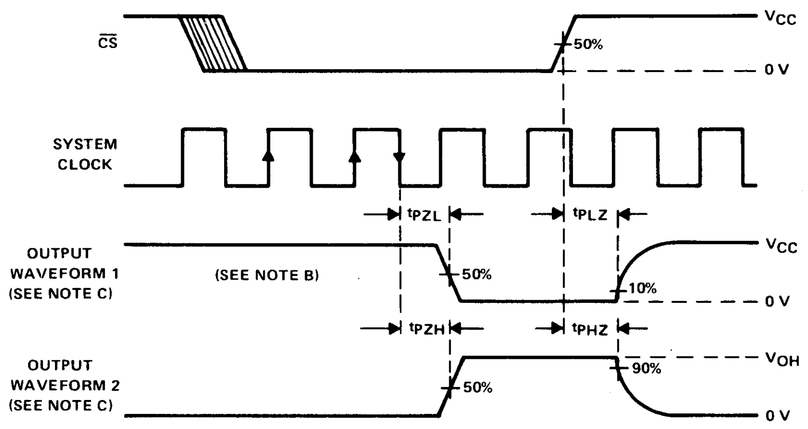
LOAD CIRCUIT FOR t_d, t_r, AND t_f



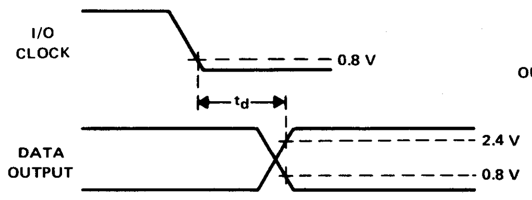
LOAD CIRCUIT FOR t_{PZH} AND t_{PHZ}



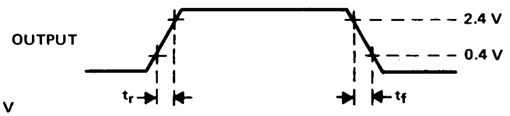
LOAD CIRCUIT FOR t_{PZL} AND t_{PLZ}



VOLTAGE WAVEFORMS FOR ENABLE AND DISABLE TIMES



VOLTAGE WAVEFORM FOR DELAY TIME



VOLTAGE WAVEFORM FOR RISE AND FALL TIMES

NOTES: A. C_L = 50 pF
B. t_{en} = t_{PZH} or t_{PZL}, t_{dis} = t_{PHZ} or t_{PLZ}.
C. Waveform 1 is for an output with internal conditions such that the output is low except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is high except when disabled by the output control.

principles of operation

The TLC1540 and TLC1541 are complete data acquisition systems on single chips. Each includes such functions as sample-and-hold, 10-bit A/D converter, data and control registers, and control logic. For flexibility and access speed, there are four control inputs; Chip Select ($\overline{\text{CS}}$), Address Input, I/O Clock, and System Clock. These control inputs and a TTL-compatible three-state output are intended for serial communications with a microprocessor or microcomputer. The TLC1540 and TLC1541 can complete conversions in a maximum of 21 microseconds, while complete input-conversion-output cycles can be repeated at a maximum of 31 microseconds.

The System and I/O Clocks are normally used independently and do not require any special speed or phase relationships between them. This independence simplifies the hardware and software control tasks for the device. Once a clock signal within the specification range is applied to the System Clock input, the control hardware and software need only be concerned with addressing the desired analog channel, reading the previous conversion result, and starting the conversion by using the I/O Clock. The System Clock will drive the "conversion crunching" circuitry so that the control hardware and software need not be concerned with this task.

When $\overline{\text{CS}}$ is high, the Data Output pin is in a three-state condition and the Address Input and I/O Clock pins are disabled. This feature allows each of these pins, with the exception of the $\overline{\text{CS}}$ pin, to share a control logic point with their counterpart pins on additional A/D devices when additional TLC1540/1541 devices are used. In this way, the above feature serves to minimize the required control logic pins when using multiple A/D devices.

The control sequence has been designed to minimize the time and effort required to initiate conversion and obtain the conversion result. A normal control sequence is:

1. $\overline{\text{CS}}$ is brought low. To minimize errors caused by noise at the $\overline{\text{CS}}$ input, the internal circuitry waits for two rising edges and then a falling edge of the System Clock after a low $\overline{\text{CS}}$ transition, before the low transition is recognized. This technique is used to protect the device against noise when the device is used in a noisy environment. The MSB of the previous conversion result will automatically appear on the Data Out pin.
2. A new positive-logic multiplexer address is shifted in on the first four rising edges of the I/O Clock. The MSB of the address is shifted in first. The negative edges of these four I/O Clock pulses shift out the second, third, fourth, and fifth most significant bits of the previous conversion result. The on-chip sample-and-hold begins sampling the newly addressed analog input after the fourth falling edge. The sampling operation basically involves the charging of internal capacitors to the level of the analog input voltage.
3. Five clock cycles are then applied to the I/O pin and the sixth, seventh, eighth, ninth, and tenth conversion bits are shifted out on the negative edges of these clock cycles.
4. The final tenth clock cycle is applied to the I/O Clock pin. The falling edge of this clock cycle completes the analog sampling process and initiates the hold function. Conversion is then performed during the next 44 System Clock cycles. After this final I/O Clock cycle, $\overline{\text{CS}}$ must go high or the I/O Clock must remain low for at least 44 System Clock cycles to allow for the conversion function.

$\overline{\text{CS}}$ can be kept low during periods of multiple conversion. When keeping $\overline{\text{CS}}$ low during periods of multiple conversion, special care must be exercised to prevent noise glitches on the I/O Clock line. If glitches occur on the I/O Clock line, the I/O sequence between the microprocessor/controller and the device will lose synchronization. Also, if $\overline{\text{CS}}$ is taken high, it must remain high until the end of the conversion. Otherwise, a valid falling edge of $\overline{\text{CS}}$ will cause a reset condition, which will abort the conversion in progress.

A new conversion may be started and the ongoing conversion simultaneously aborted by performing steps 1 through 4 before the 44 System Clock cycles occur. Such action will yield the conversion result of the previous conversion and not the ongoing conversion.

principles of operation (continued)

It is possible to connect the System and I/O Clock pins together in special situations in which controlling circuitry points must be minimized. In this case, the following special points must be considered in addition to the requirements of the normal control sequence previously described.

1. When $\overline{CS}$ is recognized by the device to be at a low level, the common clock signal is used as an I/O Clock. When $\overline{CS}$ is recognized by the device to be at a high level, the common clock signal is used to drive the "conversion crunching" circuitry.
2. The device will recognize a $\overline{CS}$ low transition only when the $\overline{CS}$ input changes and subsequently the System Clock pin receives two positive edges and then a negative edge. For this reason, after a $\overline{CS}$ negative edge, the first two clock cycles will not shift in the address because a low $\overline{CS}$ must be recognized before the I/O Clock can shift in an analog channel address. Also, upon shifting in the address, $\overline{CS}$ must be raised after the eighth I/O Clock that has been recognized by the device, so that a $\overline{CS}$ low level will be recognized upon the lowering of the tenth I/O Clock signal that is recognized by the device. Otherwise, additional common clock cycles will be recognized as I/O Clock pulses and will shift in an erroneous address.

For certain applications, such as strobing applications, it is necessary to start conversion at a specific point in time. This device will accommodate these applications. Although the on-chip sample-and-hold begins sampling upon the negative edge of the fourth I/O Clock cycle, the hold function is not initiated until the negative edge of the tenth I/O Clock cycle. Thus, the control circuitry can leave the I/O Clock signal in its high state during the tenth I/O Clock cycle until the moment at which the analog signal must be converted. The TLC1540/TLC1541 will continue sampling the analog input until the tenth falling edge of the I/O Clock. The control circuitry or software will then immediately lower the I/O Clock signal and hold the analog signal at the desired point in time and start conversion.

Detailed information on interfacing to most popular microprocessors is readily available from the factory.

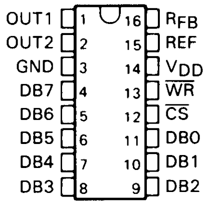
TLC7524
Advanced LinCMOS™ 8-BIT MULTIPLYING DIGITAL-TO-ANALOG CONVERTER

D3008, SEPTEMBER 1986—REVISED MARCH 1988

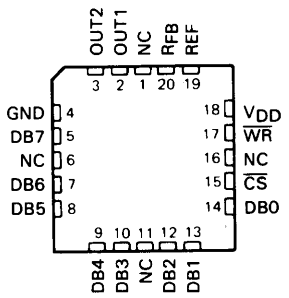
- Advanced LinCMOS™ Silicon-Gate Technology
- Easily Interfaced to Microprocessors
- On-Chip Data Latches
- Monotonic over the Entire A/D Conversion Range
- Segmented High-Order Bits Ensure Low-Glitch Output
- Designed to be Interchangeable with Analog Devices AD7524, PMI PM-7524, and Micro Power Systems MP7524
- Fast Control Signaling for Digital Signal Processor Applications Including Interface with TMS320

KEY PERFORMANCE SPECIFICATIONS	
Resolution	8 Bits
Linearity error	½ LSB Max
Power dissipation at V_{DD} = 5 V	5 mW Max
Settling time	100 ns Max
Propagation delay	80 ns Max

D OR N PACKAGE
(TOP VIEW)



FN PACKAGE
(TOP VIEW)



NC—No internal connection

description

The TLC7524 is an Advanced LinCMOS™ 8-bit digital-to-analog converter (DAC) designed for easy interface to most popular microprocessors.

The TLC7524 is an 8-bit multiplying DAC with input latches and with a load cycle similar to the "write" cycle of a random access memory. Segmenting the high-order bits minimizes glitches during changes in the most-significant bits, which produce the highest glitch impulse. The TLC7524 provides accuracy to ½ LSB without the need for thin-film resistors or laser trimming, while dissipating less than 5 milliwatts typically.

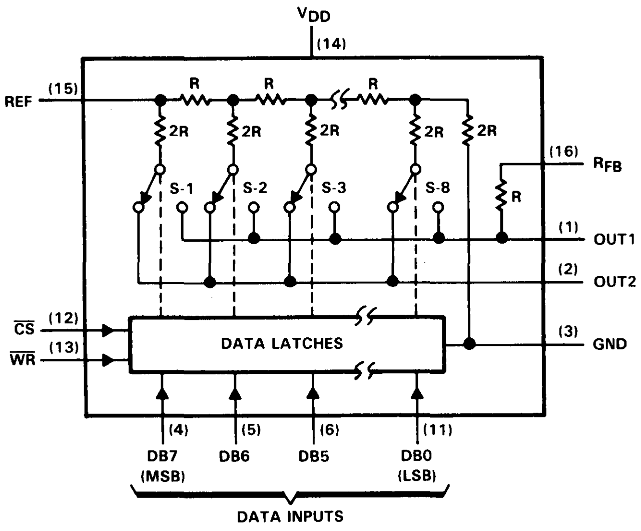
Featuring operation from a 5-V to 15-V single supply, the TLC7524 interfaces easily to most microprocessor buses or output ports. Excellent multiplying (2 or 4 quadrant) makes the TLC7524 an ideal choice for many microprocessor-controlled gain-setting and signal-control applications.

The TLC7524I is characterized for operation from −25°C to 85°C, and the TLC7524C is characterized for operation from 0°C to 70°C.

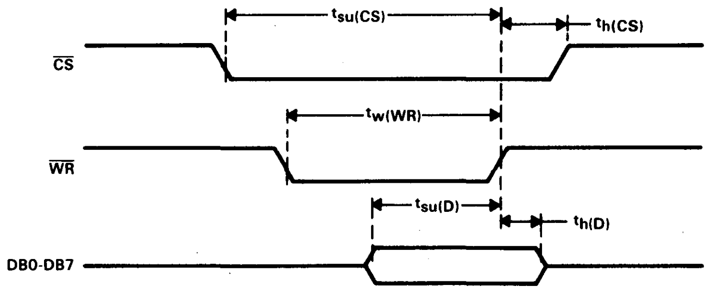
Advanced LinCMOS is a trademark of Texas Instruments Incorporated.

TEXAS INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

functional block diagram



operating sequence



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{DD}	−0.3 V to 16.5 V
Digital input voltage, V_I	−0.3 V to V_{DD}+0.3 V
Reference voltage, V_{ref}	±25 V
Peak digital input current, I_I	10 μA
Operating free-air temperature range: TLC7524I	−25°C to 85°C
TLC7524C	0°C to 70°C
Storage temperature range	−65°C to 150°C
Case temperature for 10 seconds: FN package	260°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds: D or N package	260°C

recommended operating conditions

		V_{DD} = 5 V			V_{DD} = 15 V			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{DD}		4.75	5	5.25	14.5	15	15.5	V
Reference voltage, V_{ref}			±10			±10		V
High-level input voltage, V_{IH}		2.4			13.5			V
Low-level input voltage, V_{IL}				0.8			1.5	V
CS setup time, $t_{su(CS)}$		40			40			ns
CS hold time, $t_{h(CS)}$		0			0			ns
Data bus input setup time, $t_{su(D)}$		25			25			ns
Data bus input hold time, $t_{h(D)}$		10			10			ns
Pulse duration, $\overline{WR}$ low, $t_{w(WR)}$		40			40			ns
Operating free-air temperature, T_A	TLC7524I	−25		85	−25		85	°C
	TLC7524C	0		70	0		70	

electrical characteristics over recommended operating free-air temperature range, V_{ref} = ±10 V, OUT1 and OUT2 at GND (unless otherwise noted)

PARAMETER			TEST CONDITIONS	V_{DD} = 5 V			V_{DD} = 15 V			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
I_{IH}	High-level input current		$V_I = V_{DD}$			10			10	μA
I_{IL}	Low-level input current		V_I = 0			−10			−10	μA
I_{lkg}	Output leakage current	OUT1	DB0-DB7 at 0 V, $\overline{WR}$, $\overline{CS}$ at 0 V, V_{ref} = ±10 V			±400			±200	nA
		OUT2	DB0-DB7 at V_{DD}, $\overline{WR}$, $\overline{CS}$ at 0 V, V_{ref} = ±10 V			±400			±200	
I_{DD}	Supply current	Quiescent	DB0-DB7 at V_{IH}min or V_{IL}max			1			2	mA
		Standby	DB0-DB7 at 0 V or V_{DD}			500			500	μA
k_{SVS}	Supply voltage sensitivity, Δgain/ΔV_{DD}		ΔV_{DD} = ±10%		0.01	0.16		0.005	0.04	%FSR/%
C_i	Input capacitance, DB0-DB7, $\overline{WR}$, $\overline{CS}$		V_I = 0			5			5	pF
C_o	Output capacitance	OUT1	DB0-DB7 at 0 V, $\overline{WR}$ and $\overline{CS}$ at 0 V			30			30	pF
		OUT2				120			120	
C_o	Output capacitance	OUT1	DB0-DB7 at V_{DD}, $\overline{WR}$ and $\overline{CS}$ at 0 V			120			120	pF
		OUT2				30			30	
	Reference input impedance (Pin 15 to GND)			5		20	5		20	kΩ

operating characteristics over recommended operating free-air temperature range, V_{ref} = ±10 V, OUT1 and OUT2 at GND (unless otherwise noted)

PARAMETER	TEST CONDITIONS	V_{DD} = 5 V MIN	V_{DD} = 5 V TYP†	V_{DD} = 5 V MAX	V_{DD} = 15 V MIN	V_{DD} = 15 V TYP†	V_{DD} = 15 V MAX	UNIT
Linearity error				±0.5			±0.5	LSB
Gain error	See Note 1			±2.5			±2.5	LSB
Settling time (to ½ LSB)	See Note 2			100			100	ns
Propagation delay from digital input to 90% of final analog output current	See Note 2			80			80	ns
Feedthrough at OUT1 or OUT2	V_{ref} = ±10 V (100-kHz sinewave) $\overline{WR}$ and $\overline{CS}$ at 0 V, DB0-DB7 at 0 V			0.5			0.5	%FSR
Temperature coefficient of gain	T_A = 25°C to MAX		±0.004			±0.001		%FSR/°C

†Typical values at T_A = 25°C.
NOTES: 1. Gain error is measured using the internal feedback resistor. Nominal Full Scale Range (FSR) = V_{ref} − 1 LSB.
2. OUT1 load = 100 Ω, C_{ext} = 13 pF, $\overline{WR}$ at 0 V, $\overline{CS}$ at 0 V, DB0-DB7 at 0 V to V_{DD} or V_{DD} to 0 V.

principles of operation

The TLC7524 is an 8-bit multiplying D/A converter consisting of an inverted R-2R ladder, analog switches, and data input latches. Binary weighted currents are switched between the OUT1 and OUT2 bus lines, thus maintaining a constant current in each ladder leg independent of the switch state. The high-order bits are decoded and these decoded bits, through a modification in the R-2R ladder, control three equally weighted current sources. Most applications only require the addition of an external operational amplifier and a voltage reference.

The equivalent circuit for all digital inputs low is seen in Figure 1. With all digital inputs low, the entire reference current, I_{ref}, is switched to OUT2. The current source I/256 represents the constant current flowing through the termination resistor of the R-2R ladder, while the current source I_{lkg} represents leakage currents to the substrate. The capacitances appearing at OUT1 and OUT2 are dependent upon the digital input code. With all digital inputs high, the off-state switch capacitance (30 pF maximum) appears at OUT2 and the on-state switch capacitance (120 pF maximum) appears at OUT1. With all digital inputs low, the situation is reversed as shown in Figure 1. Analysis of the circuit for all digital inputs high is similar to Figure 1; however, in this case, I_{ref} would be switched to OUT1.

Interfacing the TLC7524 D/A converter to a microprocessor is accomplished via the data bus and the $\overline{CS}$ and $\overline{WR}$ control signals. When $\overline{CS}$ and $\overline{WR}$ are both low, the TLC7524 analog output responds to the data activity on the DB0-DB7 data bus inputs. In this mode, the input latches are transparent and input data directly affects the analog output. When either the $\overline{CS}$ signal or $\overline{WR}$ signal goes high, the data on the DB0-DB7 inputs are latched until the $\overline{CS}$ and $\overline{WR}$ signals go low again. When $\overline{CS}$ is high, the data inputs are disabled regardless of the state of the $\overline{WR}$ signal.

The TLC7524 is capable of performing 2-quadrant or full 4-quadrant multiplication. Circuit configurations for 2-quadrant or 4-quadrant multiplication are shown in Figures 2 and 3. Input coding for unipolar and bipolar operation are summarized in Tables 1 and 2, respectively.

principles of operation (continued)

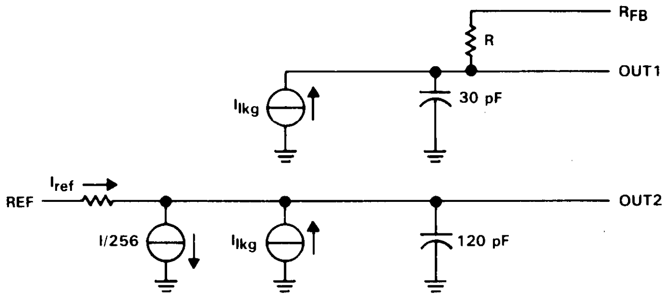


FIGURE 1. TLC7524 EQUIVALENT CIRCUIT WITH ALL DIGITAL INPUTS LOW

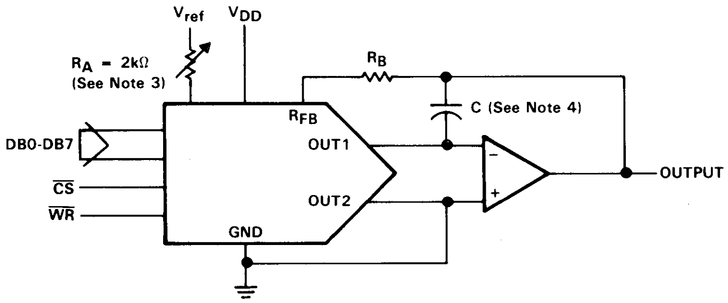


FIGURE 2. UNIPOLAR OPERATION (2-QUADRANT MULTIPLICATION)

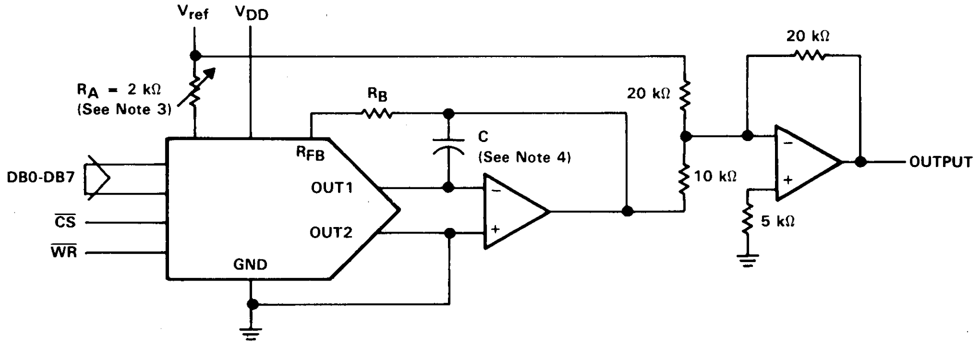


FIGURE 3. BIPOLAR OPERATION (4-QUADRANT OPERATION)

NOTES: 3. R_A and R_B used only if gain adjustment is required.
4. C phase compensation (10-15 pF) is required when using high-speed amplifiers to prevent ringing or oscillation.

principles of operation (continued)

TABLE 1. UNIPOLAR BINARY CODE

DIGITAL INPUT (SEE NOTE 5) MSB LSB	ANALOG OUTPUT
11111111	$-V_{ref}$ (255/256)
10000001	$-V_{ref}$ (129/256)
10000000	$-V_{ref}$ (128/256) = $-V_{ref}/2$
01111111	$-V_{ref}$ (127/256)
00000001	$-V_{ref}$ (1/256)
00000000	0

TABLE 2. BIPOLAR (OFFSET BINARY) CODE

DIGITAL INPUT (SEE NOTE 6) MSB LSB	ANALOG OUTPUT
11111111	V_{ref} (127/128)
10000001	V_{ref} (1/128)
10000000	0
01111111	$-V_{ref}$ (1/128)
00000001	$-V_{ref}$ (127/128)
00000000	$-V_{ref}$

NOTES: 5. LSB = 1/256 (V_{ref}).
6. LSB = 1/128 (V_{ref}).

microprocessor interfaces

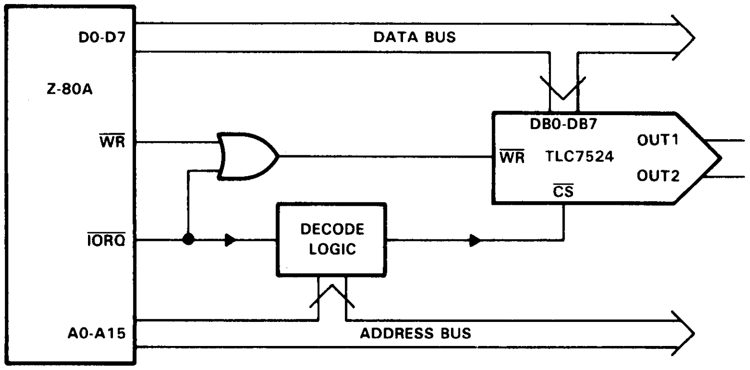


FIGURE 4. TLC7524—Z-80A INTERFACE

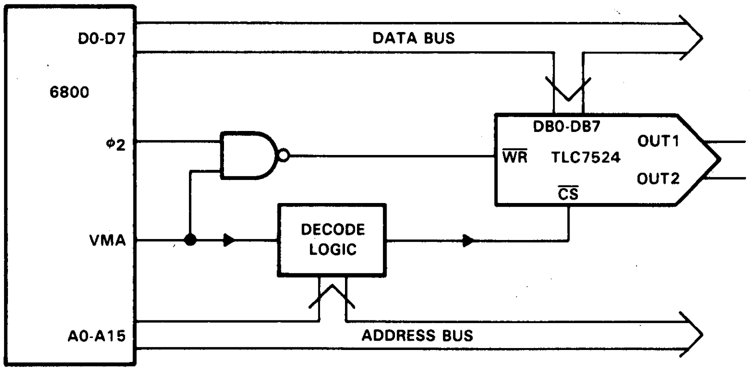


FIGURE 5. TLC7524—6800 INTERFACE

microprocessor interfaces (continued)

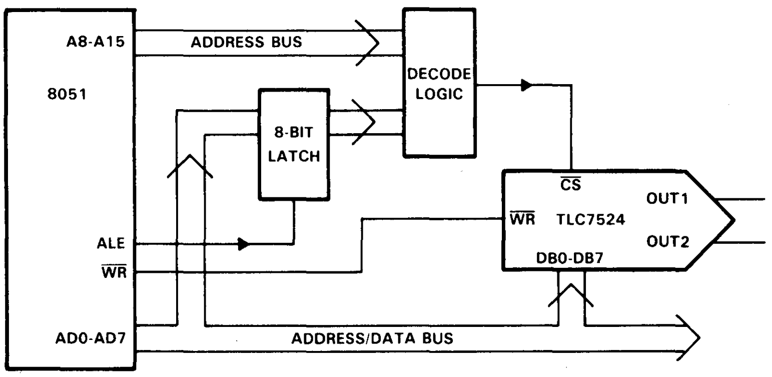


FIGURE 6. TLC7524—8051 INTERFACE

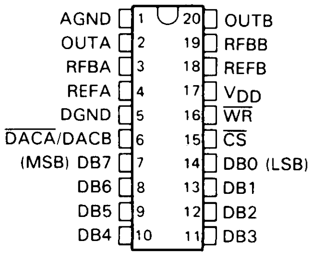
TLC7528
Advanced LinCMOS™ DUAL 8-BIT MULTIPLYING DIGITAL-TO-ANALOG CONVERTER

D2979, JANUARY 1987 – REVISED JULY 1988

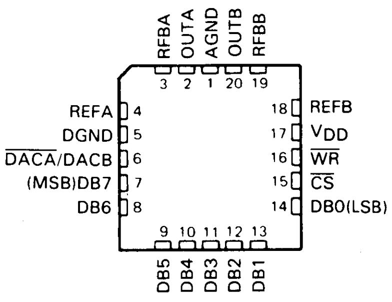
- **Advanced LinCMOS™ Silicon-Gate Technology**
- **Easily Interfaced to Microprocessors**
- **On-Chip Data Latches**
- **Monotonic Over the Entire A/D Conversion Range**
- **Designed to be Interchangeable with Analog Devices AD7528 and PMI PM-7528**
- **Fast Control Signaling for Digital Signal Processor Applications Including Interface with TMS320**

KEY PERFORMANCE SPECIFICATIONS	
Resolution	8 bits
Linearity Error	1/2 LSB
Power Dissipation at V_{DD} = 5 V	5 mW
Settling Time at V_{DD} = 5 V	100 ns
Propagation Delay at V_{DD} = 5 V	80 ns

DW OR N PACKAGE
(TOP VIEW)



FN PACKAGE
(TOP VIEW)



description

The TLC7528 is a dual 8-bit digital-to-analog converter designed with separate on-chip data latches and featuring excellent DAC-to-DAC matching. Data is transferred to either of the two DAC data latches via a common 8-bit input port. Control input $\overline{\text{DACA}}$/DACB determines which DAC is to be loaded. The "load" cycle of the TLC7528 is similar to the "write" cycle of a random-access memory, allowing easy interface to most popular microprocessor busses and output ports. Segmenting the high-order bits minimizes glitches during changes in the most significant bits, where glitch impulse is typically the strongest.

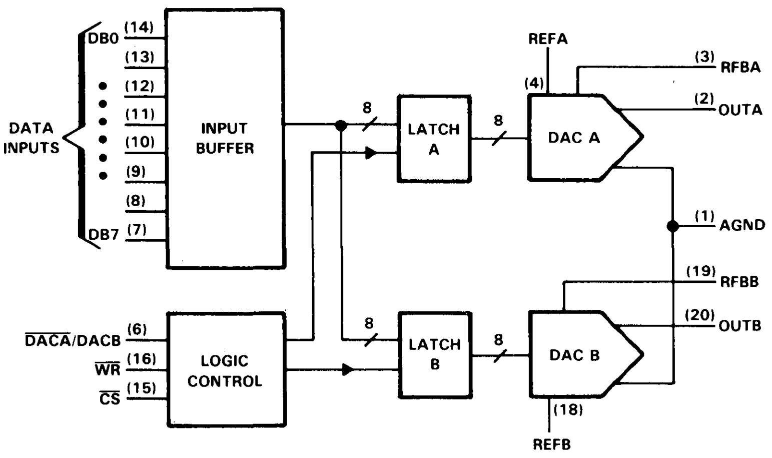
The TLC7528 operates from a 5-V to 15-V power supply and dissipates less than 15 mW (typical). Excellent 2- or 4-quadrant multiplying makes the TLC7528 a sound choice for many microprocessor-controlled gain-setting and signal-control applications.

The TLC7528I is characterized for operation from −25 to 85°C. The TLC7528C is characterized for operation from 0°C to 70°C.

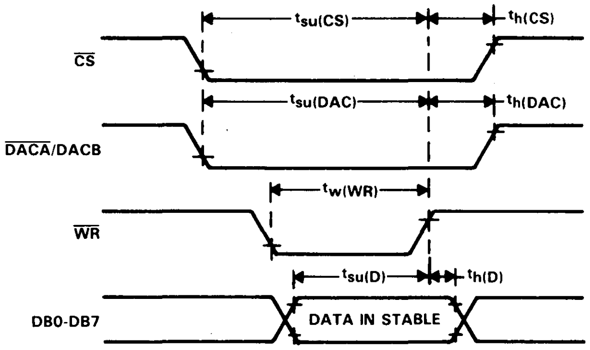
Advanced LinCMOS is a trademark of Texas Instruments Incorporated.

TEXAS INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

functional block diagram



operating sequence



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{DD} (to AGND or DGND)	−0.3 V to 16.5 V
Voltage between AGND and DGND	$\pm V_{DD}$
Input voltage, V_I (to DGND)	−0.3 V to V_{DD} + 0.3
Reference voltage, V_{refA} or V_{refB} (to AGND)	±25 V
Feedback voltage V_{RFBA} or V_{RFBB} (to AGND)	±25 V
Output voltage, V_{OA} or V_{OB} (to AGND)	±25 V
Peak input current	10 μA
Operating free-air temperature range: TLC7528I	−25°C to 85°C
TLC7528C	0°C to 70°C
Storage temperature range	−65°C to 150°C
Case temperature for 10 seconds: FN package	260°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds: DW or N package	260°C

recommended operating conditions

		V_{DD} = 4.75 V to 5.25 V			V_{DD} = 14.5 V to 15.5 V			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Reference voltage, V_{refA} or V_{refB}			±10			±10		V
High-level input voltage, V_{IH}		2.4			13.5			V
Low-level input voltage, V_{IL}				0.8			1.5	V
$\overline{CS}$ setup time, $t_{su(CS)}$		50			50			ns
$\overline{CS}$ hold time, $t_{h(CS)}$		0			0			ns
DAC select setup time, $t_{su(DAC)}$		50			50			ns
DAC select hold time, $t_{h(DAC)}$		10			10			ns
Data bus input setup time $t_{su(D)}$		25			25			ns
Data bus input hold time $t_{h(D)}$		0			0			ns
Pulse duration, $\overline{WR}$ low, $t_{w(WR)}$		50			50			ns
Operating free-air temperature, T_A	TLC7528I	−25		85	−25		85	°C
	TLC7528C	0		70	0		70	°C

electrical characteristics over recommended operating free-air temperature range, V_{refA} = V_{refB} = 10 V, V_{OA} and V_{OB} at 0 V (unless otherwise noted)

PARAMETER			TEST CONDITIONS	V_{DD} = 5 V MIN	V_{DD} = 5 V TYP†	V_{DD} = 5 V MAX	V_{DD} = 15 V MIN	V_{DD} = 15 V TYP†	V_{DD} = 15 V MAX	UNIT
I_{IH}	High-level input current		V_I = V_{DD}			10			10	μA
I_{IL}	Low-level input current		V_I = 0 V			−10			−10	μA
	Reference input impedance (Pin 15 to GND)			5	12	20	5	12	20	kΩ
I_{lkg}	Output leakage current	OUTA	DACA data latch loaded with 00000000, V_{refA} = ±10 V			±400			±200	nA
		OUTB	DACB data latch loaded with 00000000, V_{refB} = ±10 V			±400			±200	
	Input resistance match (REFA to REFB)					±1%			±1%	
	DC supply sensitivity, Δ gain/Δ V_{DD}		ΔV_{DD} = ±10%			0.04			0.02	%/%
I_{DD}	Supply current (quiescent)		DB0-DB7 at V_{IH}min or V_{IL}max			1			1	mA
I_{DD}	Supply current (standby)		DB0-DB7 at 0 V or V_{DD}			0.5			0.5	mA
C_i	Input capacitance	DB0-DB7				10			10	pF
		$\overline{WR}$, $\overline{CS}$ $\overline{DACA}$/DACB				15			15	
C_o	Output capacitance, (OUTA, OUTB)		DAC data latches loaded with 00000000			50			50	pF
			DAC data latches loaded with 11111111			120			120	

†All typical values are at T_A = 25°C.

operating characteristics over recommended operating free-air temperature range, $V_{refA} = V_{refB} = 10$ V, V_{OA} and V_{OB} at 0 V (unless otherwise noted)

PARAMETER		TEST CONDITIONS	$V_{DD} = 5$ V MIN	$V_{DD} = 5$ V TYP	$V_{DD} = 5$ V MAX	$V_{DD} = 15$ V MIN	$V_{DD} = 15$ V TYP	$V_{DD} = 15$ V MAX	UNIT
Linearity error					± 1/2			± 1/2	LSB
Settling time (to 1/2 LSB)		See Note 1			100			100	ns
Gain error		See Note 2			2.5			2.5	LSB
AC feedthrough	REFA to OUTA	See Note 3			−65			−65	dB
	REFB to OUTB				−65			−65	
Temperature coefficient of gain		See Note 4			0.007			0.0035	%FSR/°C
Propagation delay (from digital input to 90% of final analog output current)		See Note 5			80			80	ns
Channel-to-channel isolation	REFA to OUTB	See Note 6		77			77		dB
	REFB to OUTA	See Note 7		77			77		
Digital-to-analog glitch impulse area		Measured for code transition from 00000000 to 11111111, $T_A = 25$°C		160			440		nVs
Digital crosstalk glitch impulse area		Measured for code transition from 00000000 to 11111111, $T_A = 25$°C		30			60		nVs
Harmonic distortion		$V_i = 6$ V rms, f = 1 kHz, $T_A = 25$°C		−85			−85		dB

NOTES:
1. OUTA, OUTB load = 100 Ω, C_{ext} = 13 pF; $\overline{WR}$ and $\overline{CS}$ at 0 V; DB0-DB7 at 0 V to V_{DD} or V_{DD} to 0 V.
2. Gain error is measured using an internal feedback resistor. Nominal Full Scale Range (FSR) = V_{ref} − 1 LSB.
3. V_{ref} = 20 V peak-to-peak, 100-kHz sine wave; DAC data latches loaded with 00000000.
4. Temperature coefficient of gain measured from 0°C to 25°C or from 25°C to 70°C.
5. $V_{refA} = V_{refB}$ = 10 V; OUTA/OUTB load = 100 Ω, C_{ext} = 13 pF; $\overline{WR}$ and $\overline{CS}$ at 0 V; DB0-DB7 at 0 V to V_{DD} or V_{DD} to 0 V.
6. Both DAC latches loaded with 11111111; V_{refA} = 20 V peak-to-peak, 100-kHz sine wave; V_{refB} = 0; T_A = 25°C.
7. Both DAC latches loaded with 11111111; V_{refB} = 20 V peak-to-peak, 100-kHz sine wave; V_{refA} = 0; T_A = 25°C.

principles of operation

The TLC7528 contains two identical 8-bit multiplying D/A converters, DACA and DACB. Each DAC consists of an inverted R-2R ladder, analog switches, and input data latches. Binary-weighted currents are switched between DAC output and AGND, thus maintaining a constant current in each ladder leg independent of the switch state. Most applications require only the addition of an external operational amplifier and voltage reference. A simplified D/A circuit for DACA with all digital inputs low is shown in Figure 1.

Figure 2 shows the DACA equivalent circuit. A similar equivalent circuit can be drawn for DACB. Both DACs share the analog ground pin 1 (AGND). With all digital inputs high, the entire reference current flows to OUTA. A small leakage current (I_{lkg}) flows across internal junctions, and as with most semiconductor devices, doubles every 10°C. C_o is due to the parallel combination of the NMOS switches and has a value that depends on the number of switches connected to the output. The range of C_o is 50 pF to 120 pF maximum. The equivalent output resistance r_o varies with the input code from 0.8R to 3R where R is the nominal value of the ladder resistor in the R-2R network.

Interfacing the TLC7528 to a microprocessor is accomplished via the data bus, $\overline{CS}$, $\overline{WR}$, and $\overline{DACA}$/DACB control signals. When $\overline{CS}$ and $\overline{WR}$ are both low, the TLC7528 analog output, specified by the $\overline{DACA}$/DACB control line, responds to the activity on the DB0-DB7 data bus inputs. In this mode, the input latches are transparent and input data directly affects the analog output. When either the $\overline{CS}$ signal or $\overline{WR}$ signal goes high, the data on the DB0-DB7 inputs is latched until the $\overline{CS}$ and $\overline{WR}$ signals go low again. When $\overline{CS}$ is high, the data inputs are disabled regardless of the state of the $\overline{WR}$ signal.

The digital inputs of the TLC7528 provide TTL compatibility when operated from a supply voltage of 5 V. The TLC7528 may be operated with any supply voltage in the range from 5 V to 15 V, however, input logic levels are not TTL compatible above 5 V.

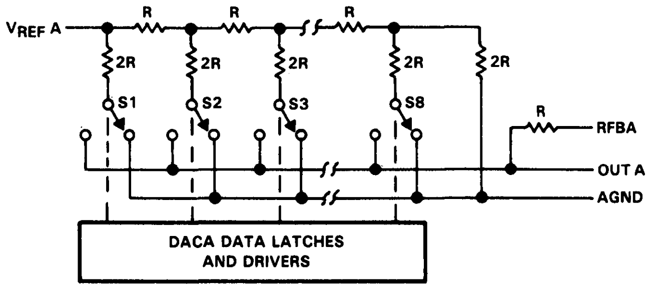


FIGURE 1. SIMPLIFIED FUNCTIONAL CIRCUIT FOR DACA

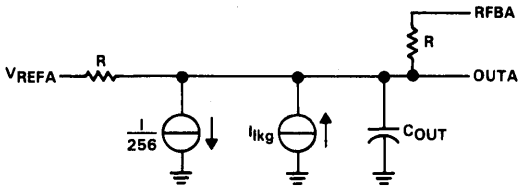


FIGURE 2. TLC7528 EQUIVALENT CIRCUIT, DACA LATCH LOADED WITH 11111111.

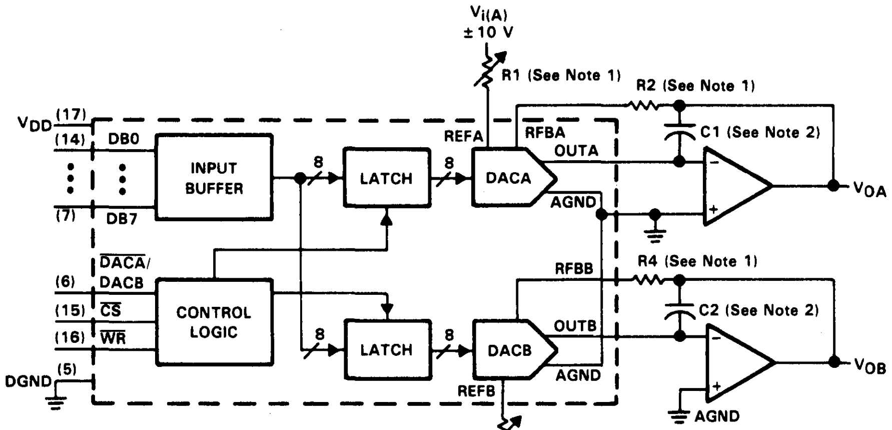
MODE SELECTION TABLE

$\overline{\text{DACA}}$/ DACB	$\overline{\text{CS}}$	$\overline{\text{WR}}$	DACA	DACB
L	L	L	WRITE	HOLD
H	L	L	HOLD	WRITE
X	H	X	HOLD	HOLD
X	X	H	HOLD	HOLD

L = low level, H = high level, X = don't care

TYPICAL APPLICATION DATA

The TLC7528 is capable of performing 2-quadrant or full 4-quadrant multiplication. Circuit configurations for 2-quadrant and 4-quadrant multiplication are shown in Figures 3 and 4. Input coding for unipolar and bipolar operation are summarized in Tables 1 and 2, respectively.

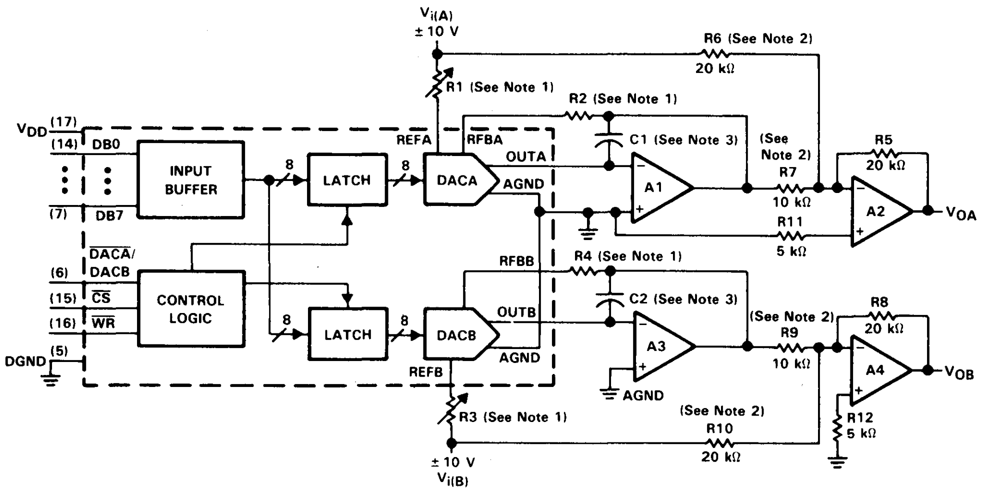


RECOMMENDED TRIM RESISTOR VALUES	
R1, R3	500 Ω
R2, R4	150 Ω

NOTES: 1. R1, R2, R3, and R4 are used only if gain adjustment is required. See table for recommended values. Make gain adjustment with digital input of 255.
2. C1 and C2 phase compensation capacitors (10 pF to 15 pF) are required when using high-speed amplifiers to prevent ringing or oscillation.

FIGURE 3. UNIPOLAR OPERATION (2-QUADRANT MULTIPLICATION)

TYPICAL APPLICATION DATA



NOTES: 1. R1, R2, R3, and R4 are used only if gain adjustment is required. See table in Figure 3 for recommended values. Adjust R1 for V_{OA} = 0 V with code 10000000 in DACA latch. Adjust R3 for V_{OB} = 0 V with 10000000 in DACB latch.
2. Matching and tracking are essential for resistor pairs R6, R7, R9, and R10.
3. C1 and C2 phase compensation capacitors (10 pF to 15 pF) may be required if A1 and A3 are high-speed amplifiers.

FIGURE 4. BIPOLAR OPERATION (4-QUADRANT OPERATION)

TABLE 1. UNIPOLAR BINARY CODE

DAC LATCH CONTENTS MSB LSB†	ANALOG OUTPUT
11111111	$-V_i$ (255/256)
10000001	$-V_i$ (129/256)
10000000	$-V_i$ (128/256) = $-V_i/2$
01111111	$-V_i$ (127/256)
00000001	$-V_i$ (1/256)
00000000	$-V_i$ (0/256) = 0

† 1 LSB = $(2^{-8})V_i$

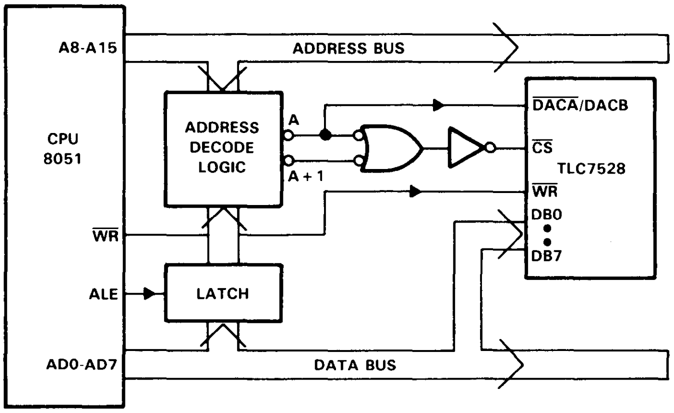
TABLE 2. BIPOLAR (OFFSET BINARY) CODE

DAC LATCH CONTENTS MSB LSB‡	ANALOG OUTPUT
11111111	V_i (127/128)
10000001	V_i (1/128)
10000000	0 V
01111111	$-V_i$ (1/128)
00000001	$-V_i$ (127/128)
00000000	$-V_i$ (128/128)

‡ 1 LSB = $(2^{-7})V_i$

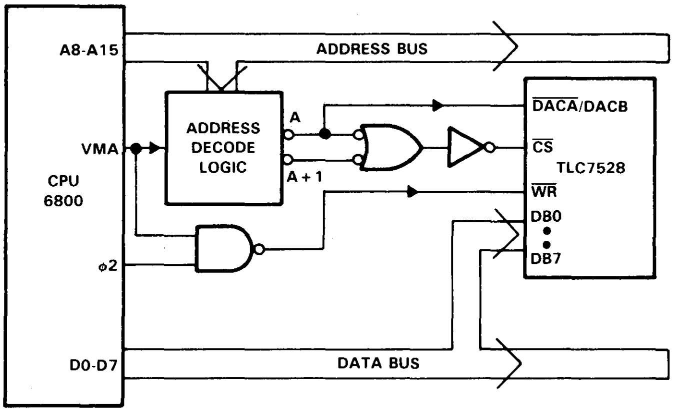
TYPICAL APPLICATION DATA

microprocessor interface information



NOTE: A = decoded address for TLC7528 DACA.
A + 1 = decoded address for TLC7528 DACB.

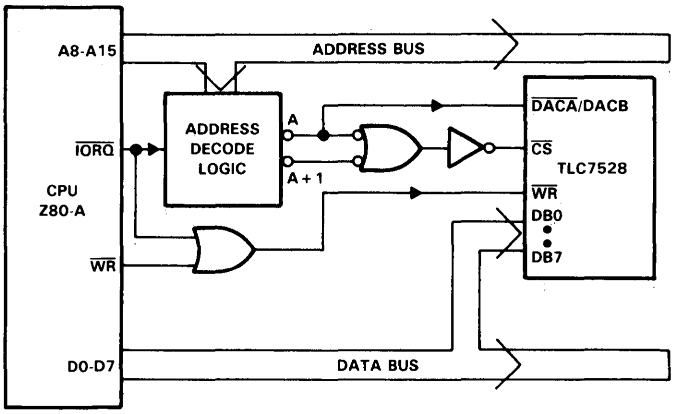
FIGURE 5. TLC7528 — INTEL 8051 INTERFACE



NOTE: A = decoded address for TLC7528 DACA.
A + 1 = decoded address for TLC7528 DACB.

FIGURE 6. TLC7528 — 6800 INTERFACE

TYPICAL APPLICATION DATA



NOTE: A = decoded address for TLC7528 DACA.
A + 1 = decoded address for TLC7528 DACB.

FIGURE 7. TLC7528 TO Z80-A INTERFACE

programmable window detector

The programmable window comparator shown in Figure 8 will determine if voltage applied to the DAC feedback resistors are within the limits programmed into the TLC7528 data latches. Input signal range depends on the reference and polarity, that is, the test input range is 0 to $-V_{ref}$. The DACA and DACB data latches are programmed with the upper and lower test limits. A signal within the programmed limits will drive the output high.

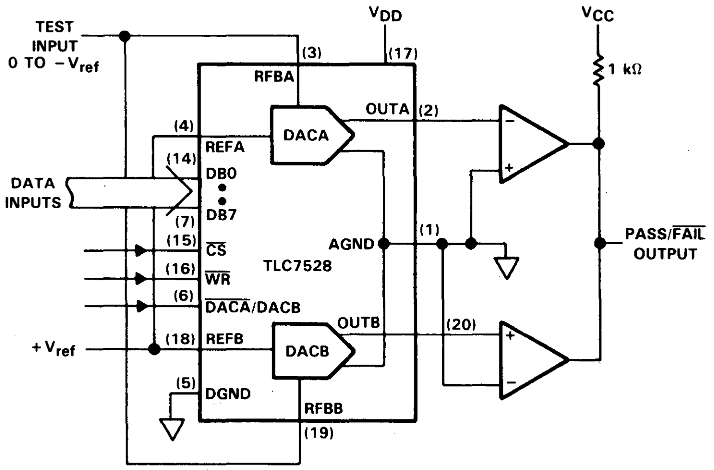


FIGURE 8. DIGITALLY PROGRAMMABLE WINDOW COMPARATOR (UPPER- AND LOWER-LIMIT TESTER)

TYPICAL APPLICATION DATA

digitally controlled signal attenuator

Figure 9 shows the TLC7528 configured as a two-channel programmable attenuator. Applications include stereo audio and telephone signal level control. Table 3 shows input codes vs attenuation for a 0 to 15.5 dB range.

Attentuation db = $-20 \log_{10}$ D/256, D = digital input code

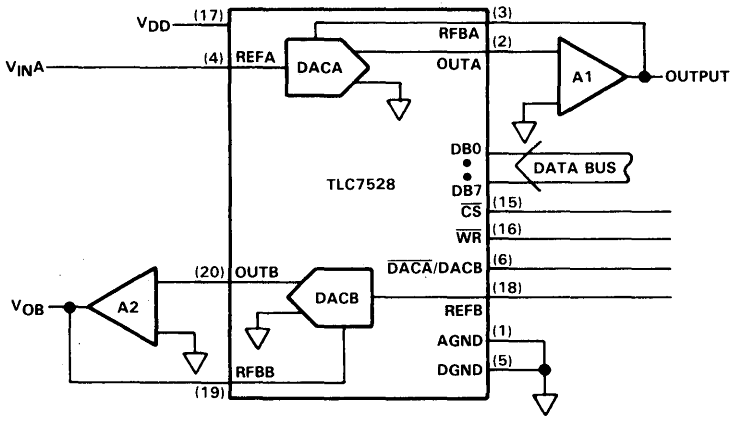


FIGURE 9. DIGITALLY CONTROLLED DUAL TELEPHONE ATTENUATOR

TABLE 3. ATTENUATION vs DACA, DACB CODE

ATTN(dB)	DAC INPUT CODE	CODE IN DECIMAL	ATTN(dB)	DAC INPUT CODE	CODE IN DECIMAL
0	11111111	255	8.0	01100110	102
0.5	11110010	242	8.5	01100000	96
1.0	11100100	228	9.0	01011011	91
1.5	11010111	215	9.5	01010110	86
2.0	11001011	203	10.0	01010001	81
2.5	11000000	192	10.5	01001100	76
3.0	10110101	181	11.0	01001000	72
3.5	10101011	171	11.5	01000100	68
4.0	10100010	162	12.0	01000000	64
4.5	10011000	152	12.5	00111101	61
5.0	10010000	144	13.0	00111001	57
5.5	10001000	136	13.5	00110110	54
6.0	10000000	128	14.0	00110011	51
6.5	01111001	121	14.5	00110000	48
7.0	01110010	114	15.0	00101110	46
7.5	01101100	108	15.5	00101011	43

TYPICAL APPLICATION DATA

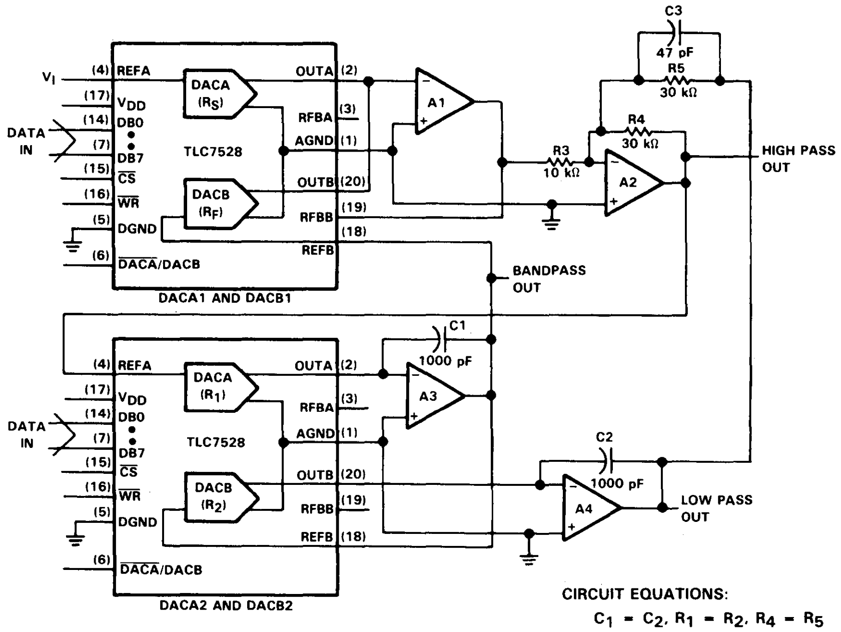
programmable state-variable filter

This programmable state-variable or universal filter configuration provides low-pass, high-pass, and band-pass outputs, and is suitable for applications in which microprocessor control of filter parameters is required.

As shown in Figure 10, DACA1 and DACB1 control the gain and Q of the filter while DACA2 and DACB2 control the cutoff frequency. Both halves of the DACA2 and DACB2 must track accurately in order for the cutoff-frequency equation to be true. With the TLC7528, this is easily achieved.

$$f_C = \frac{1}{2\pi\, R1\, C1}$$

The programmable range for the cutoff or center frequency is 0 to 15 kHz with a Q ranging from 0.3 to 4.5. This defines the limits of the component values.



CIRCUIT EQUATIONS:

$$C_1 = C_2,\ R_1 = R_2,\ R_4 = R_5$$

$$Q = \frac{R_3}{R_4} \cdot \frac{R_F}{R_{fb(DACB1)}}$$

$$A_o = -\frac{R_F}{R_S}$$

NOTES: A. Op-amps A1, A2, A3, and A4 are TL287.
B. C3 compensates for the op-amp gain-bandwidth limitations.
C. DAC equivalent resistance equals $\frac{256 \times \text{(DAC ladder resistance)}}{\text{DAC digital code}}$

FIGURE 10. DIGITALLY CONTROLLED STATE-VARIABLE FILTER

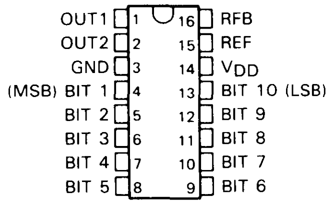
AD7533, TLC7533
Advanced LinCMOS™ DUAL 10-BIT MULTIPLYING DIGITAL-TO-ANALOG CONVERTERS

D2166, OCTOBER 1986—REVISED JULY 1988

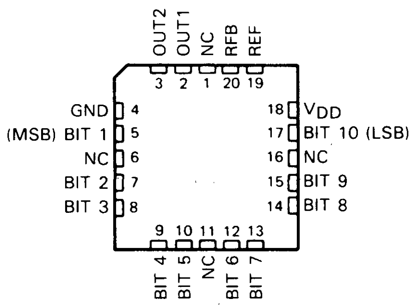
- **Advanced LinCMOS™ Silicon-Gate Technology**
- **Monotonic Over the Entire A/D Conversion Range**
- **Fast Settling Time**
- **CMOS/TTL Compatible**
- **Four-Quadrant Multiplication**
- **Designed to be Interchangeable with Analog Devices AD7533, AD7520, and PMI PM-7533**

KEY PERFORMANCE SPECIFICATIONS	
Resolution	10 Bits
Linearity Error	1/2 LSB
Power Dissipation	30 mW
Settling Time	150 ns

AD7533 . . . N PACKAGE
TLC7533 . . . D OR N PACKAGE
(TOP VIEW)



FN CHIP CARRIER PACKAGE
(TOP VIEW)



NC—No internal connections

description

The AD7533 and TLC7533 are Advanced LinCMOS™ 10-bit digital-to-analog converters featuring two- and four-quadrant multiplication.

The AD7533 and TLC7533 are functionally equivalent to the AD7520 and have the same pinout. Texas Instruments advanced thin-film-on-monolithic-CMOS fabrication process provides 10-bit linearity without laser trimming.

The AD7533 and TLC7533 feature TTL or CMOS compatibility with low input leakage currents from 5-V to 15-V power supplies. Output scaling is provided by an internal feedback resistor and an external operational amplifier. Either positive or negative reference voltages can be used.

The AD7533C and TLC7533I are characterized for operation from −25°C to 85°C. The AD7533L and TLC7533C are characterized for operation from 0°C to 70°C.

AVAILABLE OPTIONS

SYMBOLIZATION[†]		OPERATING TEMPERATURE RANGE
DEVICE	PACKAGE SUFFIX	
AD7533C	N	−25°C to 85°C
AD7533L	FN, N	0°C to 70°C
TLC7533C	D, FN, N	0°C to 70°C
TLC7533I	D, N	−25°C to 85°C

[†]In many instances, these ICs may have both AD7533 and TLC7533 symbolization on the package.

ADVANCE INFORMATION

Advanced LinCMOS is a trademark of Texas Instruments Incorporated.

TEXAS INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{DD} (see Note 1)	−0.3 V to 17 V
Digital input voltage, V_I	−0.3 to V_{DD}+0.3 V
Output voltage at T_A = 25°C, OUT1 and OUT2	±25 V
R_{FB} to ground at T_A = 25°C	−0.3 V to V_{DD}
Reference voltage, V_{ref}	±25 V
Operating free-air temperature range: AD7533C, TLC7533I	−25°C to 85°C
AD7533L, TLC7533C	0°C to 70°C
Storage temperature range	−65°C to 150°C
Case temperature for 10 seconds: FN package	260°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds: D or N package	260°C

NOTE 1: All voltage values are with respect to the network ground terminal.

recommended operating conditions

		MIN	NOM	MAX	UNIT
Supply voltage, V_{DD}		5		16.5	V
Reference voltage, V_{ref}			±10		V
High-level input voltage, V_{IH}		2.4			V
Low-level input voltage, V_{IL}				0.8	V
Operating free-air temperature, T_A	AD7533C, TLC7533I	−25		85	°C
	AD7533L, TLC7533C	0		70	°C

electrical characteristics over recommended operating temperature range, V_{DD} = 15 V, V_{ref} = ±10 V, OUT1 and OUT2 at 0 V (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	MAX	UNIT
I_{Ilkg}	Input leakage current		V_I = 0 or V_{DD}			±1	μA
r_i	Input resistance, REF (see Note 2)				5	20	kΩ
I_{Olkg}	Output leakage current	OUT1	Digital inputs at V_{IL}	Full range		±200	nA
				25°C		±50	
		OUT2	Digital inputs at V_{IH}	Full range		±200	
				25°C		±50	
k_{svs}	Supply voltage sensitivity $\Delta A_V/\Delta V_{DD}$ (see Note 3)		V_{DD} = 14 V to 17 V, Digital inputs at V_{IH} or V_{IL}	Full range		0.008	%/%
				25°C		0.005	
I_{DD}	Supply current					2	mA
C_i	Input capacitance		V_I = 0 or V_{DD}			5	pF
C_o	Output capacitance	OUT1	Digital inputs at V_{IH}			100	pF
		OUT2				35	
		OUT1	Digital inputs at V_{IL}			35	
		OUT2				100	

NOTES: 2. Temperature coefficient is approximately −300 ppm/°C.
3. A_V is the ratio of the D/A external operational amplifier output voltage to the REF input voltage when using the internal feedback resistor.

operating characteristics over recommended operating free-air temperature range, V_{DD} = 15 V, V_{ref} = 10 V, OUT1 and OUT2 at 0 V (unless otherwise noted)

PARAMETER	TEST CONDITIONS		MIN	MAX	UNIT
Relative accuracy	See Note 4			±0.05	%FSR
Gain error	Digital inputs = V_{IH}, See Notes 4 and 5	Full range		±1.5	%FS
		25°C		±1.4	
Output current settling time	To ±0.05% FSR, R_L = 100 Ω Digital inputs changing from V_{IH} to V_{IL}, or V_{IL} to V_{IH}			150	ns
Feedthrough error	Digital inputs at V_{IL}, V_{ref} = ±10 V sine wave at 100 kHz			±0.1	%FSR

NOTES 4. Practical Full Scale Range (FSR) = V_{ref} − 1 LSB.
5. Gain error is measured using an internal feedback resistor, Full Scale (FS) = V_{ref} (1023/1024). Maximum gain change from T_A = 25°C to minimum or maximum temperature is ±0.1% FSR.

ADVANCE INFORMATION

PRINCIPLES OF OPERATION

The AD7533 and TLC7533 are 10-bit multiplying D/A converters consisting of an inverted R-2R ladder and analog switches. Binary-weighted currents are switched between the OUT1 and OUT2 bus lines by NMOS current switches. The on-state resistances of these switches are binarily scaled so that the voltage drop across every switch is the same. The OUT1 and OUT2 bus lines should be maintained at the same potential so that the current in each ladder leg remains constant and is independent of the switch state. Most applications require only the addition of an external operational amplifier and a voltage reference.

The equivalent circuit for all digital inputs low is shown in Figure 1. With all of the digital inputs low, the entire reference current, I_{ref}, is switched to OUT2 as shown in Figure 2. The current source $I_{ref}/1024$ represents the constant current flowing through the termination resistor of the R-2R ladder; while the current source I_{lkg} represents leakage currents to the substrate. The output capacitances, $C_{o(1)}$ and $C_{o(2)}$, are due to the capacitance of the NMOS current switches and vary with the switch state. With all digital inputs low, all of the current switches and the entire resistor ladder are switched to the OUT2 bus line. The capacitance appearing at OUT2 is a maximum of 100 pF; at OUT1 there is a maximum of 35 pF. With all digital inputs high, all of the current switches are switched to OUT1, and 100 pF maximum appears at OUT1. A maximum of 35 pF appears at OUT2 as shown in Figure 3.

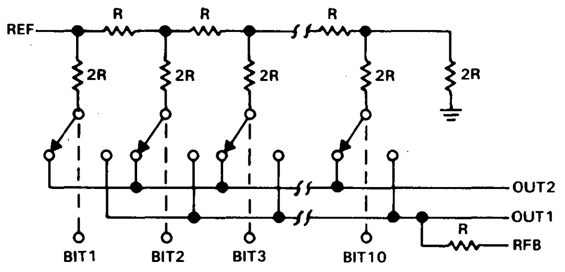
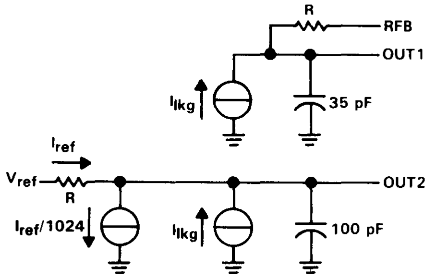


FIGURE 1. SIMPLIFIED D/A CIRCUIT — ALL DIGITAL INPUTS LOW



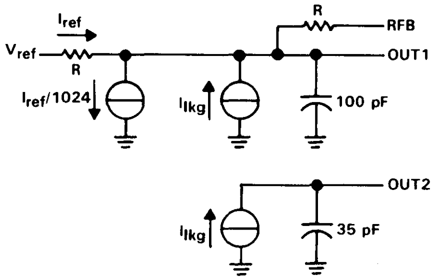


FIGURE 2. D/A EQUIVALENT CIRCUIT — ALL DIGITAL INPUTS LOW

FIGURE 3. D/A EQUIVALENT CIRCUIT — ALL DIGITAL INPUTS HIGH

TYPICAL APPLICATION DATA

The AD7533 and TLC7533 are capable of performing 2-quadrant or full 4-quadrant multiplication. Circuit configurations for 2-quadrant or 4-quadrant multiplication are shown in Figures 4 and 5. Input coding for unipolar and bipolar operation are summarized in Tables 1 and 2, respectively.

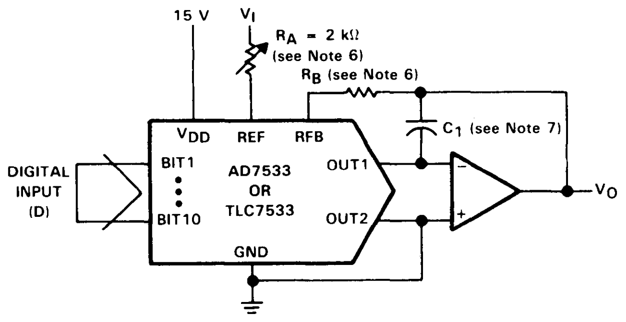


FIGURE 4. UNIPOLAR OPERATION (2-QUADRANT MULTIPLICATION)

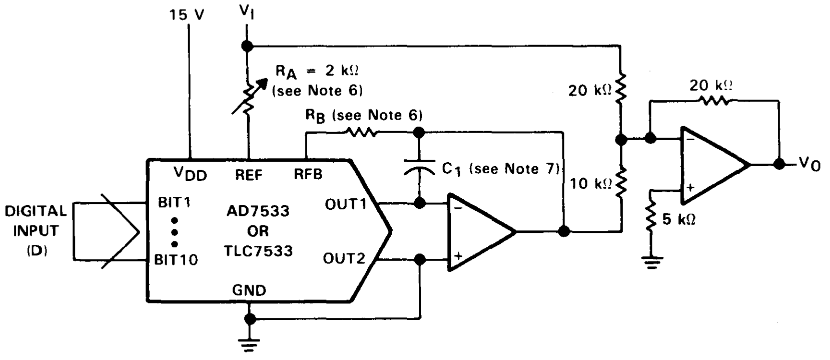


FIGURE 5. BIPOLAR OPERATION (4-QUADRANT OPERATION)

NOTES: 6. R_A and R_B are used only if gain adjustment is required.
7. C_1 (10-33 pF) may be required for phase compensation when using high-speed op-amps.

TABLE 1. UNIPOLAR BINARY CODE

DAC DIGITAL INPUT MSB LSB†	ANALOG OUTPUT
1111111111	$-V_I$ (1023/1024)
1000000001	$-V_I$ (513/1024)
1000000000	$-V_I$ (512/1024) = $-V_{ref}/2$
0111111111	$-V_I$ (511/1024)
0000000001	$-V_I$ (1/1024)
0000000000	$-V_I$ (0/1024) = 0

† 1 LSB = $(2^{-10})\ V_I$
‡ 1 LSB = $(2^{-9})\ V_I$

TABLE 2. BIPOLAR (OFFSET BINARY) CODE

DAC DIGITAL INPUT MSB LSB‡	ANALOG OUTPUT
1111111111	$+V_I$ (511/512)
1000000001	$+V_I$ (1/512)
1000000000	0
0111111111	$-V_I$ (1/512)
0000000001	$-V_I$ (511/512)
0000000000	$-V_I$ (512/512) = $-V_I$

ADVANCE INFORMATION

TYPICAL APPLICATION DATA

The AD7533 and TLC7533 may be used in voltage output operation as shown in Figure 6. In this configuration, the input voltage is applied to the OUT1 terminal and the output voltage is taken from the REF terminal. The output voltage varies with the digital input code according to the equation shown. The output should be buffered to prevent loading errors due to the high output resistance of this circuit (typically 10 kΩ). The input voltage should not exceed 1.5 V to ensure nonlinearity errors less than 1 LSB.

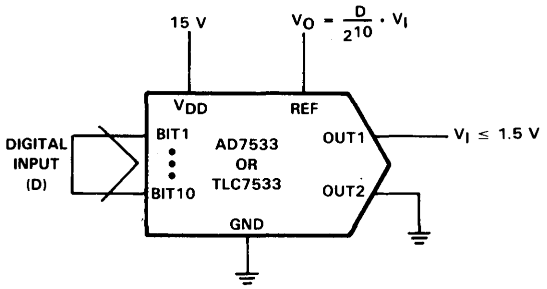
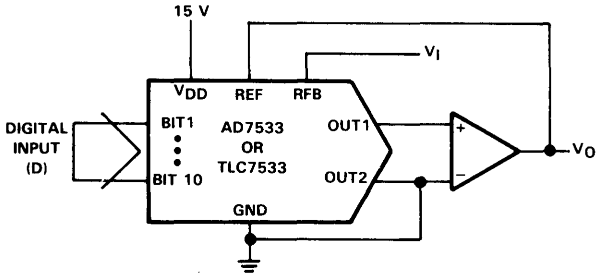


FIGURE 6. VOLTAGE OUTPUT OPERATION

By connecting the DAC in the feedback of an op-amp as shown in Figure 7, the circuit behaves as a programmable gain amplifier with the transfer function:

$$V_O = -V_I \left(\frac{1024}{D}\right)$$

where D = Digital Input Code (expressed as a decimal number)



GAIN TABLE	
D	V_O/V_I
1023	− 1.00097
512	− 2
256	− 4
128	− 8
2	− 512
1	− 1024
0	open loop

FIGURE 7. PROGRAMMABLE GAIN AMPLIFIER

TYPICAL APPLICATION DATA

The programmable function generator shown in Figure 8 produces both square and triangular wave output at a frequency determined by the digital input code. The digital input of the digitally programmable limit detector shown in Figure 9 determines the trip point of the PASS/FAIL output. For a digital input of 00000 00000, the threshold is 0 V, for 11111 11111, the threshold is $-V_{ref}$.

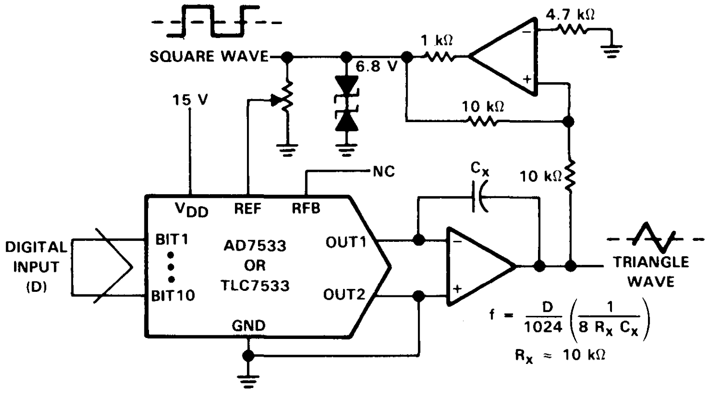


FIGURE 8. PROGRAMMABLE FUNCTION GENERATOR

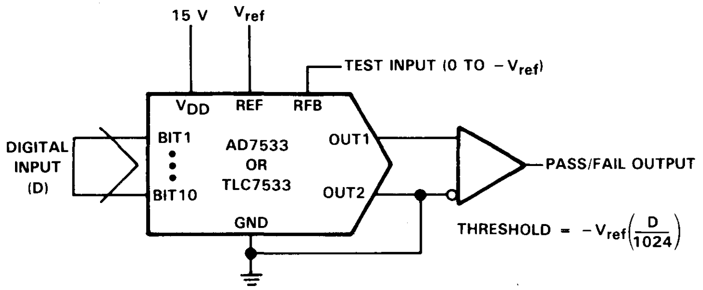
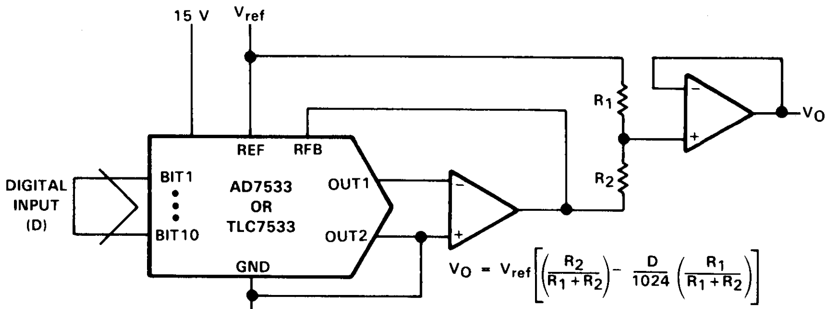


FIGURE 9. PROGRAMMABLE LIMIT DETECTOR

TYPICAL APPLICATION DATA



$$V_O = V_{ref}\left[\left(\frac{R_2}{R_1+R_2}\right) - \frac{D}{1024}\left(\frac{R_1}{R_1+R_2}\right)\right]$$

where: $0 \le D \le 1023$

FIGURE 10. MODIFIED SCALE-FACTOR AND OFFSET

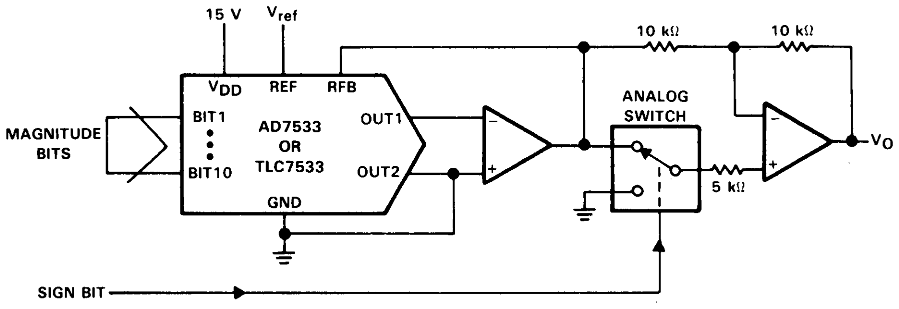


FIGURE 11. 10-BIT AND SIGN MULTIPLYING D/A

ADVANCE INFORMATION

E.2 TI Sockets

The sockets produced by Texas Instruments are designed for high-density packaging needs. The production sockets and burn-in/test sockets for DIP and PLCC packages, described in the following pages, are compatible with TMS320C1x devices.

For additional information about TI sockets, contact the nearest TI sales office or:

Texas Instruments Incorporated
Connector Systems Dept, MS 14-3
Attleboro, MA 02703
(617) 699-5242/5269
Telex: 92-7708

IC SOCKETS
PLASTIC LEADED CHIP CARRIER

PERFORMANCE SPECIFICATIONS

Mechanical

Recommended PCB thickness range: 0.062 in to 0.092 in
Recommended PCB hole size range: 0.032 in to 0.042 in
Vibration: 15 G
Shock: 100 G
Solderability: Per MIL-STD 202, Method 208
Insertion force: 0.59 lbs per position
Withdrawal force: 0.25 lbs per position
Normal force: 200 g min, 450 g typ
Wipe: 0.075 in min
Durability: 5 cycles min
Contact retention: 1.5 lbs min

Electrical

Current carrying capacity: 1 A
Insulation resistance: 5000 MΩ min
Dielectric withstanding voltage: 1000 V ac rms min
Capacitance: 1.0 pF max

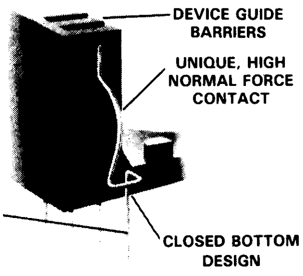
Environmental

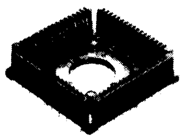
Operating temperature:
Operating: - 40°C to 85°C
Storage: −40°C to 95°C
Temperature cycling with humidity: will conform to final EIA specifications
Shelf life: 1 year min

MATERIALS

Body — Ryton R-4 (40% glass) U/L 94-VO rating
Contacts — CDA 510 spring temper
Contact finish — 90/10 tin (200 μin −400 μin) over 40 μin copper

Contact factory for detailed information

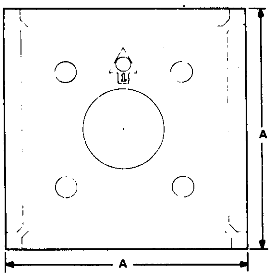




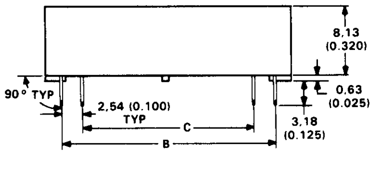
PART NUMBER SYSTEM

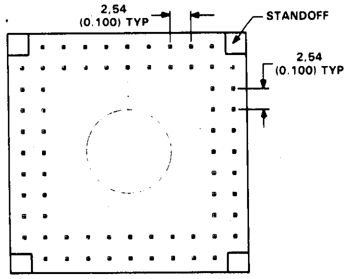
```
CPR  PH  XXX - X - X - O
 |    |    |    |   └Contact surface 1 — tin lead
 |    |    |    |    plating
 |    |    |    └Contact spacing 1 — 0.050 in
 |    |    └Number of pos (044, 052, 068, 084)
 |    └Plated thru hole, solder tail
 └TI socket Series
  Plastic leaded chip carrier
```

PLASTIC LEADER CHIP CARRIER CPR SERIES



Device guide barriers not shown





Pos	A	B	C
44	21,43 (0.844	17,78 (0.700)	12,70 (0.500)
52	23,98 (0.944)	20,32 (0.800)	15,24 (0.600)
68	29,06 (1.144)	25,40 (1.000)	20,32 (0.800)
84	34,14 (1.344)	30,48 (1.200)	25,40 (1.000)

Extraction tool available, consult factory.

TEXAS INSTRUMENTS
34 Forest Street • Attleboro, Massachusetts 02703

PRODUCT FEATURES

Can be loaded by top actuated insertion or press-in insertion, either manually or automatically
High reliability due to high pressure contact point
Open body and high stand-off design provide high efficiency in heat dissipation
High durability up to 10,000 cycles
Compact design

PERFORMANCE SPECIFICATIONS

Mechanical

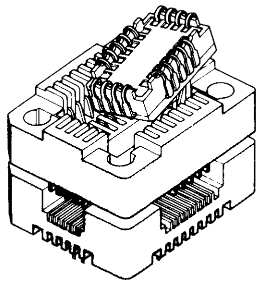
Durability: 10,000 cycles
Operating Temperature: 180 °C max

Electrical

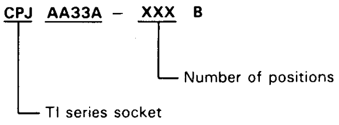
Contact rating: 1.0 A per contact
Contact resistance: 30 mΩ max
Insulation resistance: 1000 MΩ min
Dielectric withstanding voltage: 500 V ac rms min

MATERIALS

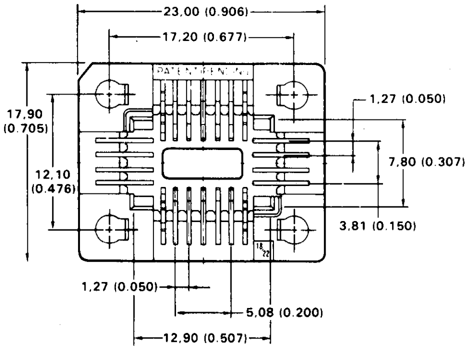
Body — ultem glass filled (U/L 94 VO)
Contact — copper alloy
Plating — overall gold plate

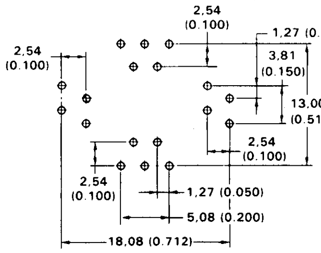


PART NUMBER SYSTEM

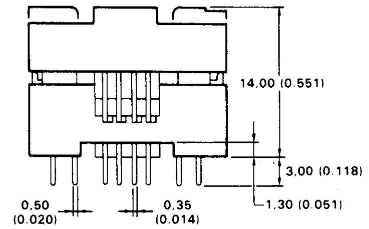


PLCC BURN-IN/TEST SOCKETS CPJ SERIES





SIZES: 18 PIN
22 PIN



Dimensions in parentheses are inches
Contact factory for detailed information

TEXAS INSTRUMENTS
34 Forest Street • Attleboro, Massachusetts 02703

PERFORMANCE SPECIFICATIONS

Mechanical

Accommodates IC leads 0.011 ± 0.003 in by 0.018 ± 0.003
Recommended PCB thickness range: 0.062 in to 0.092 in
Recommended PCB hole size range: 0.032 in to 0.042 in
Recommended hole grid pattern: 0.100 in ± 0.003 in each direction
Vibration: 15 G, 10-2000 Hz per MIL-STD 1344A, Method 2005.1 Test Condition III.
Shock: 100 G, sawtooth waveform, 2 shocks each direction per MIL-STD 202, Method 213, Test Condition I
Durability: 5 cycles, 10 mΩ max contact resistance change per MIL-STD 1344, Method 2016
Solderability: per MIL-STD 202, Method 208
Insertion force (C7X and C86): 16 oz (454 g) per pin max
Insertion force (C50): 12 oz per pin max
Withdrawal force: (40 g) per pin min

Electrical

Contact rating: 1.0 A per contact
Contact resistance: 20 mΩ max initial
Insulation resistance: 1000 MΩ at 500 V dc per MIL-STD 1344, Method 3003
Dielectric withstanding voltage: 1000 V ac rms per MIL-STD 1344, Method 3001.1
Capacitance: 1.0 pF max per MIL-STD 202, Method 305

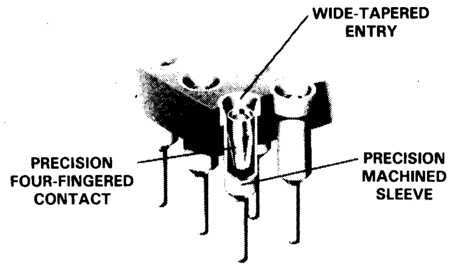
Environmental

Operating temperature: −55°C to 125°C, gold; −40°C to 100°C, tin
Corrosive atmosphere: 10 mΩ max contact resistance change when exposed to 22% ammonium sulfide for 4 hours
Gas tight: 10 mΩ max contact resistance change when exposed to nitric acid vapor for 1 hour
Temperature soak: 10 mΩ max contact resistance change when exposed to 105°C temperature for 48 hours
Shelf life: 12 months min

Materials (C7X, C50, and C86)

Body — PBT polyester U/L 94 VO rating
C7X & C50 Contacts — Outer sleeve: brass
Clip: BECU or PHBR
Contact finish Specified by Part Number — clip 30 μin gold over 50 μin nickel or 50 μin tin/lead over 50 μin nickel
— sleeve 10 μin gold over 50 μin nickel or 50 μin tin/lead over 50 μin nickel
C86 Contacts — Phosphor bronze base metal
C86 Contact-finish — Tin plate 200 μin over copper flash

C7X SERIES — SCREW MACHINE

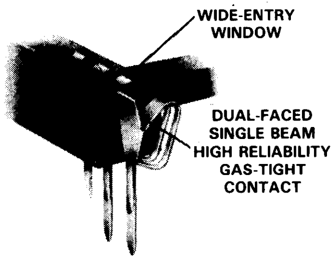


C7X SERIES PART NUMBER SYSTEM

C7X (X) XX − X X

- X — Variations
  - Solder Tail
    - 9 — Pin length 0.105/0.150
  - Wire Wrap
    - 3 — Pin length 0.510
- X — Plating (Sleeve/Clip)
  - 0 — Gold/Gold
  - 5 — Tin/Gold
- XX — Number of Positions
- (X) — S — Single-in-line package (where applicable)
- C7X — Screw Machine Socket
  - 1 — wire wrap
  - 2 — solder tail

C86 SERIES — STAMPED AND FORMED



C86 SERIES PART NUMBER SYSTEM

C 86 XX − 01

- 01 — Variation
  - 01 — Standard product
- XX — Number of positions
- 86 — Tin Dual Face Wipe Single Beam
- C — TI Socket Series

TEXAS INSTRUMENTS
34 Forest Street • Attleboro, Massachusetts 02703

PERFORMANCE SPECIFICATIONS

Mechanical

Accommodates IC leads 0.011 in by 0.018 in NOM
Recommended PCB thickness range: 0.062 in to 0.092 in
Recommended PCB hold size range: 0.032 in to 0.042 in
Durability: 10K cycles — CM Series, 5K cycles — CP/CQ
Solderability: per MIL-STD 202, Method 208

Electrical

Contact rating: 1.0 A per contact
Contact resistance: 20 mΩ max initial
Insulation resistance: 1000 MΩ at 500 V dc
Dielectric withstanding voltage: 1000 V ac rms
Capacitance: 1.0 pF max per MIL-STD 202, Method 305

Environmental

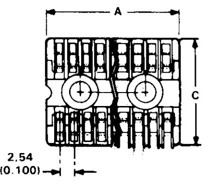
Operating temperature: −65 °C to 170 °C — CP/CM Series, −65 °C to 150 °C — CQ Series
Humidity: 10 mΩ max contact resistance
Temperature Soak: 10 mΩ max contact resistance change

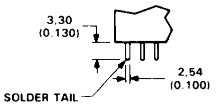
MATERIALS

Body — PPS (polyphenylen sulfide) glass filled U/L 94 VO
Contacts — Higher performance copper nickel alloy
Plating:† 4 μin of gold min over 100 μin of nickel min

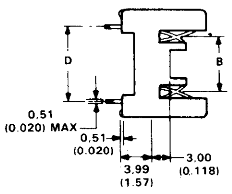
†For additional plating options consult the factory

BURN-IN/TEST DIP SOCKETS

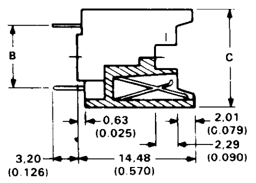




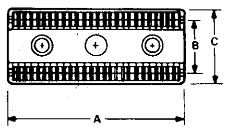
CQ37 SERIES

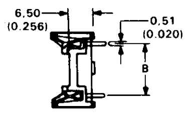


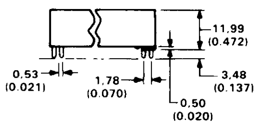
CP37 SERIES



CM37 SERIES







PART NUMBER SYSTEM

C X 37 XX − 22 S X

- X — Pin to pin: A – 0.100 centers; B – 0.070 centers
- S — PPS high temperature body material
- 22 — Copper nickel alloy Soldertail
- XX — Number of positions
- 37 — Overall gold plate
- X — Series Features: Q – Auto unloadable; P – High density mounting; M – Shrink 0.070 centers
- C — TI Socket Series

CQ37 SERIES

Number of Positions	A ±0.01 Length	D ±0.02	C ±0.01 Width	B ±0.01 Contact
14	20,32 (0.800)	12,70 (0.500)	15,24 (0.600)	7,62 (0.300)
16	22,35 (0.880)			
18	24,89 (0.980)			
20	27,43 (1.080)			
24	32,51 (1,280)	19,05 (0.750)	22,86 (0.900)	15,24 (0.600)
28	37,59 (1.480)			
40	52,83 (2.080)			
42	55,37 (2.180)			

CP37 SERIES

Number of Positions	A max Length	B ±0.02	C max Width
8	11,68 (0.460)	7,62 (0.300)	12,70 (0.500)
14	17,78 (0.700)		
16	20,32 (0.800)		
18	22,86 (0.900)		
20	25,40 (1.000)		
24	30,48 (1.200)	15,24 (0.600)	20,32 (0.800)
28	35,56 (1.400)		
40	50,80 (2.000)		

CM37 SERIES

Number of Positions	A ±0.016 Length	B ±0.02	C ±0.016 Width
28	27,18 (1.070)	10,67 (0.420)	17,20 (0.677)
40	37,85 (1.490)	16,51 (0.650)	23,11 (0.910)
42	39,62 (1.560)		
54	50,29 (1.980)		
64	59,18 (2.330)	20,32 (0.800)	26,92 (1.060)

Dimensions in parentheses are inches
Contact factory for detailed information

TEXAS INSTRUMENTS
34 Forest Street • Attleboro, Massachusetts 02703

E.3 Crystals

This section lists the commonly used crystal frequencies, crystal specification requirements, and the names of suitable vendors.

Table E-1 lists the commonly used crystal frequencies.

Table E-1. Commonly Used Crystal Frequencies

FREQUENCY	CRYSTAL MODE
14 MHz	Fundamental
18.432 MHz	Fundamental
20 MHz	Fundamental
20.48 MHz	Fundamental
25.6 MHz	May require third-overtone

A crystal connected across X1 and X2/CLKIN on the TMS320C14/E14 processor enables the internal oscillator, as shown in Figure E-1. The frequency of CLKOUT is one-fourth the crystal fundamental frequency. Crystal specifications requirements are listed below:

Load capacitance = 20 pF
Series resistance = 30 Ω
Power dissipation = 1 mW
Parallel resonant

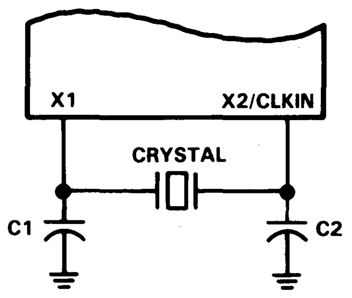


Figure E-1. Crystal Connection

Vendors of crystals suitable for the TMS320C14/E14 are listed below:

RXD,Inc.
Norfolk, NB
(800) 228-8108

N.E.L. Frequency Controls, Inc.
Burlington, WI
(414) 763-3591

CTS Knight, Inc.
Contact the local distributor

Appendix F

Programming the TMS320E14 EPROM Cell

The TMS320E14 includes a 4K x 16-bit EPROM implemented using an industry-standard EPROM cell for prototyping, early field testing, and production. The TMS320C14 with a 4K-word masked ROM then provides a migration path for cost-effective production.

Key features of the EPROM cell include standard programming and verification. The EPROM cell also includes a code protection feature that allows code to be protected against copyright violations. The protection feature can be used to protect reading the EPROM contents. This appendix describes erasure, programming and verification, and EPROM protection and verification.

This appendix contains the following major topics:

An EPROM adapter socket (part no. TMDX3270110) shown in Figure F-1, is available to provide 68-pin to 28-pin conversion for programming the TMS320E14.

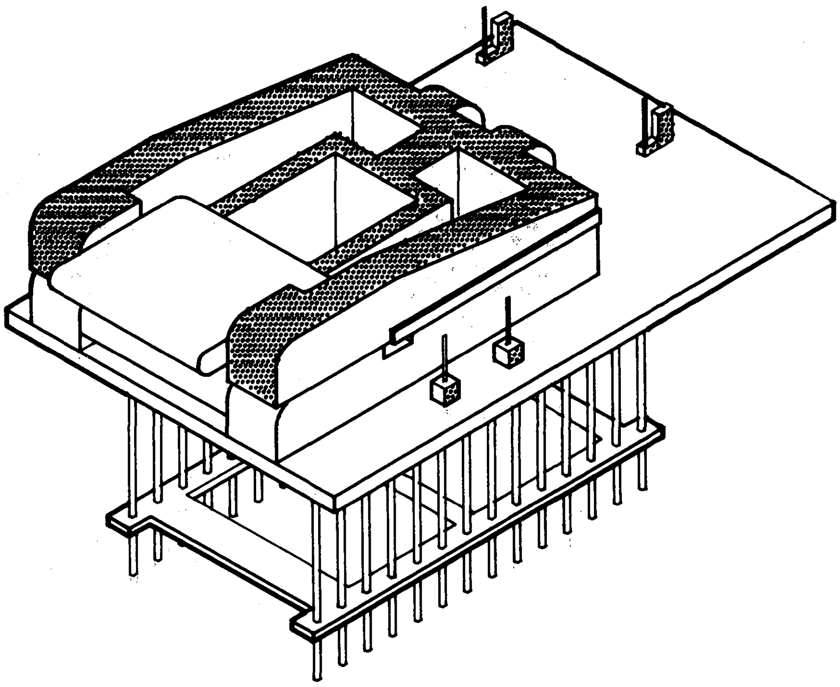


Figure F-1. EPROM Adaptor Socket

F.1 Programming and Verification

The TMS320E14 EPROM cell is programmed using the same family and device codes as the TMS27C64 8K x 8-bit EPROM. The TMS27C64 EPROM series are ultraviolet-light erasable, electrically programmable read-only memories, fabricated using HVCMOS technology. The TMS27C64 is pin-compatible with existing 28-pin ROMs and EPROMs. The TMS320E14, like the TMS27C64, operates from a single 5-V supply in the read mode and needs an additional supply (12.5-V) for programming. All programming signals are TTL level. For programming outside the system, existing EPROM programmers can be used. Locations may be programmed singly, in blocks, or at random. When programmed in blocks, the data is loaded into the EPROM cell one byte at a time, the low byte first and the high byte second. The EPROM adapter socket shown in Figure F-1has an inverter on it, so that from the EPROM programmer's point of view, data is loaded high byte, then low byte (as shown in Figure F-2). The device itself, however, expects data low byte first, high byte second.

The TMS320E14 uses 13 address lines to address the 4K-word memory in byte format (8K-byte memory). In word format, the most-significant byte of each word is assigned an odd address and the least-significant byte an even address in the byte format. Programming information should be downloaded to EPROM programmer memory in a high-byte to low-byte order for proper programming of the devices (see Figure F-2).

TMS320C14 On-Chip Program Memory (Word Format)

0(0000h)	1234h
1(0001h)	5678h
2(0002h)	9ABCh
3(0003h)	DEF0h
.	.
.	.
.	.
4095(0FFFh)	

TMS320E14 On-Chip Program Memory (Byte Format)

0(0000h)	34h
1(0001h)	12h
2(0002h)	78h
3(0003h)	56h
4(0004h)	BCh
5(0005h)	9Ah
6(0006h)	F0h
7(0007h)	DEh
.	.
.	.
.	.

EPROM Programmer Memory Byte Format with Adapter Socket

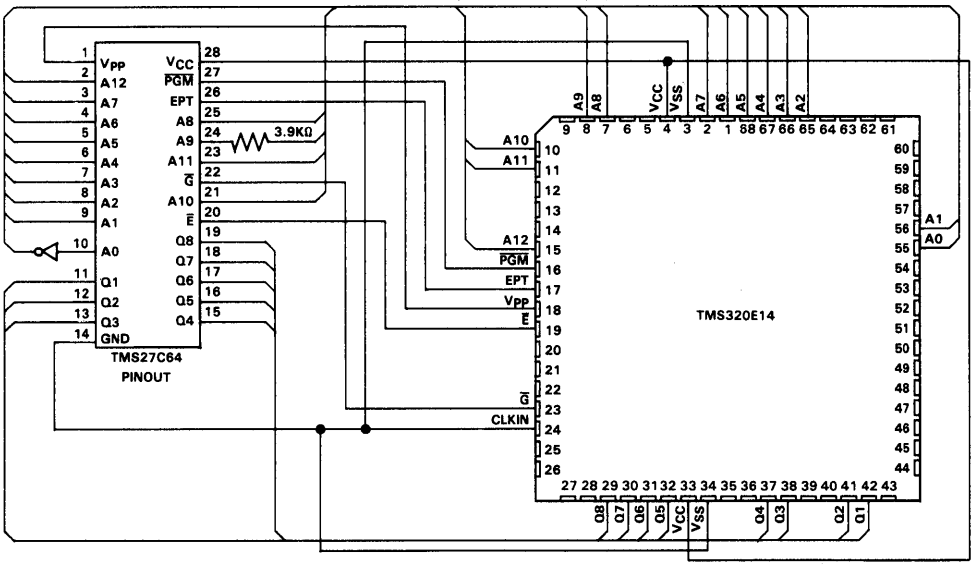
0(0000h)	12h
1(0001h)	34h
2(0002h)	56h
3(0003h)	78h
4(0004h)	9Ah
5(0005h)	BCh
6(0006h)	DEh
7(0007h)	F0h
.	.
.	.
.	.
8191(1FFFh)	

Figure F-2. EPROM Programming Data Format

Caution:

The TMS320E14 does not support the signature mode available with some EPROM programmers. The signature mode puts a high voltage (12.5 VDC) on pin A9. The TMS320E14 EPROM cell is not designed for this feature and will be damaged if subjected to it. A 3.9K ohm resistor is standard on the TI programmer socket between pin A9 and the programmer. This protects the device from unintentional use of the signature mode.

Figure F-3 shows the wiring conversion to program the TMS320E14 using the 28-pin pinout of the TMS27C64. The table of pin nomenclature provides a description of the TMS27C64 pins. The code to be programmed into the device should be in serial mode.



PIN NOMENCLATURE (TMS320E14)

NAME	I/O	DEFINITION
A12(MSB)-A0(LSB)	I	On-chip EPROM programming address lines
CLKIN	I	Clock oscillator input
$\overline{E}$	I	EPROM chip enable
EPT	I	EPROM test mode select
$\overline{G}$	I	EPROM output enable
GND	I	Ground
$\overline{PGM}$	I	EPROM write/program select
Q8(MSB)-Q1(LSB)	I/O	Data lines for byte-wide programming of on-chip 8K bytes of EPROM
$\overline{RS}$	I	Reset for initializing the device
V_{CC}	I	5-V to 6.5-V power supply
V_{PP}	I	12.5-V to 13-V power supply

Figure F-3. TMS320E14 EPROM Conversion to TMS27C64 EPROM Pinout

Table F-1 shows the programming levels required for programming, verifying, and reading the EPROM cell. The paragraphs following the table describe the function of each programming level.

Table F-1. TMS320E14 Programming Mode Levels

SIGNAL NAME †	TMS320E14 PIN	TMS27C64 PIN	PROGRAM	PROGRAM VERIFY	READ	EPROM PROTECT	PROTECT VERIFY
$\overline{E}$	19	20	V_{IL}	V_{IL}	V_{IL}	V_{IH}	V_{IL}
$\overline{G}$	23	22	V_{IH}	$\overline{PULSE}$	$\overline{PULSE}$	V_{IH}	V_{IL}
$\overline{PGM}$	16	27	$\overline{PULSE}$	V_{IH}	V_{IH}	V_{IH}	V_{IH}
V_{PP}	18	1	V_{PP}	V_{PP}	V_{CC}	V_{PP}	V_{CCP}
V_{CC}	4,33	28	V_{CCP}	V_{CCP}	V_{CC}	V_{CCP}	V_{CCP}
V_{SS}	3,34	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
CLKIN	24	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
EPT	17	26	V_{SS}	V_{SS}	V_{SS}	V_{PP}	V_{PP}
Q8-Q1	29-32,37,38, 41,42	19-15,13-11	D_{IN}	Q_{OUT}	Q_{OUT}	$Q_8 = \overline{PULSE}$	Q_8= RBIT
A12-A7	15,11,10,8, 7,2	2,23,21,24, 25,3	ADDR	ADDR	ADDR	X	X
A6	1	4	ADDR	ADDR	ADDR	X	V_{IL}
A5	68	5	ADDR	ADDR	ADDR	X	X
A4	67	6	ADDR	ADDR	ADDR	V_{IH}	X
A3-A0	66,65,56,55	7-10	ADDR	ADDR	ADDR	X	X

LEGEND:
† - Signal names shown are for TMS320E14 EPROM programming mode only.
V_{IH} = TTL high level
V_{IL} = TTL low level
ADDR = byte address bit
V_{PP} = 12.5 ± 0.25 V (FAST) or 13 ± 0.25 V (SNAP!)
V_{CC} = 5 ± 0.25 V
V_{CCP} = 6 ± 0.25 V (FAST) or 6.5 ± 0.25 V (SNAP!)
X = don't care
$\overline{PULSE}$ = low-going TTL pulse
D_{IN} = byte to be programmed at ADDR
Q_{OUT} = byte stored at ADDR

F.1.1 Erasure

Before programming, the device is erased by exposing the chip through the transparent lid to high-intensity ultraviolet light (wavelength 2537 angstroms). The recommended minimum exposure dose (UV-intensity × exposure-time) is 15 watt-seconds per square centimeter. A typical 12 milliwatt per square centimeter, filterless UV lamp will erase the device in 21 minutes. The lamp should be located about 2.5 centimeters above the chip during erasure. After erasure, all bits are in the high state. Note that normal ambient light contains the correct wavelength for erasure. Therefore, when using the TMS320E14, the window should be covered with an opaque label.

F.1.2 Fast Programming

After erasure (all memory bits in the cell are a logic one), logic zeroes are programmed into the desired locations. The fast programming algorithm, shown in Figure F-4, is normally used to program the entire EPROM contents, although individual locations may be programmed separately. A programmed logic zero can only be erased by ultraviolet light. Data is presented in parallel (eight bits) on pins Q8-Q1. Once addresses and data are stable, $\overline{PGM}$ is pulsed. The programming mode is achieved when V_{PP} = 12.5 V, $\overline{PGM}$ = V_{IL}, V_{CC} = 6.0 V, $\overline{G}$ = V_{IH}, and $\overline{E}$ = V_{IL}. More than one TMS320E14 can be programmed when the devices are connected in parallel. Locations can be programmed in any order.

Fast programming uses two types of programming pulses: prime and final. The length of the prime pulse is 1 ms. After each prime pulse, the byte being programmed is verified. If correct data is read, the final programming pulse is applied; if correct data is not read, an additional 1-ms prime pulse is applied up to a maximum of 25 times. The final programming pulse is 3X times the number of prime programming pulses applied. This sequence of programming and verification is performed at V_{CC} = 6.0 V, and V_{PP} = 12.5 V. When the full fast programming routine is complete, all bits are verified with V_{CC} = V_{PP} = 5 V.

F.1.3 SNAP! Pulse Programming

The EPROM can be programmed using the TI SNAP! Pulse programming algorithm illustrated by the flowchart of Figure F-5, which can reduce programming time to a nominal of one second. Actual programming time will vary as a function of the programmer used. Data is presented in parallel (eight bits) on pins Q8 through Q1. Once addresses and data are stable, $\overline{PGM}$ is pulsed.

The SNAP! Pulse programming algorithm uses pulses of 100 microseconds followed by a byte verification to determine when the addressed byte has been successfully programmed. Up to ten 100 microsecond pulses per byte are provided before a failure is recognized.

The programming mode is achieved when V_{PP} = 13.0 V, V_{CC} = 6.5 V, V and $\underline{G}$ = V_{IH}, and $\overline{E}$ = V_{IL}. More than one device can be programmed when the devices are connected in parallel. Locations can be programmed in any order. When the SNAP! Pulse programming routine is complete, all bits are verified with V_{CC} = V_{PP} = 5 V.

F.1.4 Program Verify

Programmed bits may be verified with V_{PP} = 12.5 V when $\overline{G}$ = V_{IL}, $\overline{E}$ = V_{IL}, and $\overline{PGM}$ = V_{IH}. Figure F-6 shows the timing for the program and verify operations for both Fast and SNAP! Pulse programming.

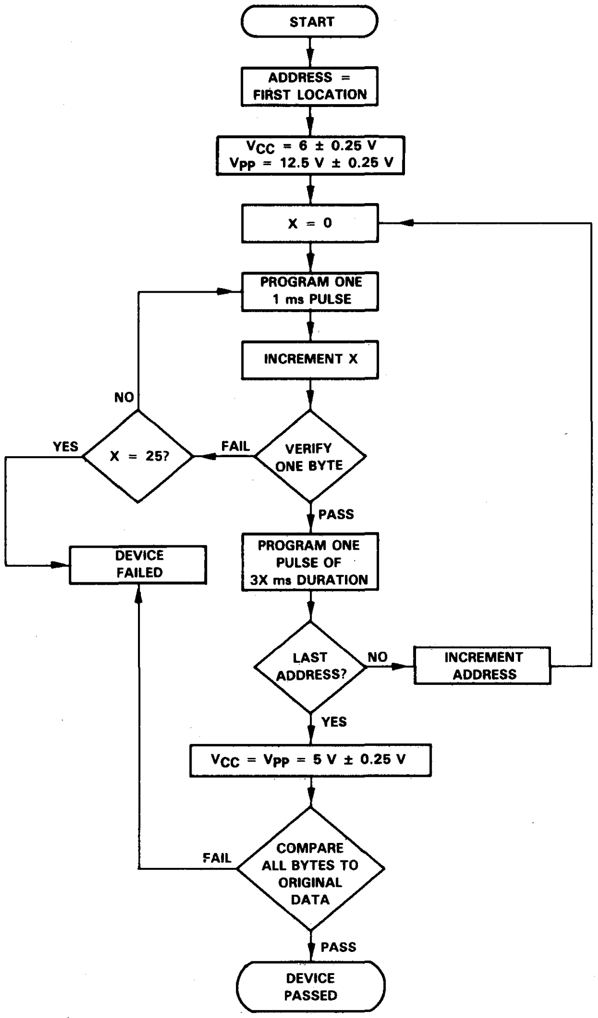


Figure F-4. Fast Programming Flowchart

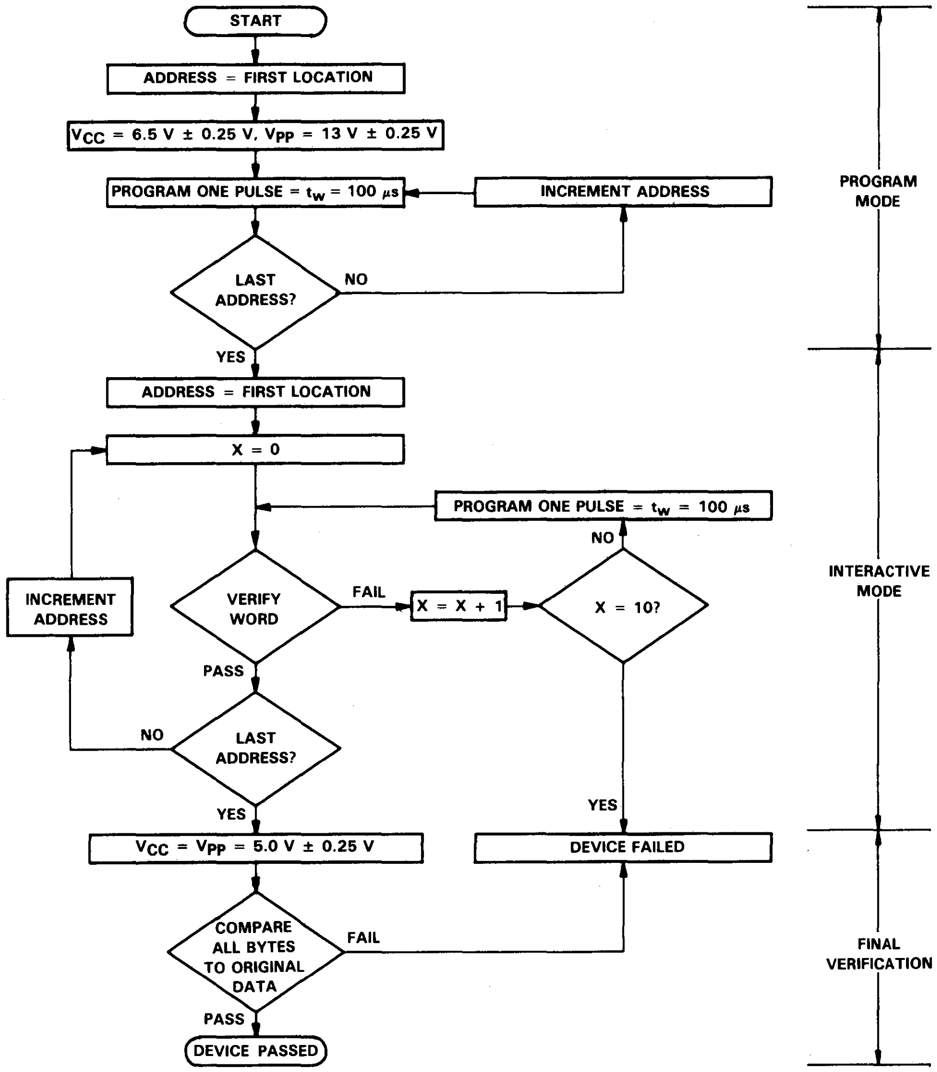


Figure F-5. SNAP! Pulse Programming Flowchart

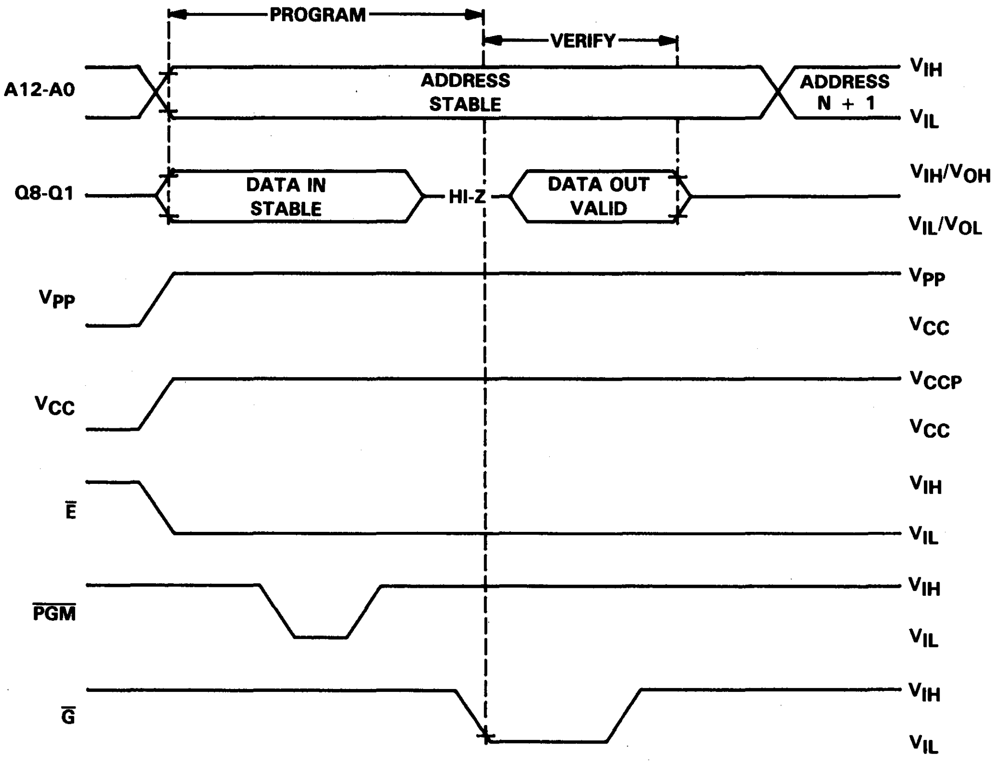


Figure F-6. Programming Timing

F.1.5 Program Inhibit

Programming may be inhibited by maintaining a high level input on the $\overline{E}$ pin or $\overline{PGM}$ pin.

F.1.6 Read

The EPROM contents may be read independent of the programming cycle, provided the RBIT (ROM protect bit) has not been programmed. The read is accomplished by setting $\overline{E}$ to zero and pulsing $\overline{G}$ low. The contents of the EPROM location, selected by the value on the address inputs, appear on Q8-Q1.

F.1.7 Output Disable

During the EPROM programming process, the EPROM data outputs may be disabled, if desired, by establishing the output disable state. This state is selected by setting the $\overline{G}$ pin high or $\overline{E}$ pin high. Whichever occurs first will determine the time to high impedance on the outputs. While output disable is selected, Q8-Q1 are placed in the high-impedance state.

F.2 EPROM Protection and Verification

This section describes the code protection feature included in the EPROM cell, which protects code against copyright violations. Table F-2 shows the programming levels required for protecting the EPROM and verifying the protection. The paragraphs following the table describe the protect and verify functions.

Table F-2. TMS320E14 EPROM Protect and Protect Verify Mode Levels

SIGNAL	TMS320E14 PIN	TMS27C64 PIN	EPROM PROTECT	PROTECT VERIFY
$\overline{E}$	19	20	V_{IH}	V_{IL}
$\overline{G}$	23	22	V_{IH}	V_{IL}
$\overline{PGM}$	16	27	V_{IH}	V_{IH}
V_{PP}	18	1	V_{PP}	V_{CCP}
V_{CC}	4	28	V_{CCP}	V_{CCP}
V_{SS}	3	14	V_{SS}	V_{SS}
CLKIN	24	14	V_{SS}	V_{SS}
EPT	17	26	V_{PP}	V_{PP}
Q8-Q1	29-32,37,38, 41,42	19-15,13-11	Q8=$\overline{PULSE}$	Q8=RBIT
A12-A10	15,11,10	2,23,21	X	X
A9-A7	8,7,2	24,25,3	X	X
A6	1	4	X	V_{IL}
A5	68	5	X	X
A4	67	6	V_{IH}	X
A3-A0	66,65,56,55	7-10	X	X

LEGEND:
V_{IH} = TTL high level; V_{IL} = low-level TTL, V_{CC} = 5 ± 0.25 V; V_{PP} = 12.5 ± 0.5 V;
V_{CCP} = 6 ± 0.25 V (FAST) or 6.5 ± 0.25 V (SNAP!);
X = don't care; $\overline{PULSE}$ = low-going TTL level pulse; RBIT = ROM protect bit

F.2.1 EPROM Protection

The EPROM protection facility is used to completely disable reading of the EPROM contents to guarantee security of proprietary algorithms. This facility is implemented through a unique EPROM cell called the RBIT (ROM protect bit) cell. Once the contents to be protected are programmed into the EPROM, the RBIT is programmed, disabling access to the EPROM contents and disabling the microprocessor mode on the device. Once programmed, the RBIT can only be cleared by erasing the entire EPROM array with ultraviolet light, thereby maintaining security of the proprietary algorithm. Programming the RBIT is accomplished using the EPROM protection cycle, which consists of setting the $\overline{E}$, $\overline{G}$, $\overline{PGM}$, and A4 pins high, V_{PP} and EPT to 12.5 ± 0.5 V, and pulsing Q8 low. The complete sequence of operations involved in programming the RBIT is shown in the flowchart of Figure F-7. The required setups in the figure are detailed in Table F-2.

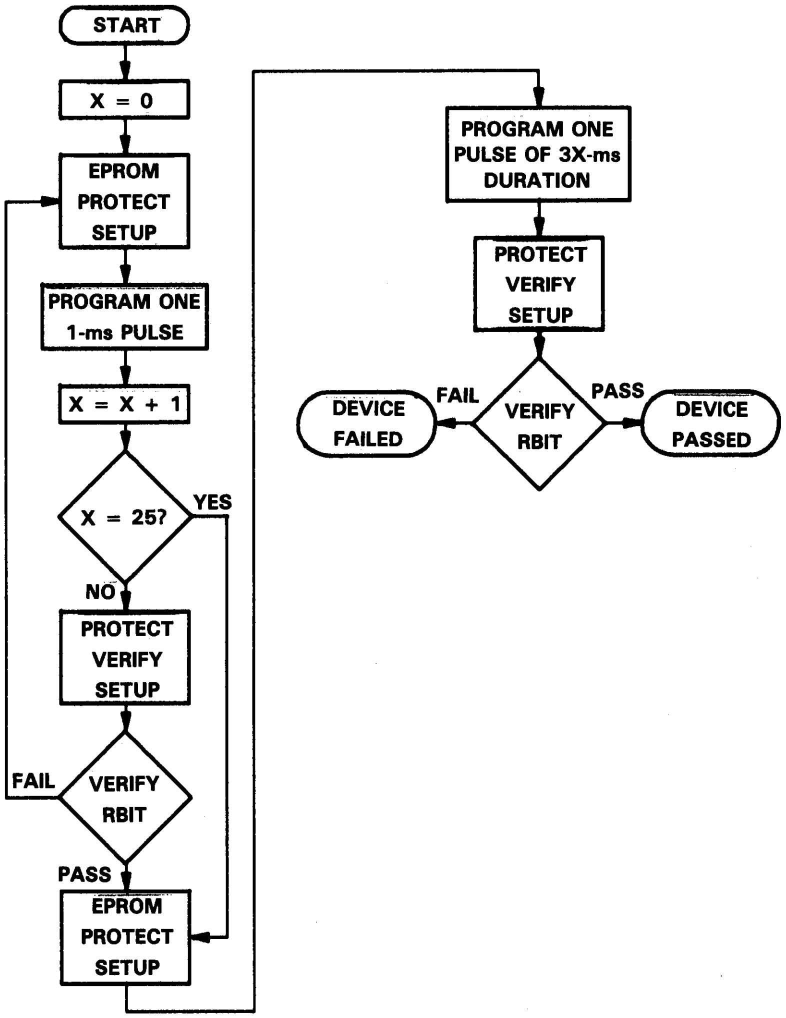
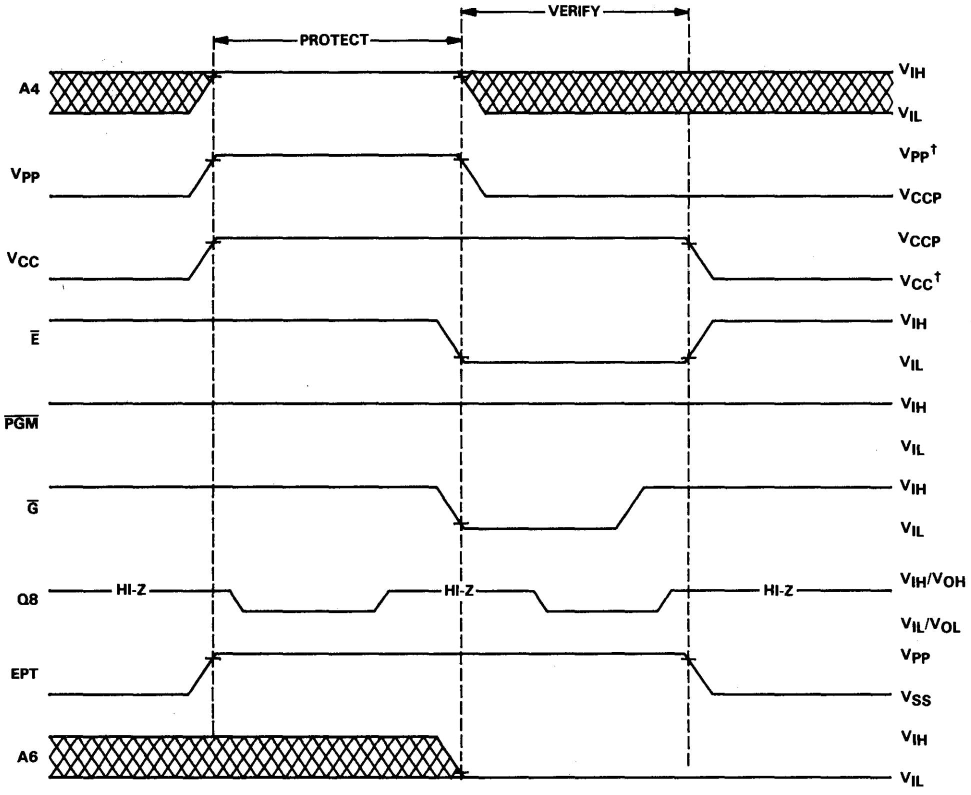


Figure F-7. EPROM Protection Flowchart

F.2.2 Protect Verify

Protect verify is used following the EPROM protection to verify correct programming of the RBIT (see Figure F-7). When using protect verify, Q8 outputs the state of the RBIT. When RBIT = 1, the EPROM is unprotected; when RBIT = 0, the EPROM is protected. The EPROM protection and verify timings are shown in Figure F-8.



† 12.5 V V_{PP} and 6.0 V V_{CC} for Fast Programming; 13.0 V V_{PP} and 6.5 V V_{CC} for SNAP! Pulse Programming.

Figure F-8. EPROM Protection Timing

Index

TEXAS INSTRUMENTS

Printed in U.S.A., August 1988
2558514-9701

SPRU032