First-Generation TMS320 User's Guide

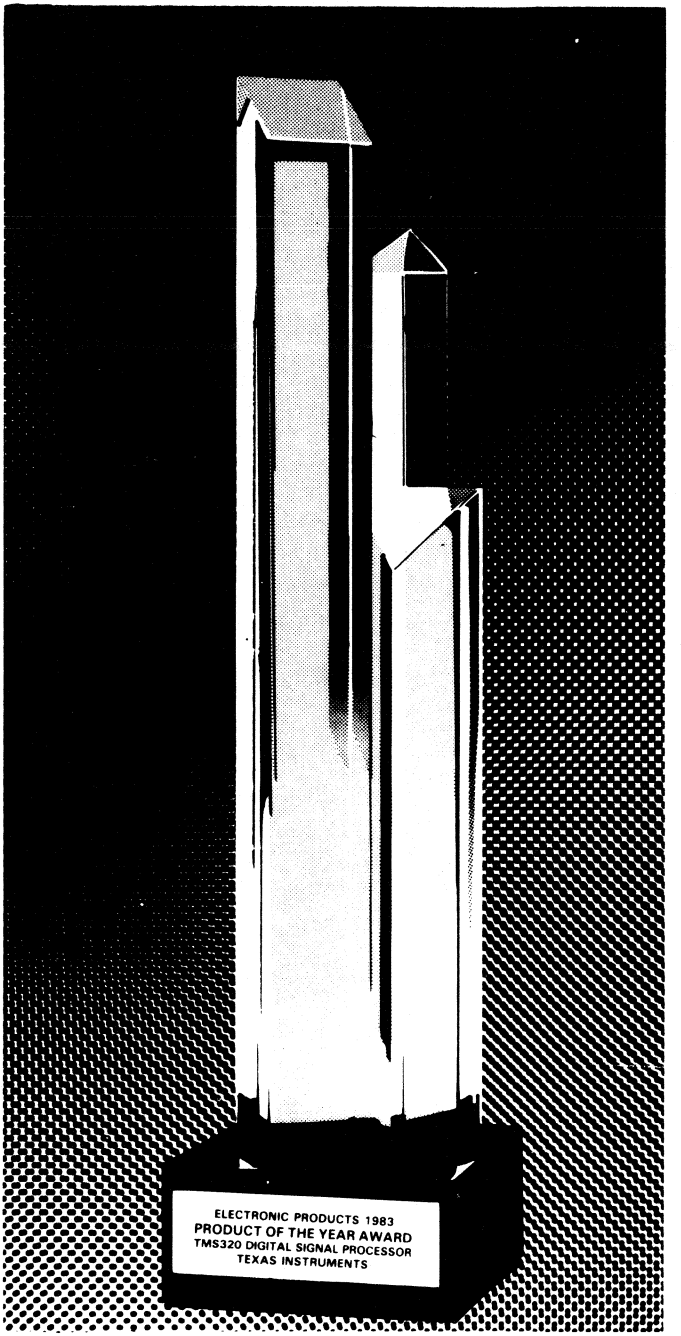
TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes in the devices or the device specifications identified in this publication without notice. TI advises its customers to obtain the latest version of device specifications to verify, before placing orders, that the information being relied upon by the customer is current.

In the absence of written agreement to the contrary, TI assumes no liability for TI applications assistance, customer's product design, or infringement of patents or copyrights of third parties by or arising from use of semiconductor devices described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor devices might be or are used.

Dépôt légal = Janvier 1988
ISBN 2-86886-024-9
Printed in France


ELECTRONIC PRODUCTS 1983
PRODUCT OF THE YEAR AWARD
TMS320 DIGITAL SIGNAL PROCESSOR
TEXAS INSTRUMENTS

Contents

Illustrations

Tables

Table *Page*

1. Introduction

The TMS320 family of 16/32-bit single-chip digital signal processors combines the flexibility of a high-speed controller with the numerical capability of an array processor, offering an inexpensive alternative to custom VLSI and multichip bit-slice processors.

The TMS32010, the first digital signal processor in the TMS320 family, was introduced in 1983. During that year, the TMS32010 was named "Product of the Year" by the magazine, *Electronic Products*. 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 to many telecommunications, computer, commercial, industrial, and military applications.

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.

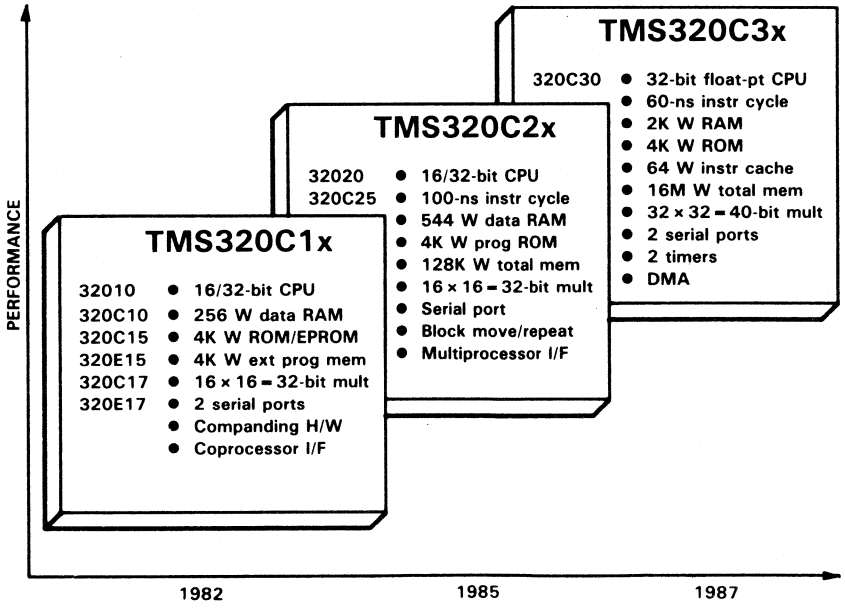


Figure 1-1. TMS320 Device Evolution

Throughout this document, the first-generation device group within the TMS320 family will be referred to as TMS320C1x. The specific members of the first-generation TMS320 include:

- TMS32010, the first 20-MHz digital signal processor
- TMS32010-16, a 16-MHz version of the TMS32010
- TMS32010-25, a 25-MHz version of the TMS32010
- TMS320C10, a CMOS 20-MHz version of the TMS32010
- TMS320C10-25, a 25-MHz version of the TMS320C10
- TMS320C15, a TMS320C10 with expanded ROM and RAM
- TMS320E15, an EPROM version of the TMS320C15
- TMS320C15-25, a 25-MHz version of the TMS320C15
- TMS320C17, a TMS320C15 with serial and coprocessor ports
- TMS320E17, an EPROM version of the TMS320C17
- TMS320C17-25, a 25-MHz version of the TMS320C17.

Plans for expansion of the TMS320 family include more spinoffs of the existing generations as well as more powerful future generations of digital signal processors.

The TMS320 family combines the high performance and specialized features necessary in digital signal processing (DSP) applications with an extensive program of development support, including hardware and software development products, product documentation, textbooks, newsletters, DSP design workshops, and a variety of application reports. See Appendix E for a discussion of the wide range of development tools available.

1.1 General Description

The combination of the TMS320's Harvard-type architecture (separate program and data buses) and its special digital signal processing (DSP) instruction set provide speed and flexibility to produce a microprocessor family capable of executing 6.25 MIPS (million instructions per second). The TMS320 family optimizes speed by implementing functions in hardware that other processors implement through software or microcode. This hardware-intensive approach provides the design engineer with power previously unavailable on a single chip.

Table 1-1 provides an overview of the TMS320C1x group of processors with comparisons of technology, memory, I/O, cycle timing, package type, and military support.

Table 1-1. TMS320C1x Processors Overview

DEVICE	TECH	MEMORY ON-CHIP RAM	ROM	EPROM	OFF-CHIP PROG	I/O* SER	PAR	CYCLE TIME (ns)	PACKAGE TYPE DIP	PLCC
TMS32010-16	NMOS	144	1.5K	–	4K	–	8x16	250	40	–
TMS32010†	NMOS	144	1.5K	–	4K	–	8x16	200	40	–
TMS32010-25	NMOS	144	1.5K	–	4K	–	8x16	160	40	–
TMS32Q11	NMOS	144	1.5K	–	–	2	6x16	200	40	–
TMS320C10‡	CMOS	144	1.5K	–	4K	–	8x16	200	40	44
TMS320C10-25	CMOS	144	1.5K	–	4K	–	8x16	160	40	44
TMS320C15‡	CMOS	256	4K	–	4K	–	8x16	200	40	44
TMS320C15-25	CMOS	256	4K	–	4K	–	8x16	160	40	44
TMS320E15‡	CMOS	256	–	4K	4K	–	8x16	200	40	–
TMS320C17	CMOS	256	4K	–	–	2	6x16	200	40	44
TMS320C17-25	CMOS	256	4K	–	–	2	6x16	160	40	44
TMS320E17	CMOS	256	–	4K	–	2	6x16	200	40	–

*SER = serial; PAR = parallel.
†Military version available.
‡Military versions planned; contact nearest sales office for availability.

The first generation of the TMS320 family includes the TMS32010, TMS32010-16, TMS32010-25, processed in NMOS technology, and the TMS320C10, TMS320C10-25, TMS320C15/E15, TMS320C15-25, TMS320C17/E17, and TMS320C17-25, processed in CMOS technology.

The **TMS32010**, the first TMS320 family member, is a microprocessor capable of achieving a 16 x 16-bit multiply in a single 200-ns cycle. On-chip data memory of 144 words is available. Up to 4K words of off-chip program memory can be executed at full speed. The TMS32010 is also available in a microcomputer version, with 1.5K words of on-chip program ROM and up to 2.5K words of off-chip program memory for a total of 4K words. This ROM-code version can also operate entirely from off-chip ROM for ease of prototyping, code update, and field upgradeability.

The **TMS32010-16**, a 16 MHz version of the TMS32010, provides a low-cost alternative for DSP applications not requiring the maximum operating fre-

quency of the TMS32010. Some applications for which the TMS32010-16 is well suited include servo control, high-speed controllers, low-end modems, audio processing, data encryption, and vibration analysis. The device can execute 4 million instructions per second and perform a 16 x 16-bit multiply in 250 ns. The TMS32010-16 provides a direct EPROM interface for single-cycle program memory access, thereby offering a cost-effective method for system development and modification. The device is pin-for-pin and software compatible with the higher-frequency, 20-MHz TMS32010 and its development tools.

The **TMS32010-25**, a 160-ns instruction cycle time version of the TMS32010, is intended for higher-performance applications that use off-chip program memory and require 25 percent greater processor throughput (6.25 million instructions per second) than the TMS32010. Existing TMS32010 designs can take advantage of the enhanced throughput simply by increasing the input clock cycle time to 25 MHz without rewriting software.

The **TMS320C10** has a 200-ns instruction cycle time and is object-code and pin-for-pin compatible with the TMS32010. The TMS320C10 is processed in CMOS technology, achieving a power dissipation less than one-sixth that of the NMOS device. Because of its low-power dissipation (165 mW), the TMS320C10 is ideal for power-sensitive applications such as digital telephony and portable consumer products. A masked ROM option is available for the TMS320C10.

The **TMS320C10-25**, a 25 MHz version of the TMS320C10, has a 160-ns instruction cycle time. Its lower power and higher speed make it well suited for high-performance DSP applications.

The **TMS320C15** and **TMS320E15** are fully object-code and pin-for-pin compatible with the TMS32010 and offer expanded on-chip RAM of 256 words and on-chip program ROM (TMS320C15) or EPROM (TMS320E15) of 4K words. The devices are processed in CMOS technology. The TMS320C15 is also available in a 160-ns version, the **TMS320C15-25**.

The **TMS320C17** and **TMS320E17** are dedicated microcomputers with 256 words of on-chip RAM and 4K words of on-chip program ROM (TMS320C17) or EPROM (TMS320E17). The TMS320C17/E17 features a dual-channel serial interface, on-chip companding hardware (μ-law/A-law), a serial port timer, and a latched 16-bit coprocessor port for direct microprocessor I/O interface. The devices are object-code compatible with the TMS32010, and processed in CMOS technology. The TMS320C17 is also available in a 160-ns version, the **TMS320C17-25**.

1.2 Key Features

Some of the key features of the TMS320C1x devices are listed below. Specific devices for a particular feature are enclosed in parentheses.

- Instruction cycle timing:
  - 160 ns (TMS32010-25/C10-25/C15-25/C17-25)
  - 200 ns (TMS32010/C10/C15/E15/C17/E17)
  - 250 ns (TMS32010-16)
- 144/256-word on-chip data RAM
- 1.5K/4K-word on-chip program ROM
- 4K-word on-chip program EPROM (TMS320E15/E17)
- EPROM code protection for copyright security
- 4K-word total external memory at full speed
- 16-bit bidirectional data bus at 50-Mbps transfer rate
- 32-bit ALU/accumulator
- 16 x 16-bit parallel multiplier with a 32-bit product
- 0 to 16-bit barrel shifter
- On-chip clock generator
- Eight input and eight output channels
- Dual-channel serial port with timer (TMS320C17/E17)
- Direct interface to combo-codecs (TMS320C17/E17)
- On-chip μ-law/A-law companding hardware (TMS320C17/E17)
- 16-bit coprocessor interface (TMS320C17/E17)
- Single 5-V supply
- Device packaging:
  - 40-pin DIP (TMS32010/C10/C15/E15/C17/E17)
  - 44-lead PLCC (TMS320C10; available in 1988 for the TMS320C15/C17)
- Technology:
  - NMOS (TMS32010)
  - CMOS (TMS320C10/C15/E15/C17/E17)
- Commercial and military versions available.

1.3 Typical Applications

The TMS320 family's unique versatility and realtime performance offer flexible design approaches in a variety of applications. In addition, TMS320 devices can simultaneously provide the multiple functions often required in those complex applications. Table 1-2 lists typical TMS320 family applications.

Table 1-2. Typical Applications of the TMS320 Family

GENERAL-PURPOSE DSP	GRAPHICS/IMAGING	INSTRUMENTATION
Digital Filtering Convolution Correlation Hilbert Transforms Fast Fourier Transforms Adaptive Filtering Windowing Waveform Generation	3-D Rotation Robot Vision Image Transmission/ Compression Pattern Recognition Image Enhancement Homomorphic Processing Workstations Animation/Digital Map	Spectrum Analysis Function Generation Pattern Matching Seismic Processing Transient Analysis Digital Filtering Phase-Locked Loops
VOICE/SPEECH	**CONTROL**	**MILITARY**
Voice Mail Speech Vocoding Speech Recognition Speaker Verification Speech Enhancement Speech Synthesis Text-to-Speech	Disk Control Servo Control Robot Control Laser Printer Control Engine Control Motor Control	Secure Communications Radar Processing Sonar Processing Image Processing Navigation Missile Guidance Radio Frequency Modems
TELECOMMUNICATIONS		**AUTOMOTIVE**
Echo Cancellation ADPCM Transcoders Digital PBXs Line Repeaters Channel Multiplexing 1200 to 19200-bps Modems Adaptive Equalizers DTMF Encoding/Decoding Data Encryption	FAX Cellular Telephones Speaker Phones Digital Speech Interpolation (DSI) X.25 Packet Switching Video Conferencing Spread Spectrum Communications	Engine Control Vibration Analysis Antiskid Brakes Adaptive Ride Control Global Positioning Navigation Voice Commands Digital Radio Cellular Telephones
CONSUMER	**INDUSTRIAL**	**MEDICAL**
Radar Detectors Power Tools Digital Audio/TV Music Synthesizer Educational Toys	Robotics Numeric Control Security Access Power Line Monitors	Hearing Aids Patient Monitoring Ultrasound Equipment Diagnostic Tools Prosthetics Fetal Monitors

1.4 How To Use This Manual

The purpose of this user's guide is to serve as a reference book for the first-generation TMS320 digital signal processors. Sections 2 through 6 provide specific information about the architecture and operation of the device. Electrical specifications and mechanical data can be found in the data sheet (Appendix A).

The following table lists each section and briefly describes the section contents.

Section 2. Pinouts and Signal Descriptions. Drawings of the DIP and PLCC packages for TMS320C1x devices. Functional listings of the signals, their pin locations, and descriptions.

Section 3. Architecture. TMS320C1x design description, hardware components, and device operation. Functional block diagrams and internal hardware summary table.

Section 4. Assembly Language Instructions. Addressing modes and format descriptions. Instruction set summary listed according to function. Alphabetized individual instruction descriptions with examples.

Section 5. Software Applications. Software application examples for the use of various TMS320C1x instruction set features.

Section 6. Hardware Applications. Hardware design techniques and application examples for interfacing to codecs, external memory, or common 4/8/16/32-bit microcomputers and microprocessors.

Seven appendices are included to provide additional information.

Appendix A. First-Generation TMS320 Data Sheet. Electrical specifications, timing, and mechanical data for all TMS320C1x devices.

Appendix B. SMJ32010/C10 Data Sheets. Electrical specifications, timing, and mechanical data for these military devices.

Appendix C. ROM Codes. Discussion of ROM codes (mask options) and the procedure for implementation.

Appendix D. Quality and Reliability. Discussion of Texas Instruments quality and reliability criteria for evaluating performance.

Appendix E. Development Support/Part Order Information. Listings of the hardware and software available to support the TMS320C1x devices.

Appendix F. DSP Memories, Peripherals, and Sockets. Listings of the memories, peripherals, and sockets available to support the TMS320C1x devices in DSP applications.

Appendix G. Programming the TMS320E15/E17 EPROM Cell. Procedure for programming and verifying the EPROM cell using the 28-pin TMS27C64.

1.5 References

The following reference list contains useful information regarding functions, operations, and applications of digital signal processing. These books also list other references to many useful technical papers. The references are organized into categories of general DSP, speech, image processing, and digital control theory.

General Digital Signal Processing:

Antoniou, Andreas, *Digital Filters: Analysis and Design*. New York, NY: McGraw-Hill Company, Inc., 1979.

Brigham, E. Oran, *The Fast Fourier Transform*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1974.

Burrus, C.S. and Parks, T.W., *DFT/FFT and Convolution Algorithms*. New York, NY: John Wiley and Sons, Inc., 1984.

Digital Signal Processing Applications with the TMS320 Family, Texas Instruments, 1986.

Gold, Bernard and Rabiner, Lawrence R., *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

Gold, Bernard and Rader, C.M., *Digital Processing of Signals*. New York, NY: McGraw-Hill Company, Inc., 1969.

Hamming, R.W., *Digital Filters*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1977.

IEEE ASSP DSP Committee (Editor), *Programs for Digital Signal Processing*. New York, NY: IEEE Press, 1979.

Jackson, Leland B., *Digital Filters and Signal Processing*. Hingham, MA: Kluwer Academic Publishers, 1986.

Jones, D.L. and Parks, T.W., *A Digital Signal Processing Laboratory Using the TMS32010*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.

Morris, L. Robert, *Digital Signal Processing Software*. Ottawa, Canada: Carleton University, 1983.

Oppenheim, Alan V. (Editor), *Applications of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978.

Oppenheim, Alan V. and Schafer, R.W., *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

Oppenheim, Alan V. and Willsky, A.N. with Young, I.T., *Signals and Systems*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1983.

Parks, T.W. and Burrus, C.S., *Digital Filter Design*. New York, NY: John Wiley and Sons, Inc., 1987.

Treichler, J.R., Johnson, Jr., C.R., and Larimore, M.G., *A Practical Guide to Adaptive Filter Design*. New York, NY: John Wiley and Sons, Inc., 1987.

Speech:

Gray, A.H. and Markel, J.D., *Linear Production of Speech*. New York, NY: Springer-Verlag, 1976.

Jayant, N.S. and Noll, Peter, *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1984.

Papamichalis, Panos, *Practical Approaches to Speech Coding*. Englewood Cliffs NJ: Prentice-Hall, Inc., 1987.

Rabiner, Lawrence R. and Schafer, R.W., *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978.

Image Processing:

Andrews, H.C. and Hunt, B.R., *Digital Image Restoration*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1977.

Gonzales, Rafael C. and Wintz, Paul, *Digital Image Processing*. Reading, MA: Addison-Wesley Publishing Company, Inc., 1977.

Pratt, William K., *Digital Image Processing*. New York, NY: John Wiley and Sons, 1978.

Digital Control Theory:

Jacquot, R., *Modern Digital Control Systems*. New York, NY: Marcel Dekker, Inc., 1981.

Katz, P., *Digital Control Using Microprocessors*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981.

Kuo, B.C., *Digital Control Systems*. New York, NY: Holt, Reinholt and Winston, Inc., 1980.

Moroney, P., *Issues in the Implementation of Digital Feedback Compensators*. Cambridge, MA: The MIT Press, 1983.

Phillips, C. and Nagle, H., *Digital Control System Analysis and Design*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1984.

2. Pinouts and Signal Descriptions

The TMS320C1x (first-generation TMS320) digital signal processors are all available in a 40-pin dual-in-line (DIP) package. The TMS320C10 is also packaged in a 44-pin plastic-leaded chip carrier (PLCC). The TMS320C15 and TMS320C17 will be available in a PLCC package in the future. Contact the nearest TI sales office for availability.

This section provides the pinouts and signal definitions in the following subsections:

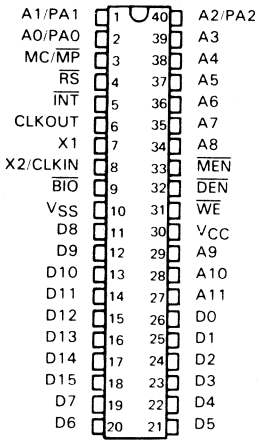
- TMS320C1x Pinouts (Section 2.1 on page 2-2)
- TMS32010/C10/C15/E15 Signal Descriptions (Section 2.2 on page 2-3)
- TMS320C17/E17 Signal Descriptions (Section 2.3 on page 2-6)

Electrical specifications and mechanical data are given in Appendix A, the First-Generation TMS320 Data Sheet. Refer to Appendix G for the pinout used in programming the TMS320E15/E17 EPROM.

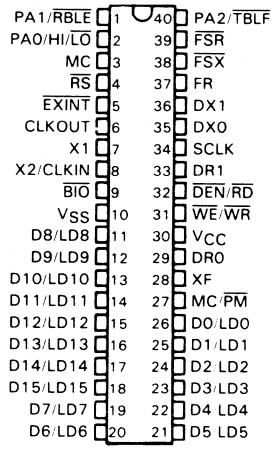
2.1 TMS320C1x Pinouts

Figure 2-1 shows pinouts of the DIP packages for the TMS320C1x devices and the PLCC package for the TMS320C10.

TMS32010, TMS320C10
TMS320C15, TMS320E15
N/JD PACKAGE
(TOP VIEW)



TMS320C17, TMS320E17
N/JD PACKAGE
(TOP VIEW)



TMS320C10, TMS320C15
FN PACKAGE
(TOP VIEW)

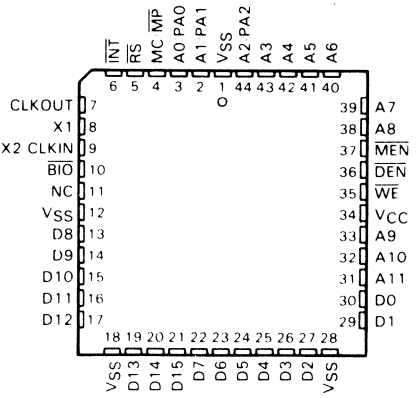


Figure 2-1. TMS320C1x Pin Assignments

2.2 TMS32010/C10/C15/E15 Signal Descriptions

The signal descriptions for the TMS32010/C10 and TMS320C15/E15 devices are provided in this section. Table 2-1 lists each signal, its pin location (DIP/PLCC), function, and operating mode(s), i.e., input, output, or high-impedance state as indicated by I, O, or Z. The signals in Table 2-1 are grouped according to function and alphabetized within that grouping.

Table 2-1. TMS32010/C10/C15/E15 Signal Descriptions

SIGNAL	PIN (DIP/PLCC)	I/O/Z†	DESCRIPTION
ADDRESS/DATA BUSES			
A11 MSB A10 A9 A8 A7 A6 A5 A4 A3 A2/PA2 A1/PA1 A0/PA0	27/31 28/32 29/33 34/38 35/39 36/40 37/41 38/42 39/43 40/44 1/2 2/3	O	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. During execution of the IN and OUT instructions, pins A2 through A0 carry the port addresses PA2 through PA0.
D15 MSB D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 LSB	18/21 17/20 16/19 15/17 14/16 13/15 12/14 11/13 19/22 20/23 21/24 22/25 23/26 24/27 25/29 26/30	I/O/Z	Parallel data bus D15 (MSB) through D0 (LSB). The data bus is always in the high-impedance state except when $\overline{WE}$ is active (low).
INTERRUPT AND MISCELLANEOUS SIGNALS			
$\overline{BIO}$	9/10	I	External polling input. Polled by BIOZ instruction. If low, the device branches to the address specified by the instruction.
$\overline{DEN}$	32/36	O	Data enable for device input data. When active low, $\overline{DEN}$ indicates that the device will accept data from the data bus. $\overline{DEN}$ is only active during the first cycle of the IN instruction. $\overline{MEN}$ and $\overline{WE}$ will always be inactive (high) when $\overline{DEN}$ is active.

† Input/Output/High-impedance state

Table 2-1. TMS32010/C10/C15/E15 Signal Descriptions (Concluded)

SIGNAL	PIN (DIP/PLCC)	I/O/Z†	DESCRIPTION
$\overline{INT}$	5/6	I	External interrupt input. The interrupt signal is generated by applying a negative-going edge to the $\overline{INT}$ pin. The edge is used to latch the interrupt flag register (INTF) until an interrupt is granted by the device. An active low level will also be sensed.
MC/$\overline{MP}$	3/4	I	Memory mode select pin. High selects the microcomputer mode, in which 1.5K words (4K on the TMS320C15/E15) of on-chip program memory are available. This mode also allows an additional 2.5K words of program memory to reside off-chip on the TMS32010/C10. A low on the MC/$\overline{MP}$ pin enables the microprocessor mode. In this mode, the entire memory space is external, i.e., addresses 0 through 4095.
$\overline{MEN}$	33/37	O	Memory enable. $\overline{MEN}$ will be active low on every machine cycle except when $\overline{WE}$ and $\overline{DEN}$ are active. $\overline{MEN}$ is a control signal generated by the device to enable instruction fetches from program memory. $\overline{MEN}$ will be active on instructions fetched from both internal and external memory.
$\overline{RS}$	4/5	I	Reset input for initializing the device. When an active low is placed on the $\overline{RS}$ pin for a minimum of five clock cycles, $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ are forced high, and the data bus (D15 through D0) is not driven. The program counter (PC) and the address bus (A11 through A0) are then synchronously cleared after the next complete clock cycle from the falling edge of $\overline{RS}$. Reset also disables the interrupt, clears the interrupt flag register, and leaves the overflow mode register unchanged. The device can be held in the reset state indefinitely.
$\overline{WE}$	31/35	O	Write enable for device output data. When active low, $\overline{WE}$ indicates that data will be output from the device on the data bus. $\overline{WE}$ is only active during the first cycle of the OUT instruction and the second cycle of the TBLW instruction. $\overline{MEN}$ and $\overline{DEN}$ will always be inactive (high) when $\overline{WE}$ is active.
SUPPLY/OSCILLATOR SIGNALS			
CLKOUT	6/7	O	System clock output (one-fourth crystal/CLKIN frequency). Duty cycle is fifty percent.
V_{CC}	30/34	I	5-V supply pin.
V_{SS}	10/12	I	Ground pin.
X1	7/8	O	Crystal output pin for internal oscillator. If an internal oscillator is not used, this pin should be left unconnected.
X2/CLKIN	8/9	I	Input pin to the internal oscillator (X2) from the crystal. Alternatively, an input pin for the external oscillator (CLKIN).

† Input/Output/High-impedance state

2.3 TMS320C17/E17 Signal Descriptions

Table 2-2 lists each signal provided on the TMS 320C17/E17, its pin location, function, and operating mode(s), i.e., input, output, or high-impedance state as indicated by I, O, or Z. The signals in Table 2-2 are grouped according to function and alphabetized within that grouping.
The first signal and the signal following the slash are both used on the TMS320C17/E17.

Table 2-2. TMS320C17/E17 Signal Descriptions

SIGNAL	PIN (DIP)	I/O/Z†	DESCRIPTION
		BIDIRECTIONAL DATA BUS	
D15/LD15 D14/LD14 D13/LD13 D12/LD12 D11/LD11 D10/LD10 D9/LD9 D8/LD8 D7/LD7 D6/LD6 D5/LD5 D4/LD4 D3/LD3 D2/LD2 D1/LD1 D0/LD0	18 17 16 15 14 13 12 11 19 20 21 22 23 24 25 26	I/O/Z	16-bit parallel data bus D15 through D0. The data bus is always in the high-impedance state except when $\overline{WE}$ is active (low) or when executing an IN instruction from port 0 or port 1. On the TMS320C17/E17, the 16-bit data lines (LD15 through LD0) are used for a coprocessor latch.
		PORT ADDRESS BUS	
PA2/$\overline{TBLF}$ PA1/$\overline{RBLE}$ PA0/HI/$\overline{LO}$	40 1 2	O O I/O/Z	I/O port address output/transmit buffer latch full flag. I/O port address output/receive buffer latch empty flag. I/O port address output/latch byte select pin. During IN and OUT instructions, PA2-PA0 carry the port address. These pins always output the three LSBs of the program counter. On the TMS320C17/E17, these pins are used by the coprocessor latch.
		INTERRUPT AND MISCELLANEOUS SIGNALS	
$\overline{BIO}$	9	I	External polling input. Polled by BIOZ instruction. If low, the device branches to the address specified by the instruction. When in the TMS320C17/E17 coprocessor mode, the $\overline{BIO}$ line is reserved for coprocessor interface and cannot be driven externally.

† Input/Output/High-impedance state

Table 2-2. TMS320C17/E17 Signal Descriptions (Concluded)

SIGNAL	PIN (DIP)	I/O/Z†	DESCRIPTION
$\overline{DEN}/\overline{RD}$	32	I/O/Z	Data enable for device input data/external read for the output latch. When active low, $\overline{DEN}$ indicates that the device will accept data from the data bus. $\overline{DEN}$ is only active during the first cycle of the IN instruction. $\overline{WE}$ will always be inactive (high) when $\overline{DEN}$ is active. In the TMS320C17/E17 coprocessor mode, the external processor reads from the coprocessor latch by driving the $\overline{RD}$ line active (low), thus enabling the output latch to drive the latched data. When the data has been read, the external device will bring the $\overline{RD}$ line high.
$\overline{EXINT}$	5	I	External interrupt input. The interrupt signal is generated by applying a logic low level to the $\overline{EXINT}$ (TMS320C17/E17) pin. The edge is used to latch the system control register flag bit (CR0) until an interrupt is granted by the device. When in the TMS320C17/E17 coprocessor mode, the $\overline{EXINT}$ line is reserved for coprocessor interface and cannot be driven externally.
MC	3	I	On the TMS320C17/E17, the MC pin must be connected to the same state as the MC/$\overline{PM}$ pin. When these pins are low, the coprocessor port is enabled. When these pins are high, the microcomputer mode is enabled.
MC/$\overline{PM}$	27	I	On the TMS320C17/E17, this pin must be connected to the same state as the MC pin. When these pins are low, the coprocessor port is enabled. When these pins are high, the microcomputer mode is enabled.
$\overline{RS}$	4	I	Reset input for initializing the device. When an active low is placed on the $\overline{RS}$ pin for a minimum of five clock cycles, $\overline{DEN}$ and $\overline{WE}$ are forced high, and the data bus (D15 through D0) goes to a high-impedance state. The serial port clock and transmit outputs also go to the high-impedance state. The program counter (PC) and the port address bus (PA2 through PA0) are then synchronously cleared after the next complete clock cycle from the falling edge of $\overline{RS}$.
$\overline{WE}/\overline{WR}$	31	I/O	Write enable for device output data/external write enable for the input latch. When active low, $\overline{WE}$ indicates that data will be output from the device on the data bus. $\overline{WE}$ is only active during the first cycle of the OUT instruction and the second cycle of the TBLW instruction. $\overline{DEN}$ will always be inactive (high) when $\overline{WE}$ is active. In the TMS320C17/E17 coprocessor mode, the external processor lowers the $\overline{WR}$ line and places data on the bus. It next raises the $\overline{WR}$ line to clock the data into the on-chip latch.

† Input/Output/High-impedance state

Table 2-2. TMS320C17/E17 Signal Descriptions (Concluded)

SIGNAL	PIN (DIP)	I/O/Z†	DESCRIPTION
XF	28	O	External logic output flag. Programmable via system control register bit 10 (CR10). This pin is the direct output of the CR10 latch.
SUPPLY/OSCILLATOR SIGNALS			
CLKOUT	6	O	System clock output (one-fourth crystal/CLKIN frequency).
V_{CC}	30	I	5-V supply pin.
V_{SS}	10	I	Ground pin.
X1	7	O	Crystal output pin for internal oscillator. If an internal oscillator is not used, this pin should be left unconnected.
X2/CLKIN	8	I	Input pin to the internal oscillator (X2) from the crystal. Alternatively, an input pin for the external oscillator (CLKIN).
SERIAL PORT SIGNALS			
DR1 DR0	33 29	I	Serial-port receive-channel inputs. Serial data is received in the receive registers via these pins.
DX1 DX0	36 35	O/Z	Serial-port transmit-channel outputs. Serial data is transmitted from the transmit registers on these pins. These outputs are in the high-impedance state when not transmitting.
FR	37	O	Internal serial-port framing output. If internal framing is enabled, serial-port transmit and receive operations occur simultaneously on an active (high) FR framing pulse. Both short and long FR pulses are selectable to provide fixed and variable data-rate framing pulses for combo-codec interface. The FR frequency is derived from the serial-port clock (SCLK) and system control register bits CR23-CR16.
$\overline{FSR}$	39	I	External serial-port receive-framing input. If external framing is enabled via the system control register, data is received via the receive pins (DR1 and DR0) on the active (low) FSR input. The falling edge of FSR initiates the receive process, and the rising edge sets the flag bit (CR1) in the system control register, causing an interrupt to occur if enabled.
$\overline{FSX}$	38	I	External serial-port transmit-framing input. If external framing is enabled, data is transmitted on the transmit pins (DX1,DX0) on the active (low) FSX input. The falling edge of FSX initiates the transmit process, and the rising edge sets the flag bit (CR2) in the system control register, causing an interrupt to occur if enabled.
SCLK	34	I/O/Z	Serial-port clock. Master clock for transmitting and receiving serial-port data. Configurable as an input or output. SCLK must always be present for serial-port operation. As an input, SCLK is the external clock that controls data transfers with the serial port. As an output, SCLK provides the serial clock for data transfers and framing-pulse synchronization. Its frequency is derived from the TMS 320C17/E17 system clock, X2/CLKIN, and system control register bits CR27-CR24. Reset ($\overline{RS}$) forces SCLK to the high-impedance state.

† Input/Output/High-impedance state

3. Architecture

The modified Harvard architecture of the TMS320C1x (first-generation TMS320) microprocessors increases throughput by allowing program fetch to overlap data operations. The hardware-intensive design of these devices provides performance previously unavailable on a single chip. Hardware is used to implement functions that other processors typically perform in software. For example, the TMS320C1x devices contain a hardware multiplier to perform a multiplication in a single instruction cycle. Flexibility is further enhanced by a comprehensive instruction set that supports both general-purpose and digital signal processing applications.

Major topics discussed in this section are listed below.

3.1 Architectural Overview

The TMS320 family utilizes a modified Harvard architecture 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 RAM, eliminating the need for a separate coefficient ROM. It also makes available immediate instructions and subroutines based on computed values.

The TMS320C1x 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 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.

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 fundamental DSP operations such as convolution, correlation, and filtering.

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.

The TMS320C1x devices have 144/256 words of on-chip data RAM and 1.5K/4K words of on-chip program ROM/EPROM to support program development. The EPROM cell utilizes standard PROM programmers and programs identically to a 64K CMOS EPROM (TMS27C64). The TMS320C1x devices are capable of executing programs from up to 4K words of memory at full speed for those applications requiring external program memory space. This allows for external RAM-based systems to provide multiple functionality. The TMS 320C17/E17 do not provide memory expansion capability.

The TMS32010/C10 and TMS320C15/E15 devices offer two modes of operation defined by the state of the MC/$\overline{MP}$ pin: the microcomputer mode (high level) or the microprocessor mode (low level). In the microcomputer mode, on-chip ROM is mapped into the memory space with up to 4K words of

memory available. In the microprocessor mode, all 4K words of memory are external.

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. The interrupts used in these devices are maskable.

The 16-bit parallel data bus can be utilized to perform I/O functions in two cycles. The I/O ports are addressed by the three LSBs on the address lines. In addition, a polling input for bit test and branch operations ($\overline{BIO}$) and an interrupt pin ($\overline{INT}$) have been incorporated for increased system flexibility. Two of the I/O ports on the TMS320C17/E17 are dedicated to the serial port and companding hardware. I/O port 0 is dedicated to control register 0, which controls the serial port, interrupts, and companding hardware.
I/O port 1 accesses control register 1, as well as both serial port channels, and the companding hardware. The six remaining I/O ports are available for external parallel interfaces. On the TMS320C17/E17, port 5 may be used for coprocessor interface.

The TMS 320C17/E17 offers a dual-channel serial port capable of full-duplex serial communication and direct interface to combo-codecs. Receive and transmit registers that operate with 8-bit data samples are I/O-mapped. Either internal or external framing signals for serial data transfers are selected through the system control register. The serial port clock provides the bit timing for transfers with the serial port, and may be either an input or output. A framing pulse signal provides framing pulses for combo-codec circuits, an 8-kHz sample clock for voice-band systems, or a timer for control applications.

On-chip hardware enables the TMS320C17/E17 to compand (COMpress/-exPAND) data in either μ-law (U.S. and Japan) or A-law (European) format. The companding logic operation is configured via the system control register. Data may be companded in either a serial mode for operation on serial port data (converting between linear and logarithmic PCM) or a parallel mode for computation inside the device. The TMS320C17/E17 allows the hardware companding logic to operate with either sign-magnitude or two's-complement numbers.

The coprocessor port on the TMS320C17/E17 provides a direct connection to most 4/8-bit microcomputers and 16/32-bit microprocessors. In the coprocessor mode, the 16-bit parallel port is reconfigured to operate as a 16-bit latched bus interface. Data widths of either 8 or 16 bits may be selected for the coprocessor port, accessed through I/O port 5 using IN and OUT instructions. The coprocessor interface allows the device to act as a peripheral (slave) microcomputer to a microprocessor, or as a master to a peripheral microcomputer. In that mode, the 16 data lines are used for the 6 parallel 16-bit I/O ports.

3.2 Functional Block Diagrams

The functional block diagrams shown in this section outline the principal blocks and data paths within the TMS320C1x processors. Further details of functional blocks are given in the succeeding sections. The two block diagrams also show all the device interface pins for the respective processors.

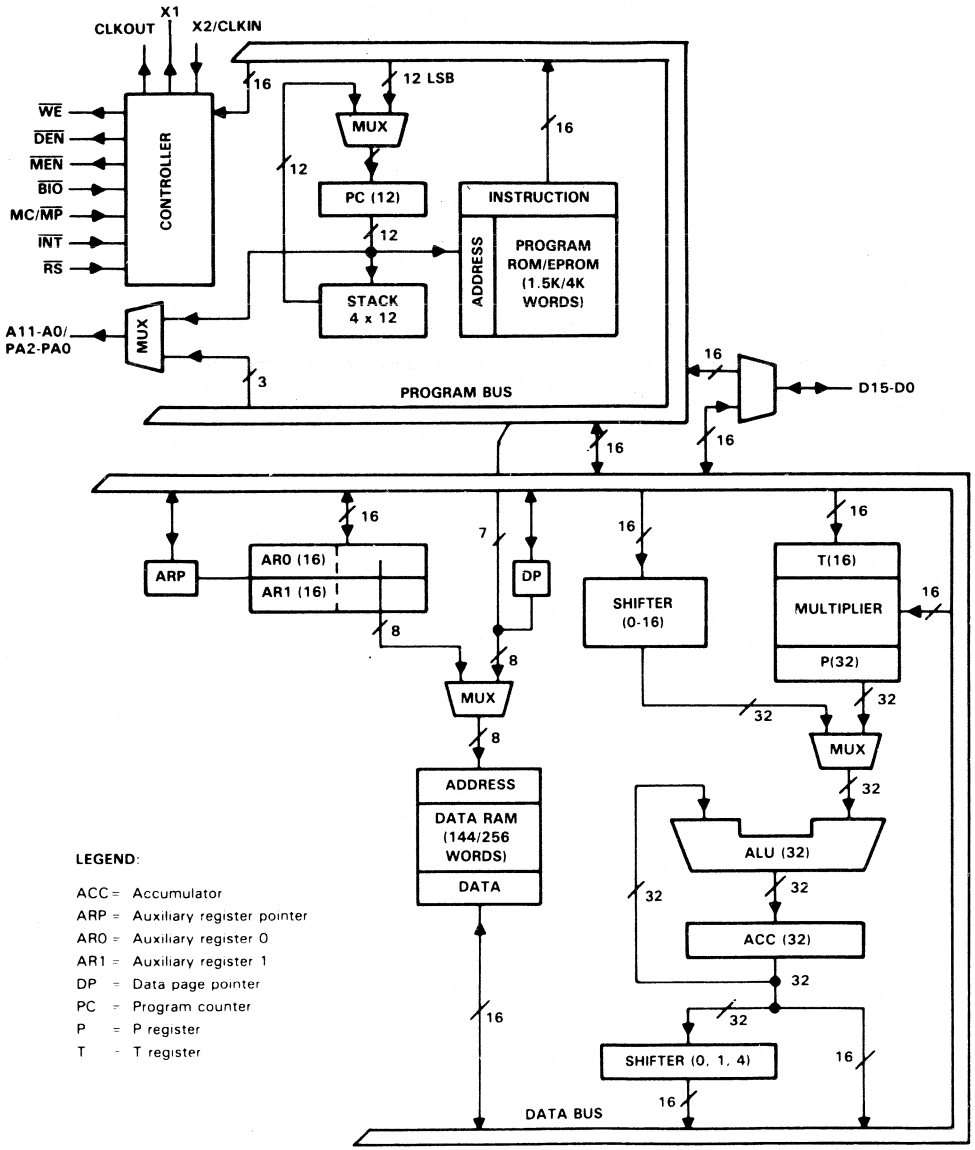
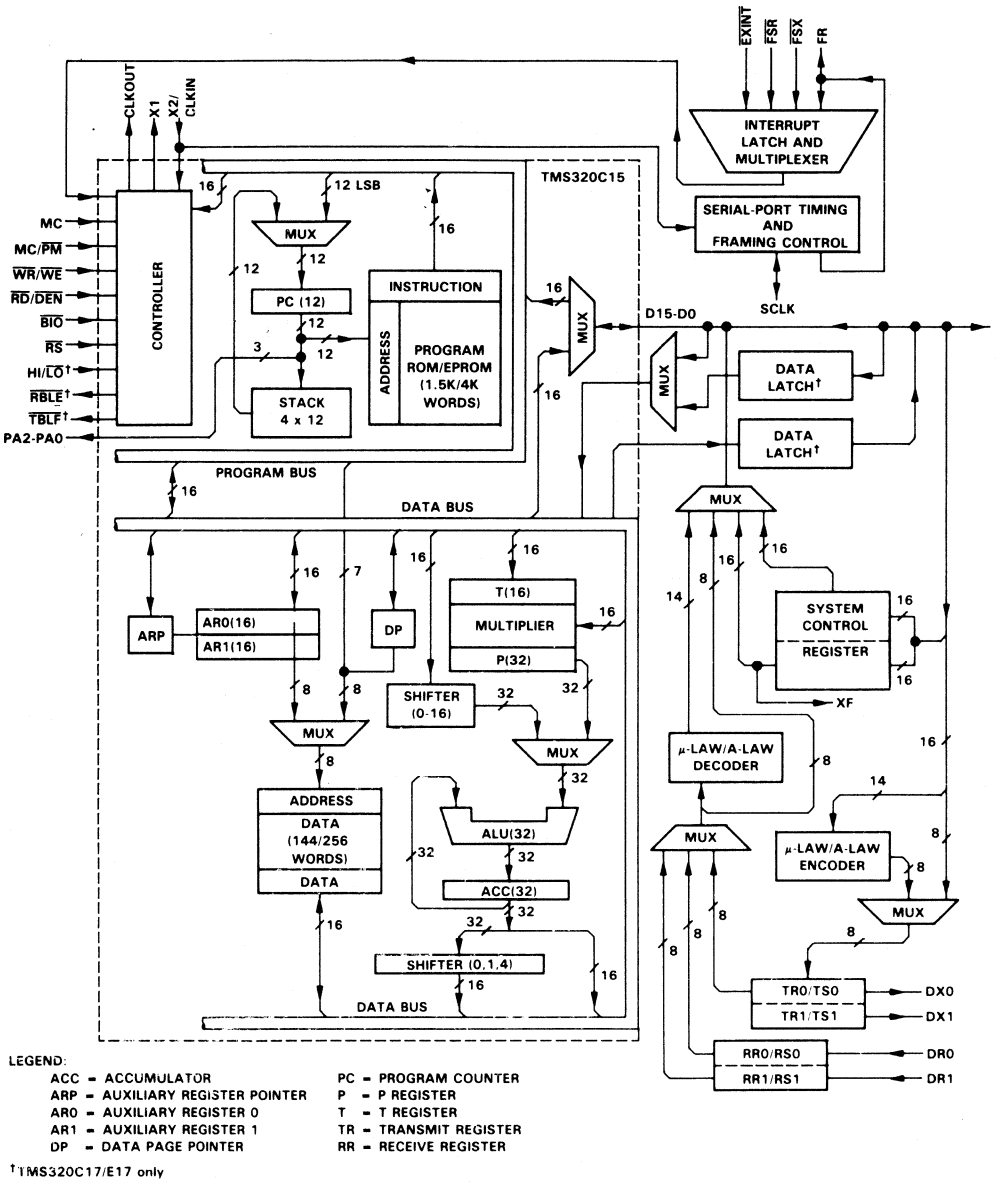


Figure 3-1. TMS32010/C10/C15/E15 Block Diagram



LEGEND:

ACC = ACCUMULATOR	PC = PROGRAM COUNTER
ARP = AUXILIARY REGISTER POINTER	P = P REGISTER
AR0 = AUXILIARY REGISTER 0	T = T REGISTER
AR1 = AUXILIARY REGISTER 1	TR = TRANSMIT REGISTER
DP = DATA PAGE POINTER	RR = RECEIVE REGISTER

†TMS320C17/E17 only

Figure 3-2. TMS320C17/E17 Block Diagram

3.3 Internal Hardware Summary

The TMS320C1x internal hardware implements functions that other processors typically perform in software or microcode. For example, the device contains hardware for single-cycle 16 x 16-bit multiplication, data shifting, and address manipulation. This hardware-intensive approach provides computing power previously unavailable on a single chip.

Table 3-1 presents a summary of the TMS320C1x internal hardware. This summary table, which includes the internal processing elements, registers, and buses, is alphabetized within each functional grouping. All of the symbols used in this table correspond to the symbols used in the block diagrams of Section 3.2, the succeeding block diagrams in this section, and the text throughout this document.

Table 3-1. TMS320C1x Internal Hardware

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 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	-	144 or 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	-	1.5K or 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 hardware stack used to store the PC during interrupts or calls.

Table 3-1. TMS320C1x Internal Hardware (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.
Additional Hardware on the TMS320C17/E17		
Companding Hardware		Data companding encoder/decoder in either μ-law or A-law PCM conversion format. Two modes of operation: serial mode for operating on serial port data (linear/logarithmic PCM conversions), or parallel mode for computation inside the device. Companding is selected through the control register.
Latched Data Bus	LD(15-0)	A 16-bit bidirectional latched data bus used in coprocessor mode. This bus is connected internally to two latches, one for input and one for output.
Serial Port Clock	SCLK	The clock that provides the timing control for data transfers with the serial port. SCLK is configured through the control register.
Serial Port Framing Control	FR	The FR signal provides serial port framing compatible with combo-codec devices. The FR pulse signifies a transmit/receive of new data on the serial port.
Serial Port Receive Registers	RR0,RR1	8-bit serial port registers that receive 8-bit data samples.
Serial Port Receive Shift Registers	RS0,RS1	8-bit registers used to shift in serial port data from pin DR0 or DR1.
Serial Port Transmit Registers	TR0,TR1	8-bit serial port transmit registers in a FIFO (first in, first out) configuration.
Serial Port Transmit Shift Registers	TS0,TS1	8-bit registers used to shift out serial port data onto pin DX0 or DX1.
System Control Register	CR(31-0)	A 32-bit register that controls interrupts, serial port channels, companding hardware, and coprocessor port channels. Control register 1, accessed through port 1, consists of the upper 16 bits (CR31-CR16). Control register 0, accessed through port 0, consists of the lower 16 bits (CR15-CR0).

3.4 Memory Organization

The TMS320C1x devices utilize a Harvard architecture, in which data and program memory reside in two separate spaces. The TMS320C1x provides 144/256 16-bit words of on-chip data RAM and 1.5K/4K words of program ROM. On-chip program EPROM versions are available. This section describes the TMS320C1x data and program memory, data movement, memory maps, auxiliary registers, microcomputer/microprocessor modes, and memory addressing modes.

3.4.1 Data Memory

Data memory consists of 144/256 words of 16-bit on-chip RAM (see Figure 3-3). The TMS32010/C10 provides 144 words. The TMS320C15/C17 offers expanded on-chip RAM of 256 words. See Section 3.4.4 for memory map configurations.

To expand data memory, the data operands may be stored off-chip, and then read into the on-chip RAM as they are needed. Two instruction pairs, TBLR/TBLW and IN/OUT, are available for accomplishing this. The table read (TBLR) instruction can transfer values from program memory, either on-chip ROM or off-chip ROM/RAM, to the on-chip data RAM. The table write (TBLW) instruction transfers values from the data RAM to off-chip program RAM. These instructions take three cycles to execute. When using the IN/OUT instruction pair, the IN instruction reads data from a peripheral and transfers it to the data RAM. With some extra hardware, the IN instruction, together with the OUT instruction, can be used to read and write from the data RAM to large amounts of external storage addressed as a peripheral. This method is faster since IN and OUT instructions take only two cycles to execute. See Section 6.1 for hardware applications using RAM/ROM expansion.

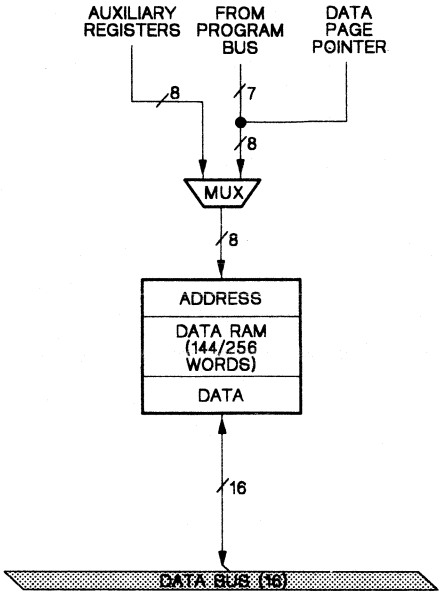


Figure 3-3. On-Chip Data Memory

3.4.2 Program Memory

Program memory consists of 1.5K/4K words on the TMS320C1x devices. The TMS32010/C10 provides 1.5K words, and the TMS320C15/C17 provides 4K words. The on-chip program ROM of up to 4K words allows program execution at full speed without the need for high-speed external program memory. On-chip program EPROM of 4K words, provided on the TMS320E15/E17, presents two additional benefits. First, application development is greatly facilitated since the EPROM can be directly programmed by the user. Second, these devices implement a security feature that can be used to protect proprietary algorithms by preventing the EPROM contents from being read.

Program memory operation is user-selectable by means of the MC/$\overline{\text{MP}}$ (microcomputer/microprocessor) pin. Setting MC/$\overline{\text{MP}}$ high places the device in the microcomputer mode. Holding the pin low places the device in the microprocessor mode.

In the microcomputer mode, only locations 0 through 1523 of the ROM on the TMS32010/C10 are available for the user's program.
Locations 1524-1535 are reserved by Texas Instruments for testing purposes. The device architecture allows for an additional 2.5K words of program memory to reside off-chip on the TMS32010/C10. ROM locations 0 through 3999 on the TMS320C15/C17 are available for the user's program; locations 4000

through 4095 are reserved for testing purposes. Reserved locations may not be utilized by the user. In the microprocessor mode, all 4K words of memory are external. Note that the microprocessor mode is not available for the TMS320C17/E17. See section 3.4.4 for memory map configurations.

External RAM or ROM can be interfaced to the TMS320C1x (see Section 6.1) for those applications requiring external program memory space. This provides multiple functionality for external RAM-based systems. The TMS320C17/E17 provides no direct program memory expansion capability.

Twelve output pins are available for addressing external memory. These pins, A11 (MSB) through A0 (LSB), contain the buffered outputs of the program counter or the I/O port address. When an instruction is fetched from off-chip memory, the $\overline{MEN}$ (memory enable) strobe will be generated to enable the external memory. The instruction word is then transferred to the processor via the data bus (see Section 3.7).

When in the microcomputer mode, the processor selects internal program memory. The $\overline{MEN}$ strobe will still become active in this mode, and the address lines A11 through A0 will still output the current value of the program counter although the instruction word will be read from internal program memory. Note that $\overline{MEN}$ is never active at the same time as the $\overline{WE}$ or $\overline{DEN}$ signals. In effect, $\overline{MEN}$ will go low every clock cycle except when an I/O function is being performed by the IN, OUT, or TBLW instructions. In these multicycle instructions, $\overline{MEN}$ goes low during the clock cycles in which $\overline{WE}$ or $\overline{DEN}$ do not go low.

Figure 3-4 gives an example of external program memory expansion. Even when executing from external memory, the TMS320C1x performs at full speed. Note that some ports are reserved for on-chip peripheral logic.

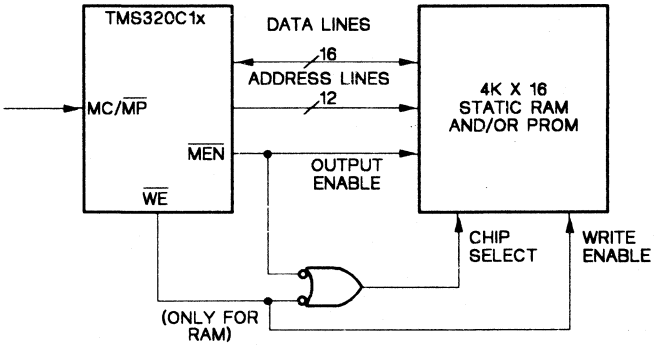


Figure 3-4. External Program Memory Expansion Example

3.4.3 Data Movement

The TMS320C1x provides instructions for data movement functions that efficiently utilize the on-chip RAM. The DMOV (data move) function is useful for implementing algorithms that use the z^{-1} delay operation, such as convolutions and digital filtering where data is being passed through a time window.

Implemented in on-chip RAM, the DMOV function 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 in the same cycle (e.g., by the CALU). The LTD (load T register, accumulate previous product, and move data) instruction uses the data move function.

3.4.4 Memory Maps

The TMS320C1x devices provide three separate address spaces for program memory, data memory, and I/O, as shown in Figure 3-5 and Figure 3-6. Program memory is configured according to the state of the MC/$\overline{MP}$ pin. For further information about data and program memory, see Sections 3.4.1, 3.4.2, and 3.4.3. I/O functions are discussed in Section 3.7.

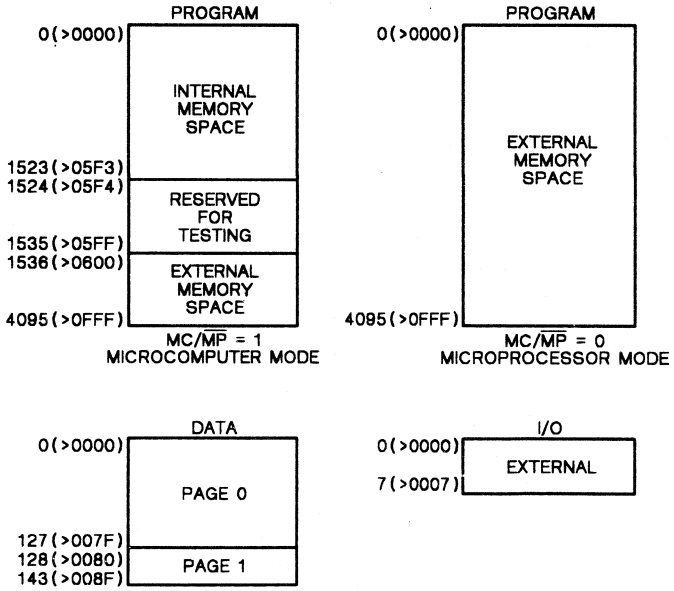
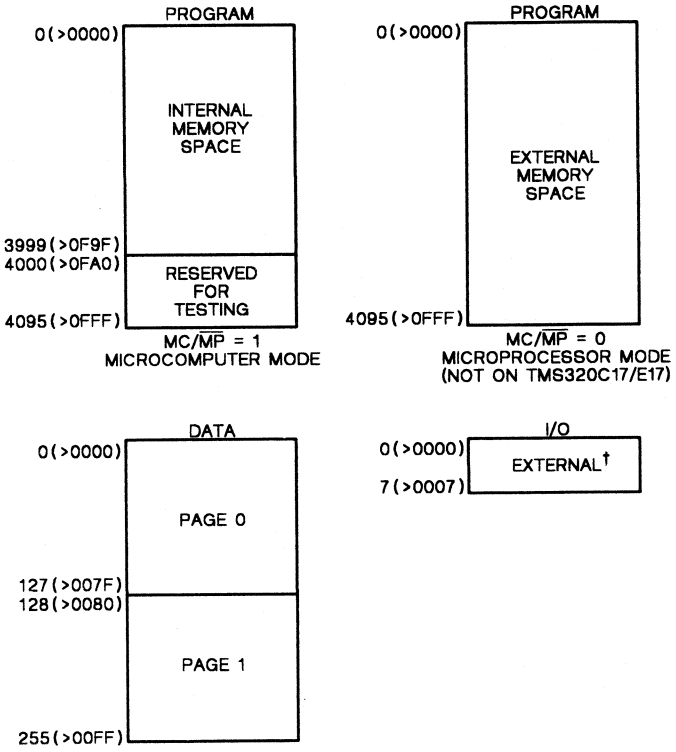


Figure 3-5. Memory Maps for the TMS32010/C10



†On the TMS320C17/E17, ports 0 and 1 are dedicated to the internal control register; no external I/O is available in the coprocessor mode.

Figure 3-6. Memory Maps for the TMS320C15/E15 and TMS320C17/E17

3.4.5 Auxiliary Registers

The TMS320C1x 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 3-7). This counter portion of an auxiliary register is a 9-bit counter, as shown in Figure 3-8 and Figure 3-9.

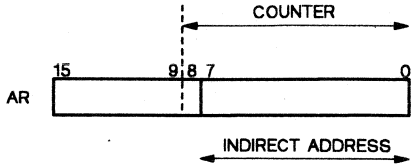


Figure 3-7. Auxiliary Register Counter

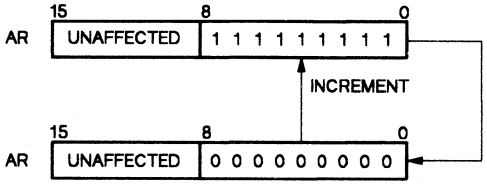


Figure 3-8. Indirect Addressing Autoincrement

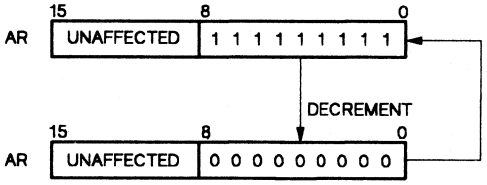


Figure 3-9. Indirect Addressing Autodecrement

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.

3.4.6 Memory Addressing Modes

The TMS320C1x can address up to 4K words of program memory and up to 144/256 words of data memory. Three forms of instruction operand addressing can be used: direct, indirect, and immediate addressing. Figure 3-10 illustrates operand addressing in the three modes. The addressing modes are described in detail in Section 4.1.

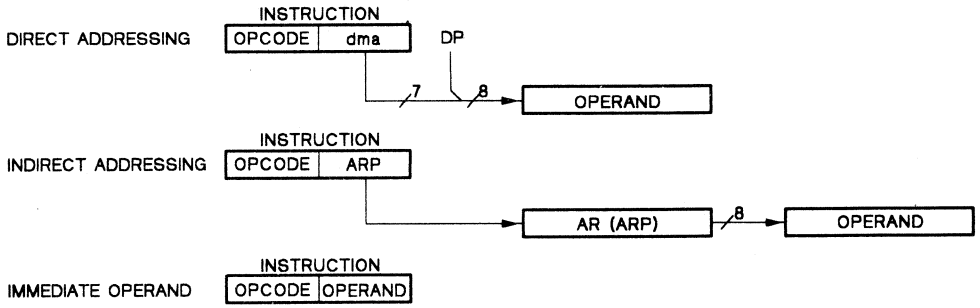


Figure 3-10. Methods of Instruction Operand Addressing

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 128-143/255. 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 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 3-11 is a block diagram showing the components of the CALU.

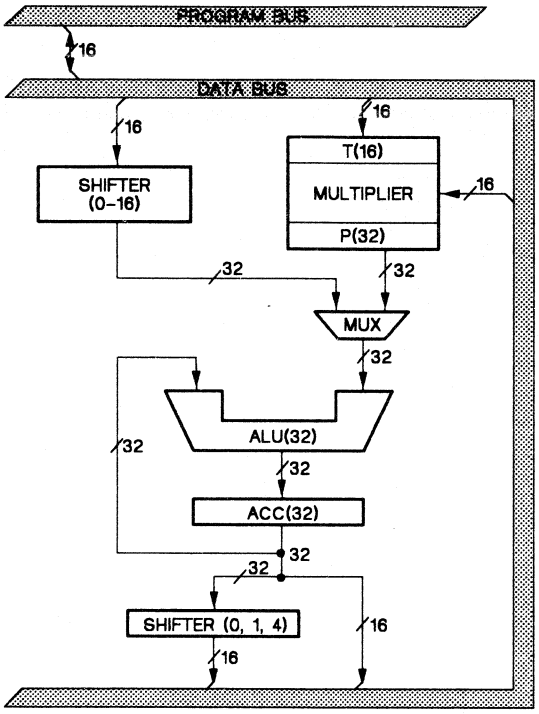


Figure 3-11. Central Arithmetic Logic Unit (CALU)

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, and

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.

3.5.1 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 3-11).

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.

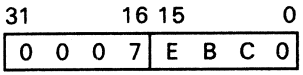
The following exampies illustrate the barrel shifter's function:

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

  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:

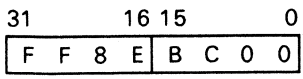
  Since the MSB of >7EBC is a zero, the upper accumulator was zero-filled.

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

  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:

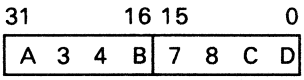
  

Since the MSB of >8EBC is a one, the upper accumulator was filled with ones.

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:



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: >34B7. The accumulator still retains >A34B78CD.

3.5.2 ALU and Accumulator

The 32-bit ALU and accumulator (see Figure 3-11) 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 3-2. 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 3-2. 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. When the low-order word is shifted left, the LSBs are zero-filled, 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 >7FFFFFFF (positive) or >80000000 (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 TMS320C1x 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 TMS320C1x instructions).

3.5.3 Multiplier, T and P Registers

The TMS320C1x utilizes a 16 x 16-bit hardware multiplier (see Figure 3-11), 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 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.

3.6 System Control

System control on the TMS320C1x processors is provided by the program counter and stack, the external reset signal, interrupts (see Section 3.8), and the status register. This section explains the function of these components in system control. In addition to these functions, the TMS320C17/E17 system control register controls the operation of the coprocessor port. The system control register for the TMS320C17/E17 is discussed in Section 3.12.

3.6.1 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 TMS320C1x 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 ($\overline{RS}$) line.

The TMS320C1x 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 3-12 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.

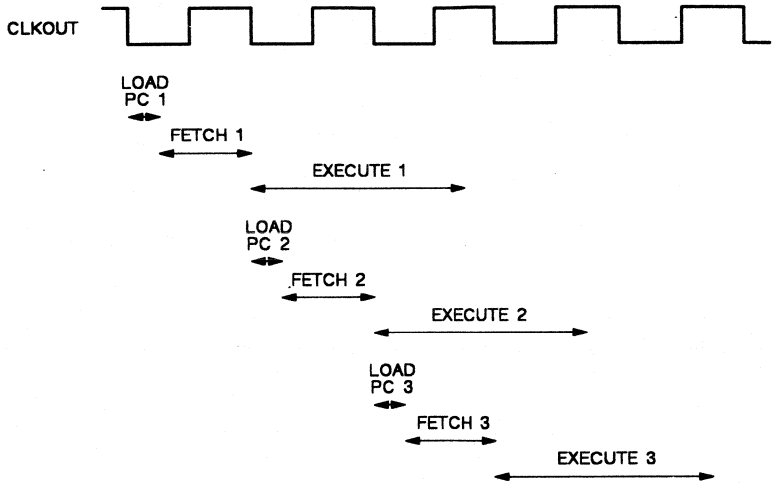


Figure 3-12. Harvard Architecture

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.

3.6.2 Reset

Reset ($\overline{RS}$) is a non-maskable external interrupt that can be used at any time to put the TMS320C1x into a known state. Reset is typically applied after powerup when the machine is in a random state. The reset input must be held low for a minimum of five clock cycles.

Driving the $\overline{RS}$ signal low causes the TMS320C1x to terminate execution and forces the program counter to zero. $\overline{RS}$ affects various registers and status bits. At powerup, the state of the processor is undefined. For correct system operation after powerup, a reset signal must be asserted low to guarantee a reset of the device (see Section 5.1 for other important reset considerations). Processor execution begins at location 0, which normally contains a B (branch) statement to also direct program execution to the system initialization routine (see Section 5.1 for an initialization routine example).

Upon receiving an $\overline{RS}$ signal, the following actions take place:

1) The control lines for $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ are forced high.

2) The data bus D15-D0 is placed in the high-impedance state.

3) The Program Counter (PC) is set to 0, and the address bus A11-A0 is driven with all zeroes after the next clock cycle from $\overline{RS}$ going low.

4) The interrupt is disabled, and the interrupt flag register is reset to all zeroes.

5) Control register bits on the TMS320C17/E17 are set as follows: CR11 is set to 0; CR15 is set to 1; CR29 is set to 0.

The TMS320C1x can be held in the reset state indefinitely. Note that the ARP, DP, and OVM status bits are not initialized by reset. Accordingly, it is critical that these bits be initialized in software by the user following reset.

3.6.3 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 3-3 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 3-3. 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 16 bits. DP = 0 selects the first 128 words of data memory, i.e., page 0. DP = 1 selects page 1, the remaining 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 $\overline{RS}$ interrupt. 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. 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. Thus, an SST instruction using the direct addressing mode can only specify an address less than 16 on the TMS32010/C10 since the second page of memory contains only 16 words. The second page of memory on the TMS320C15/E15 and TMS320C17/E17 contains 128 words. 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 3-13 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 3-13. Status Register Organization

The LST instruction may be executed to load the status register. LST does not assume status bits are on page one, so 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.

3.7 Input/Output Functions

The TMS320C1x implements a variety of different I/O functions for use in communicating with external devices. The 16-bit parallel data bus can be utilized to perform I/O functions in two cycles using the IN and OUT instructions. The I/O ports are addressed by the three LSBs of the address bus (PA2-PA0). In addition, a polling input for bit test and branch operations ($\overline{BIO}$) and an interrupt input ($\overline{INT}$) have been incorporated for increased system flexibility. An external flag output pin (XF) is available on the TMS 320C17/E17 to implement single-bit digital output.

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

Input/output of data to and from a peripheral is accomplished by the IN and OUT instructions. Data is transferred over the 16-bit data bus to and from data memory by two independent strobes: data enable ($\overline{DEN}$) and write enable ($\overline{WE}$).

The bidirectional external data bus is always in the high-impedance state, except when $\overline{WE}$ is active (low), or during an IN instruction from port 0 or port 1 on the TMS 320C17/E17 (see Section 3.7.1). $\overline{WE}$ goes low during the first cycle of the OUT instruction and the second cycle of the TBLW instruction.

Eight I/O addresses are available on the TMS32010/C10 and TMS320C15/E15 for interfacing to peripheral devices: eight 16-bit multiplexed input ports and eight 16-bit multiplexed output ports (see Figure 3-14). Since the system control register, serial port transmit and receive registers, and companding hardware have been mapped into I/O ports 0 and 1, only six input and six output ports are available on the TMS320C17/E17 for interfacing to peripheral devices.

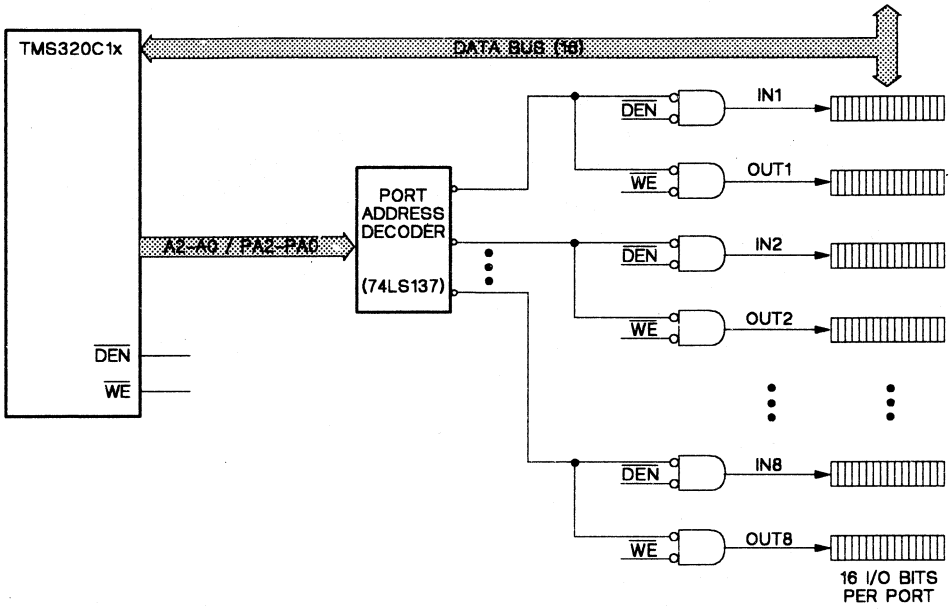


Figure 3-14. TMS320C1x External Device Interface

3.7.1 Input/Output Operation

The three port address pins (PA2-PA0) output the port address during IN and OUT instructions. Execution of an IN instruction generates the $\overline{DEN}$ strobe for transferring data from a peripheral device to the data RAM (see Figure 3-15). The IN instruction is the only instruction for which $\overline{DEN}$ will become active. Execution of an OUT instruction generates the $\overline{WE}$ strobe for transferring data from the data RAM to a peripheral device (see Figure 3-16). $\overline{WE}$ becomes active only during the OUT and TBLW (table write) instructions (see Appendix A for timing information).

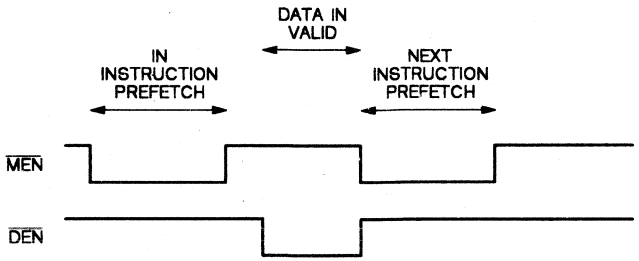


Figure 3-15. Input Instruction Timing

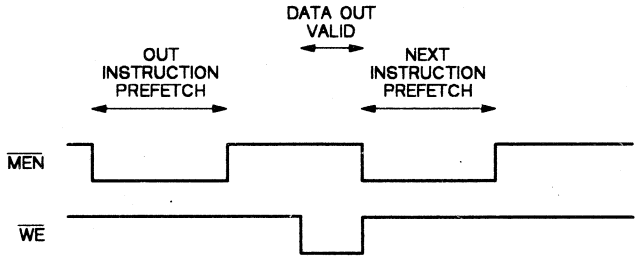


Figure 3-16. Output Instruction Timing

While the three multiplexed LSBs of the address bus (PA2-PA0) are used as a port address by the IN and OUT instructions, the remaining higher-order bits of the address bus (A11 through A3) are held at logic zero during execution of these instructions.

On the TMS320C17/E17, the port address pins PA2-PA0 output the three LSBs of the program counter during IN and OUT instructions (see Section 3.13). These three pins address the serial port, companding hardware, and coprocessor port (TMS320C17/E17). During reset, the pins along with the program counter are synchronously cleared to zero during the cycle following $\overline{RS}$ low. Because all program and data memory are contained on-chip, only these three address lines are output from the device. The memory enable ($\overline{MEN}$) signal is not implemented on the TMS320C17/E17 devices since all instruction execution is from on-chip program ROM.

The bidirectional external data bus on the TMS320C17/E17 is always in the high-impedance state, except when $\overline{WE}$ is active (low) or during an IN instruction from port 0 or port 1. $\overline{WE}$ goes low during the first cycle of the OUT instruction to provide the write strobe for writing data to a peripheral.

On the TMS320C17/E17, the system control register (see Section 3.12), serial port transmit and receive registers (Sections 3.9.1 and 3.9.2), and the companding hardware (Section 3.10) have been mapped into I/O ports 0 and 1. During an OUT or IN instruction to port 0 or port 1, data appears on the external data bus (D15-D0). The data bus is not in the high-impedance state while accessing these dedicated I/O ports. Peripheral device interface should be to port addresses 2 through 7 to prevent bus conflicts with the system control register and serial port. Six 16-bit multiplexed input ports and six 16-bit multiplexed output ports are available for interfacing to peripheral devices.

3.7.2 Table Read/Table Write Operation

The TBLR and TBLW instructions allow words to be transferred between program and data spaces. TBLR is used to read words from on-chip ROM or off-chip program ROM/RAM into the data RAM. TBLW is used to write words from on-chip data RAM to off-chip program RAM on the TMS32010/C10/C15. External program memory cannot be addressed on the TMS320C17/E17.

Execution of the TBLR instruction generates $\overline{\text{MEN}}$ strobes to read the word from program memory (see Figure 3-17). Execution of a TBLW instruction generates a $\overline{\text{WE}}$ strobe (see Figure 3-18). Note that the data bus will be driven and the $\overline{\text{WE}}$ strobe will be generated even if the device is in the microcomputer mode and a TBLW is performed to a program location residing in on-chip ROM.

The dummy prefetch in Figure 3-17 and Figure 3-18 is a prefetch of the instruction following the TBLR or TBLW instruction and is discarded. The instruction following TBLR or TBLW is prefetched again at the end of the TBLR or TBLW instruction.

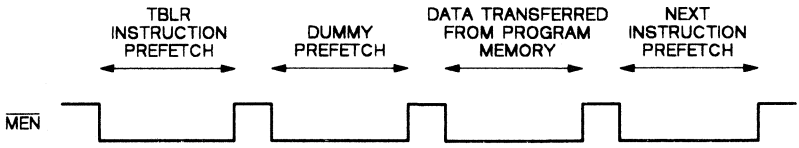


Figure 3-17. TBLR Instruction Timing

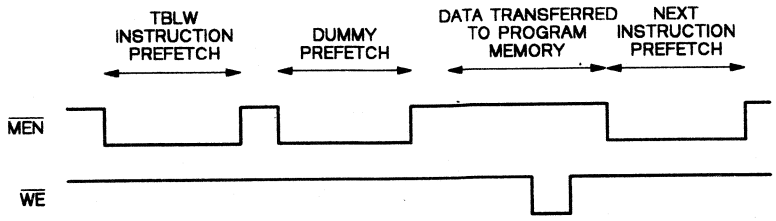


Figure 3-18. TBLW Instruction Timing

The $\overline{MEN}$, $\overline{DEN}$, and $\overline{WE}$ interface stobes are mutually exclusive. There are some very important considerations for those designs that utilize program memory. Since the OUT and TBLW instructions use only the $\overline{WE}$ signal to indicate valid data, these instructions cannot be distinguished from one another on the basis of the interface strobes. Execution of TBLW instructions will write data to peripherals, and execution of OUT instructions will overwrite program memory locations 0 through 7. Since it is impossible to use TBLW to uniquely write to program memory locations 0 through 7, it is advisable to avoid mapping both I/O and external program RAM into locations 0 through 7.

3.7.3 General-Purpose I/O Pins ($\overline{BIO}$ and XF)

The TMS320C1x provides two general-purpose pins which are software-controlled. The $\overline{BIO}$ pin is a branch control input pin for all of the TMS320C1x processors. The XF pin on the TMS320C17/E17 is an external flag output pin.

The $\overline{BIO}$ pin is an external pin that supports bit test and branch operations. When the $\overline{BIO}$ input pin is active (low), execution of the BIOZ instruction causes a branch to occur. The $\overline{BIO}$ pin is useful for monitoring peripheral device status. It is especially useful as an alternative to using an interrupt when time-critical loops must not be disturbed.

For systems using asynchronous inputs to the $\overline{BIO}$ pin on a TMS32010 (NMOS) device, external hardware is required to ensure proper execution of the BIOZ instruction. This hardware synchronizes the $\overline{BIO}$ input signal with the rising edge of CLKOUT on the TMS32010. See Appendix A for information regarding this system design consideration.

The XF (external flag) output pin, specific to the TMS320C17/E17, is an external logic output flag. Programmed through control register bit 10 (CR10), this pin is the direct output of the CR10 latch. When the CR10 bit is set to a 1, the XF pin is set to a logic high; when CR10 is reset to a 0, the XF pin is driven low.

3.8 Interrupts

The TMS320C1x provides an external interrupt input for communication with time-critical external operations. The interrupt can be generated either by applying a negative-going edge or a logic low level to the interrupt input pin, On the TMS320C17/E17, there are also three additional internal interrupts, which are generated by the two serial ports. All interrupts on the TMS320C1x are maskable through the use of the status register interrupt mode bit and various mask bits. When operating in the coprocessor mode on the TMS320C17/E17, the external interrupt and $\overline{BIO}$ will be ignored.

For systems using asynchronous inputs to the interrupt ($\overline{INT}$) pin on a TMS32010 (NMOS) device, external hardware is required to ensure proper processing of interrupts. This hardware synchronizes the $\overline{INT}$ input signal with the rising edge of CLKOUT on the TMS32010. See Appendix A for information regarding this system design consideration.

A simplified diagram of the internal interrupt circuitry for TMS320C1x CMOS devices is shown in Figure 3-19. Note that the TMS32010 requires external synchronizing flip-flops on interrupts and $\overline{BIO}$. These synchronizing flip-flops are not required on the TMS320C10/C15/C17.

When interrupts are enabled, an interrupt becomes active either due to a low-voltage input on the $\overline{INT}$ pin or when a negative edge has been latched into the interrupt flag (INTF). If the interrupt mode register (INTM) is set to zero, an interrupt active signal to the internal interrupt processor becomes valid.

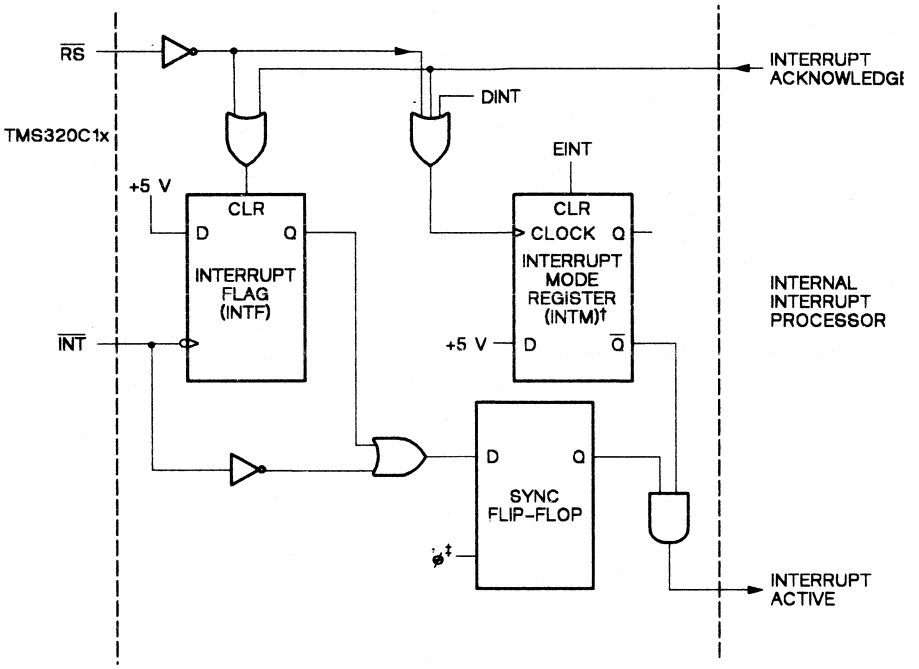
The processor begins interrupt servicing by causing a branch to location 2 in program memory. Interrupt servicing will be delayed in each of the following cases:

1) Until the end of all cycles of a multicycle instruction,

2) Until the instruction following the MPY or MPYK instruction has completed, or

3) Until the instruction following the EINT instruction has been executed (when interrupts have been previously disabled). This allows the RET instruction to be executed after interrupts become enabled at the end of an interrupt routine.

When an interrupt service routine begins, the TMS320C1x transmits an interrupt acknowledge signal that presets the INTM register (disabling interrupts) and clears the interrupt flag (INTF). A DINT instruction or a hardware reset will also set the INTM register to one (see Figure 3-19), disabling interrupts, while the EINT instruction will clear the INTM register (set to zero). Interrupts will continue to be latched while they are disabled. Note that DINT or EINT do not affect the INTF.



† Q = 0 indicates interrupts enabled.
Q = 1 indicates interrupts disabled.
‡ φ = phase of internal clock.
NOTE: The TMS32010 requires external synchronizing flip-flops.

Figure 3-19. TMS320C1x Simplified Interrupt Logic Diagram

Figure 3-20 shows the instruction sequence that occurs once an interrupt becomes active. The dummy fetch is an instruction that is fetched but not executed. This instruction will be refetched and executed after the interrupt routine is completed.

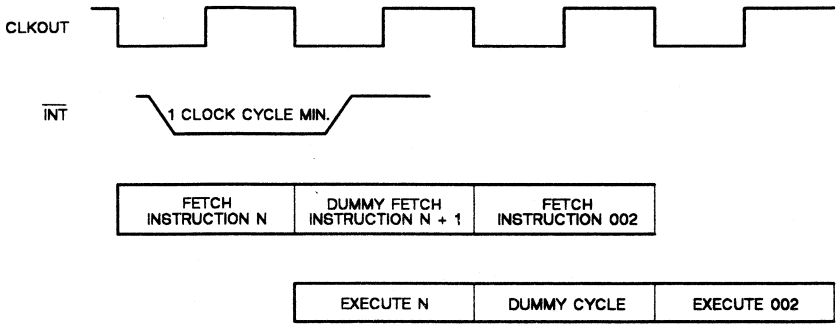


Figure 3-20. Interrupt Timing

The TMS320C17/E17 has four maskable interrupts: $\overline{\text{EXINT}}$ (TMS320C17/E17), $\overline{\text{FSR}}$, $\overline{\text{FSX}}$, and FR. On these devices, the TMS32010/C10/C15 interrupt function has been expanded to fully support the serial-port interface. An interrupt latch and multiplexer is used to generate the master interrupt signal, which functions identically to the $\overline{\text{INT}}$ interrupt on the TMS32010. Thus, all the maskable interrupts have the same priority and require the use of interrupt polling techniques when multiple interrupts are enabled.

Two steps must be taken to enable an active interrupt to the device. First, the individual interrupt must be enabled by writing a logic 1 to the appropriate system control register bit (CR7-CR4). Then, the master interrupt circuitry is enabled via the EINT instruction. In a reset initialization routine, the interrupt flag bits (CR3-CR0) should be cleared before the EINT instruction to insure that a false interrupt does not occur (see Section 3.12 for detailed interrupt bit descriptions).

The interrupt latch synchronizes all interrupts to the device output clock (CLKOUT). A block diagram of the interrupt latch and multiplexer is shown in Figure 3-21. The external interrupt ($\overline{\text{EXINT}}$) is either an asynchronous input to the device for external control or a master processor interrupt signal. The other three interrupts are all associated with the serial port framing signals, although the external framing pulse interrupts ($\overline{\text{FSX}}$ and $\overline{\text{FSR}}$) may be used as system interrupts when not being utilized by the serial port.

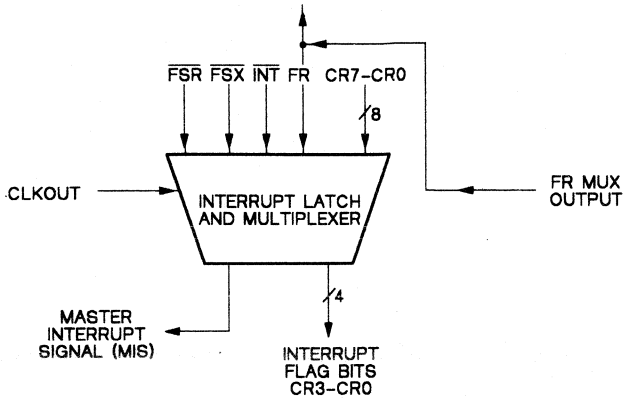


Figure 3-21. Interrupt Latch and Multiplexer

Due to the asynchronous operation of the interrupts, the time between the occurrence of an active interrupt signal and the device actually vectoring to ROM location 2 is four CLKOUT cycles (see Appendix A for further timing information).

3.9 Serial Port (TMS320C17/E17)

Two of the I/O ports on the TMS320C17/E17 are dedicated to the serial port and companding hardware. I/O port 0 is dedicated to control register 0, which controls the serial port, interrupts, and companding hardware. I/O port 1 accesses control register 1, as well as both serial port channels, and the companding hardware. The six remaining I/O ports are available for external parallel interfaces.

The on-chip dual-channel serial port, provided on the TMS320C17/E17, is capable of full-duplex serial communications and direct interface to combo-codec PCM systems, serial A/D converters, and other serial systems. The interface signals are directly compatible with codecs and many other serial devices, and require a minimum of external hardware. An example of a codec interface is provided in Section 6.2. For additional information on combo-codecs, refer to the *TCM29C13/C14/C16/C17 Combined Single-Chip PCM Codec and Filter Data Sheet*.

Two receive and two transmit registers are mapped into I/O port 1, and operate with 8-bit data samples. Either internal or external framing signals for serial data transfers (MSB first) are selected via the system control register. The serial port clock, SCLK, provides the bit timing for transfers with the serial port, and may be either an input or output. A framing pulse signal provides framing pulses for combo-codec circuits, a sample clock for voice-band systems, or a timer for control applications. The serial port is accessed through IN and OUT instructions. A block diagram of the serial port and companding hardware is shown in Figure 3-22.

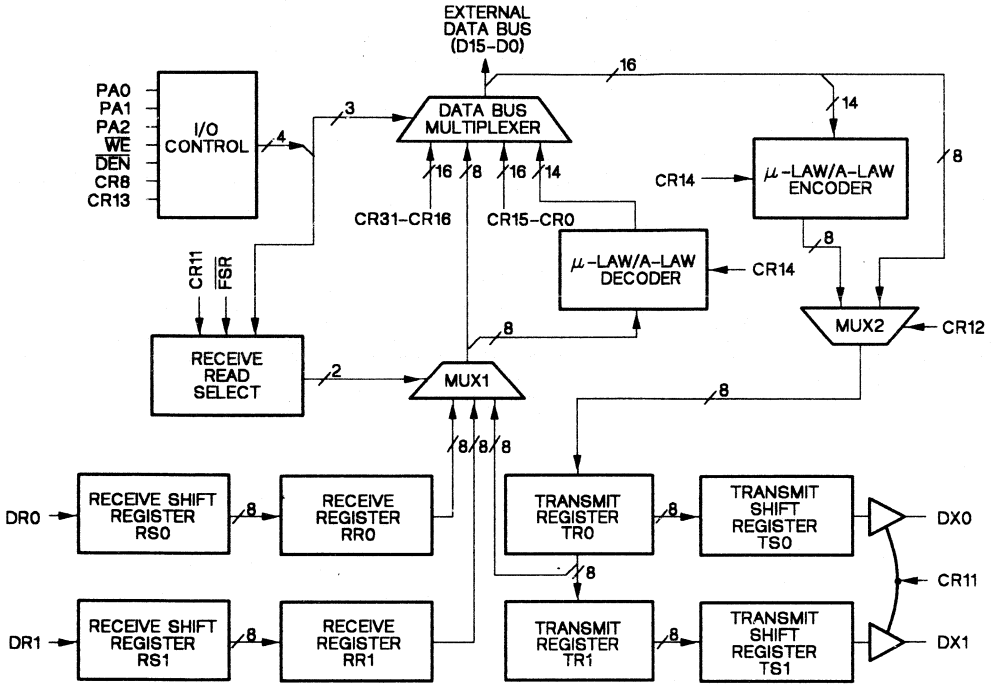


Figure 3-22. Serial Port and Companding Hardware

3.9.1 Receive Registers

Two receive registers are mapped into I/O port 1 via the port decode logic. Data is clocked into the shift registers on the next eight negative serial clock (SCLK) transitions after an active framing pulse is detected. SCLK controls the bit-level timing for all serial-port data transfers. Note that the MSB is always shifted first.

On an active framing pulse, serial data is clocked into the receive registers from the DR pins. Channel 0 data is received in shift register RS0 from pin DR0, and channel 1 data is received in shift register RS1 from pin DR1. To read the data from the registers, an IN instruction is executed from port 1. On the first IN instruction after a framing pulse, channel 0 data is output onto the external data bus. On the second IN instruction, channel 1 data is output onto the external data bus.

An active framing pulse initiates the receive operation, as shown in Figure 3-23. External framing pulses ($\overline{FSR}$) are active low, and the internal framing (FR) signal is active high. With external framing ($\overline{FSR}$), the falling edge of the framing pulse gates the serial-port clock to the receive shift registers, and the data is clocked into the shift registers on the next eight consecutive negative

transitions of the clock. The rising edge of the framing pulse transfers the data from the receive shift registers to the receive registers and sets the $\overline{FSR}$ flag bit (CR1) in the system control register, causing an interrupt to occur if the $\overline{FSR}$ is enabled.

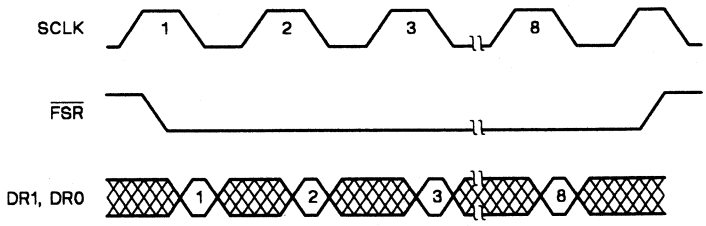


Figure 3-23. Receive Timing for External Framing

Internal framing (FR) pulses can be selected in either fixed data-rate or variable data-rate modes for combo-codec interface. With the fixed data-rate mode, the FR pulse is one SCLK cycle wide, and appears in the cycle preceding the first data bit. The falling edge of the pulse initiates both the transmit and receive operations, as shown in Figure 3-24. Received data is clocked into the receive shift registers on the next eight consecutive negative transitions of the clock. After data bit 8 has been received, data is transferred from the receive shift registers to the receive registers, and an interrupt is generated when the FR flag bit (CR3) is set in the system control register, thus causing an interrupt to occur if enabled.

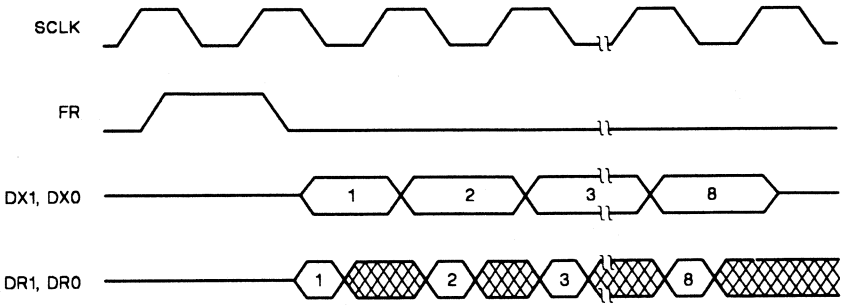


Figure 3-24. Fixed-Data Rate for Internal Framing

In the variable data-rate mode shown in Figure 3-25, the FR pulse is eight SCLK cycles wide, and appears in the same SCLK cycle as the first data bit. The rising edge of the pulse initiates the transmit and receive operations. The falling edge of the pulse transfers data from the receive shift registers to the receive registers and sets the FR flag bit (CR3) in the system control register, causing an interrupt to occur if enabled.

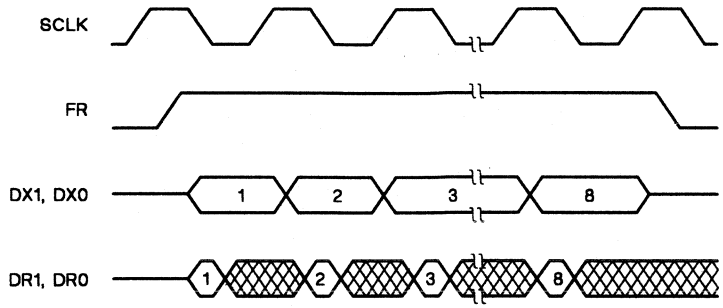


Figure 3-25. Variable-Data Rate for Internal Framing

3.9.2 Transmit Registers

Two transmit registers are mapped into I/O port 1 via the port decode logic. The transmit registers are connected to the port 1 data bus in a FIFO (first in, first out) configuration. On the first OUT instruction to port 1 after a framing pulse, the data to be transmitted is put into transmit register TR0. On the next framing pulse, the TR0 contents are latched into transmit shift register TS0 and the data is transmitted on channel 0 (pin DX0) on the next eight positive transitions of the serial-port clock (SCLK), as shown in Figure 3-26. Data sent to port 1 is always put into the transmit registers. Only when control register bit 11 (CR11) is high will the data be enabled onto the transmit pins. The transmit pins are in the high-impedance state when not transmitting.

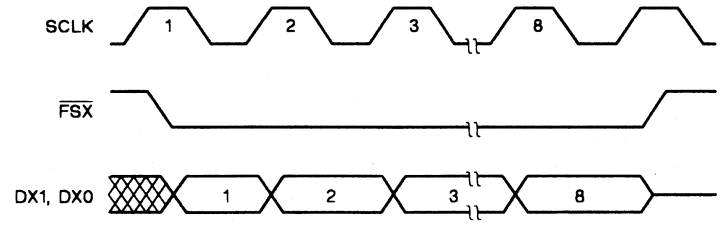


Figure 3-26. Transmit Timing for External Framing

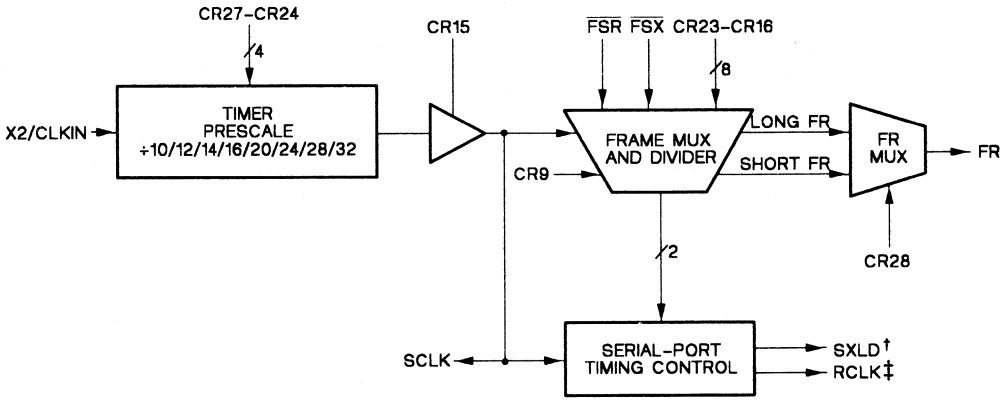
When two OUT instructions to port 1 are executed between framing pulses, both transmit registers are loaded with data for transmission. The first OUT instruction loads data into transmit register TR0. The second OUT pushes the data from TR0 into TR1 and puts the new data into TR0. On an active framing pulse edge, the transmit register contents are latched into the transmit shift registers and the data clocked out on the next eight consecutive positive transitions of SCLK. Thus, for single-channel operation, only one OUT instruction to port 1 should be executed between framing pulses to insure data

transmission on channel 0. Only TR0 may be read back to the serial-port data bus by an IN instruction. This feature is used for the parallel companding mode.

Both transmit channels always output data on an active framing pulse when CR11 is high. During single-channel operation (using channel 0), channel 1 still transmits the data from transmit register TR1. Transmit channel 1 cannot be disabled during single-channel operation.

3.9.3 Timing and Framing Control

The serial-port timing and framing control is shown in Figure 3-27. The serial-port clock (SCLK) provides the timing control for data transfers with the serial port. SCLK may be configured as either an input or output through the control register. As an input, SCLK is an external serial system clock that provides the framing synchronization and timing for the serial port. As an output, SCLK provides the system clock for standalone serial applications and is derived from the microcomputer system clock (X2/CLKIN).



†SXLD = Load transmit shift registers (TS0,TS1) from transmit registers (TR0,TR1)
‡RCLK = Load receive registers (RR0,RR1) from receive shift registers (RS0,RS1)

Figure 3-27. Serial-Port Timing and Framing Control

The serial-port clock prescaler determines the divide ratio for SCLK when configured as an output. The TMS 320C17/E17 system clock (X2/CLKIN) is input to the prescaler, along with control register bits CR27-CR24. Table 3-4 shows the prescale divide ratios selectable as divide by 10, 12, 14, 16, 20, 24, 28, and 32 through system control register bits CR27-CR24.

Table 3-4. Serial Clock (SCLK) Divide Ratios (X2/CLKIN = 20.48 MHz)

CR27	CR26	CR25	CR24	DIVIDE RATIO	SCLK FREQUENCY	UNIT
0	0	0	0	32	0.640	MHz
0	0	0	1	28	0.731	MHz
0	0	1	0	24	0.853	MHz
0	1	0	0	20	1.024	MHz
1	0	0	0	16	1.280	MHz
1	0	0	1	14	1.463	MHz
1	0	1	0	12	1.706	MHz
1	1	0	0	10	2.048	MHz

The divide ratios are available only for SCLK when it is configured as an output from the device (see Section 3.12 for control register bit configurations). These bits determine the divide ratio, which is equal to SCLK/(CNT + 2) where CNT is the binary value of CR23-CR16.

The frame multiplexer determines which framing pulses cause serial-port data transfers to occur and configures the internal framing pulse (FR) frequency. The inputs to the multiplexer are SCLK, control register bit 9 (CR9), control register bits CR23-CR16, external transmit framing ($\overline{FSX}$) pulse, and external receive framing ($\overline{FSR}$) pulse. The outputs of the multiplexer go to the serial-port control for receive and transmit timing generation for the serial-port registers and to the FR multiplexer for determining which FR framing pulse will be generated.

The outputs of the frame counter are input to the FR multiplexer for selection of long or short FR pulses. The short FR pulse provides fixed data-rate framing pulses for standalone serial interface to the Texas Instruments TCM29Cxx family of combo-codec circuits. The long FR framing pulse provides variable data-rate framing pulses to the combo-codec.

The FR frequency is determined at the beginning of the framing pulse cycle. When reconfiguring the frequency, the upper control register bits determine the new divide ratio. However, the new frequency is not implemented until the next FR framing pulse.

3.10 Companding Hardware (TMS320C17/E17)

The on-chip companding hardware enables the TMS320C17/E17 to compand (COMpress and exPAND) data in either μ-law or A-law format with either sign-magnitude or two's-complement numbers.
The standard employed in the United States and Japan is μ-law companding. The European standard is referred to as A-law companding. Configuration and connections of the encoder and decoder (see Figure 3-22) are controlled through the system control register.

No bias is required when operating in two's-complement notation and for A-law companding. For μ-law encoding, a bias of 33 must be added to the sign magnitude before encoding; likewise, after μ-law decoding, the bias of 33 must be subtracted from the sign magnitude. Upon reset, the TMS320C17/E17 is programmed to operate in sign-magnitude mode. This mode can be changed by modifying control register bit 29 (CR29). For further information on companding, see the *TCM29C13/TCM29C14/TCM29C16/TCM29C17 Combined Single-Chip PCM Codec and Filter Data Sheet*. If the companding hardware is not used but software companding is desired, the application report, "Companding Routines for the TMS32010/TMS32020," in the book, *Digital Signal Processing Applications with the TMS320 Family*, describes algorithms that accomplish this.

The specification for μ-law and A-law log PCM is part of the CCITT G.711 recommendation. Part of the coding format specifies certain bits to be inverted prior to transmission or upon receipt of transmitted data. For the μ-law format, all of the data bits are inverted. Refer to the data sheet in Appendix A for diagrams of the codec interface circuits used for μ-law and A-law formats on the TMS320C17/E17 devices.

Data may be companded via four modes: serial-port encode, serial-port decode, parallel encode, and parallel decode. In the serial mode, transmitted data is encoded according to the specified companding law, and received data is decoded to sign-magnitude format. In the parallel modes, encoding or decoding is performed on data from the RAM for computations within the device.

Table 3-5 shows the control register bit combinations that determine the serial or parallel modes of the companding hardware operation. Note that the serial and parallel companding modes require separate control register settings. When using the serial mode, parallel companding is not available unless the control register is reconfigured.

Table 3-5. Serial- and Parallel-Mode Bit Configurations

CR BIT # 13	12	11	MODE OF OPERATION
0	0	0	Parallel mode. Encoder and decoder are disabled. No operation performed on data written to or read from port 1.
0	0	1	Serial mode. Encoder and decoder are disabled. The transmit registers are enabled for data transmission on an active framing pulse. The 8-bit value written to port 1 is transmitted and the 8-bit value in the receive register is read with an IN instruction from port 1.
0	1	0	Parallel encode. Encoder is enabled. A linear sample written to port 1 with an OUT instruction is compressed to 8-bit log PCM. The 8-bit value is then read from port 1 with an IN instruction.
0	1	1	Serial encode. Encoder is enabled. A linear sample written to port 1 is compressed to 8-bit log PCM and put into the transmit register for transmission on an active framing pulse.
1	0	0	Parallel decode. Decoder is enabled. An 8-bit log PCM data written to port 1 is decoded to linear notation with an IN instruction from port 1.
1	0	1	Serial decode. Decoder is enabled. An 8-bit log PCM sample from one of the receive registers is expanded to linear notation with an IN instruction from port 1.
1	1	0	Parallel encode and decode. Encoder and decoder enabled. This is not a usual state, since data is compressed on an OUT instruction to port 1 and then expanded with the IN instruction from the port.
1	1	1	Serial encode and decode. Encoder and decoder enabled. Linear data written to port 1 is encoded and put into one of the transmit registers for serial transmission. The 8-bit log PCM data from one of the receive registers is decoded with an IN instruction from port 1.

3.10.1 μ-Law/A-Law Encoder

The encoder compresses linear PCM (13 bits of dynamic range for μ-law format or 12 bits of dynamic range for A-law format) to 8-bit logarithmic PCM. Selection between μ-law or A-law conversion is determined by the system control register bit 14 (CR14). This bit is input directly to the encoder to determine the conversion law to be used. The μ-255 law conversion is performed if CR14 is logic 0, and A-law conversion if CR14 is logic 1. Data is input to the encoder from the data bus with an OUT instruction to port 1. The converted 8-bit log PCM sample is then presented to the multiplexer (MUX2 shown in Figure 3-22). The multiplexer controls whether the encoder output or the eight low-order data bus bits are input to transmit register TR0 of the serial port. Note that the transmit registers are connected to the port 1 data bus in a FIFO (first in, first out) configuration. The encoder compresses data written to port 1 at all times, but the output will be enabled to the TR0 only when CR12 is logic 1.

In the serial-encode mode, data written to port 1 is encoded, and the value put into transmit register TR0. The transmit register is then loaded with the 8-bit value on an active framing pulse, and the 8 bits are clocked out on the positive edge of SCLK.

For the parallel-encode mode, the linear-PCM value is written to port 1 with an OUT instruction. The encoded 8-bit value is then stored in TR0. An IN instruction from port 1 reads TR0 to the data bus for storage in RAM. Care should be taken to have only one OUT and one IN instruction to port 1 for each data sample in the parallel-encode mode. If there are two OUT instructions to port 1, the first sample will be pushed into transmit register TR1, which cannot be read back to the data bus.

3.10.2 μ-Law/A-Law Decoder

The μ-law/A-law decoder converts 8-bit log-PCM samples to linear PCM. The conversion-law selection is governed by control register bit 14 (CR14). The μ-law conversion is performed if CR14 is logic 0, and A-law conversion if CR14 is logic 1. Data input to the decoder may come from either the serial-port receive registers or transmit register TR0. The multiplexer (MUX1 shown in Figure 3-22) sends data to the data bus either through the decoder or directly to the bus. This multiplexer is controlled in part by control register bit 13 (CR13). If this bit is logic 0, the multiplexer output is sent to the data bus directly. If the bit is logic 1, the multiplexer output is sent to the data bus through the decoder.

In the serial-decode mode, received data from the serial-port receive registers is input to the decoder from the multiplexer, and the received data is decoded according to either μ-law or A-law format.

For the parallel-decode mode, the 8-bit PCM sample to be decoded is written to port 1 with an OUT instruction. This stores the sample in transmit register TR0. The sample is then decoded by reading the value from port 1 with an IN instruction. The IN instruction brings the sample from TR0 through the multiplexer (MUX1) to the decoder, which performs the expansion on the 8-bit sample. Again, there should be only one OUT and one IN instruction to port 1 for each sample to be decoded in order to avoid losing a sample in transmit register TR1.

3.11 Coprocessor Port (TMS320C17/E17)

The coprocessor port on the TMS320C17/E17 provides a direct interface to most 4/8-bit microcomputers and 16/32-bit microprocessors. The port is accessed through I/O port 5 using IN and OUT instructions. The coprocessor interface allows the device to act as a peripheral (slave) microcomputer to a microprocessor, or as a master to a peripheral microcomputer such as the TMS7042. The coprocessor port is enabled by setting MC/$\overline{PM}$ and MC low. The microcomputer mode is enabled by setting these two pins high. (Note that the MC/$\overline{PM}$ and MC pins must be in the same state.)
The 16 data lines are then used for the 6 parallel 16-bit I/O ports.

In coprocessor mode, the 16-bit data bus is reconfigured to operate as a 16-bit latched bus interface. Control bit 30 (CR30) in control register 1 is used to configure the coprocessor port to either an 8-bit or a 16-bit length for data transfer. When CR30 is high, the coprocessor port is 16 bits wide, thereby making all 16 bits of the data port available for 16-bit transfers to 16/32-bit microprocessors. When CR30 is low, the port is 8 bits wide and mapped to the low byte of the data port for interfacing to 4/8-bit microcomputers. When operating in the 8-bit mode, both halves of the 16-bit latch can be addressed by the external device using the HI/$\overline{LO}$ pin, thus allowing 16-bit transfers over 8 data lines. This requires two external bus cycles but only one internal port access. When not in the coprocessor mode, port 5 can be used as a generic I/O port.

Interprocessor communication through the coprocessor interface is accomplished asynchronously as in memory-mapped I/O operations. For a write to the TMS320C17/E17, the external processor lowers the $\overline{WR}$ line and places data on the bus (see Figure 3-28). It then raises the $\overline{WR}$ line to clock the data into the on-chip latch. The falling edge of $\overline{WR}$ clears the $\overline{RBLE}$ (receive buffer latch empty) flag, and the rising edge of $\overline{WR}$ automatically creates an interrupt to the TMS320C17. (Note that when reading or writing in the 8-bit mode, accesses to the high byte will not activate an interrupt or $\overline{BIO}$.) The external processor reads from the latch by driving the $\overline{RD}$ line active low, thus enabling the output latch to drive the latched data (see Figure 3-29). When the data has been read, the external device will again bring the $\overline{RD}$ line high. This activates the $\overline{BIO}$ line to signal that the transfer is complete and the latch is available for the next transfer. The falling edge of $\overline{RD}$ resets the $\overline{TBLF}$ (transmit buffer latch full) flag. Note that the $\overline{EXINT}$ and $\overline{BIO}$ lines are reserved for coprocessor interface and cannot be driven externally when in the coprocessor mode.

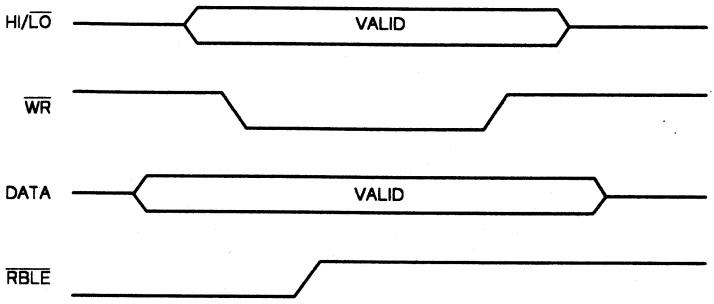


Figure 3-28. External Write Timing to the Coprocessor Port

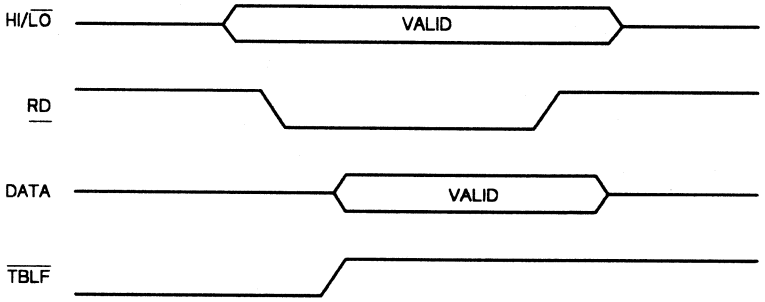


Figure 3-29. External Read Timing from the Coprocessor Port

Examples of the use of a coprocessor interface are provided in Section 6.5 and the data sheet of Appendix A.

3.12 System Control Register (TMS320C17/E17)

The TMS320C17/E17 provides additional hardware for interfacing ease in serial applications. This hardware is interfaced to the micro-computer portion of the device via the external data bus (D15-D0). The additional hardware is controlled by a 32-bit system control register (see Figure 3-30), thereby eliminating any additions to the TMS320 instruction set.

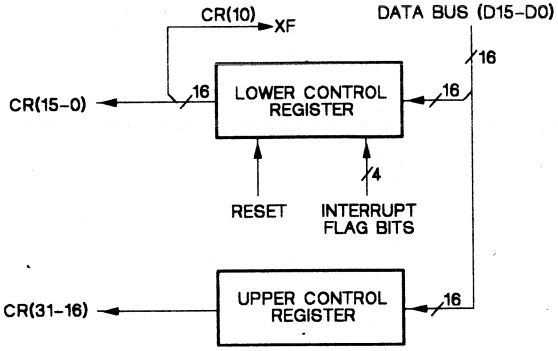


Figure 3-30. System Control Register

The lower 16 register bits (CR15-CR0) are accessed through port 0. These bits control interrupts, serial-port configuration, the external logic output flag, internal and external framing pulses, and the μ-law/A-law encoder and decoder. The interrupt inputs ($\overline{INT}$, $\overline{FSX}$, $\overline{FSR}$, and FR) are synchronized to CLKOUT and control the interrupt flag bits (CR3-CR0). The interrupts are maskable via the interrupt enable bits (CR7-CR4). Bit 8 (CR8) controls I/O port 1 configuration.

The upper 16 bits (CR31-CR16) are accessed through port 1. These bits control the internal framing pulse (FR) output frequency, serial-clock divide ratios, pulse-width control for the FR framing pulse, and companding conversions. On the TMS320C17/E17, the bit width of the coprocessor mode is controlled by CR30.

The external data bus provides on-chip communication with the system control register, serial port, companding hardware, and coprocessor port. With a write to port 0, the lower control register is addressed and data latched into the register by the rising edge of the write enable ($\overline{WE}$) signal. To write to the upper control register bits, bit 8 of the lower control register must be set to logic 1. If CR8 is logic 0, a write to port 1 accesses the serial port and companding hardware.

Table 3-6 gives a detailed description of the control register bits and their operation. The control register bits are configured through OUT instructions to port 0 and port 1. $\overline{WE}$ goes low during the first cycle of the OUT instruction, enabling the port data onto the external data bus. The control register bits are latched on the rising edge of $\overline{WE}$. There is a propagation delay time for these

bits to access the appropriate hardware (see Appendix A for timing information). An allowance for this write delay should be made when reconfiguring the control register. The most critical factor is receiving an external framing pulse while reconfiguring the control register. If an external framing pulse is received at that time, it may not be detected and the serial-port registers will contain random data (see Section 3.9 for further details).

Table 3-6. Control Register Bit Definitions

CR BIT #	DESCRIPTION
3-0	Interrupt flags. When an interrupt occurs on any of the four maskable interrupts, the appropriate flag is set to logic 1 whether the interrupt is enabled or disabled. To clear the flag, a logic 1 is written to the appropriate bit by an OUT instruction to port 0. The bits may be read by an IN instruction to determine interrupt sources when multiple interrupts are enabled. Bit # — Flag 0 — $\overline{\text{EXINT}}$ 1 — $\overline{\text{FSR}}$ 2 — $\overline{\text{FSX}}$ 3 — FR
7-4	Interrupt enable bits. When one of these bits is set to logic 1, an interrupt occurring on that input sets the appropriate flag and activates the microcomputer interrupt circuitry. When disabled, the interrupt flag is still set, but the device is not interrupted. Bit # — Flag 4 — $\overline{\text{EXINT}}$ 5 — $\overline{\text{FSR}}$ 6 — $\overline{\text{FSX}}$ 7 — FR
8	Port 1 control bit. When set to logic 0, I/O port 1 is connected to either the serial-port registers or the companding hardware, depending on the state of CR11. When set to logic 1, I/O port 1 is connected to the upper control register. This bit must be set with an OUT instruction to port 0 before port 1 may access the upper control register bits CR31-CR16.
9	External framing enable. This bit controls which framing pulses cause serial port data transmission to occur. When set to logic 0, serial port transmit and receive operations occur simultaneously and are controlled by the internal framing (FR) pulse. When set to logic 1, transmit operations are controlled by the external transmit framing pulse ($\overline{\text{FSX}}$), and receive operations are controlled by the external receive framing pulse ($\overline{\text{FSR}}$).
10	XF output latch. This bit controls the logic level of the external logic output flag (XF) pin. A write delay time occurs when reconfiguring this latch (see Appendix A for timing information).
11	Serial port enable. When set to logic 0, the transmit and receive registers are disabled in order to use the parallel companding mode. When set to logic 1, the serial port registers are enabled and data transfers with the serial port are via OUT and IN instructions to port 1. A reset sets this bit to zero.

Table 3-6. Control Register Bit Definitions (Concluded)

CR BIT #	DESCRIPTION
12	μ-law/A-law encoder enable. When set to logic 0, the encoder is disabled. When set to logic 1, the encoder is enabled, and data written to port 1 is μ-law or A-law encoded. The encoder must be enabled for compression of linear data in both the serial and parallel modes of operation.
13	μ-law/A-law decoder enable. When set to logic 0, the decoder is disabled. When set to logic 1, the decoder is enabled, and data read from port 1 is μ-law or A-law decoded to linear format. The decoder must be enabled for expansion of log PCM data in both the serial and parallel modes of operation.
14	μ-law or A-law encode/decode select. When set to logic 0, the companding hardware performs μ-255-law conversion. When set to logic 1, the companding hardware performs A-law conversion.
15	Serial clock control. When set to logic 0, the serial port clock (SCLK) is an output, and its frequency is derived from the microcomputer system clock, X2/CLKIN. When set to logic 1, SCLK is an input that provides the clock for all data transfers with the serial port and the frame counter in timing logic. A reset sets this bit to one.
23-16	Frame counter modulus. The value of these bits determines the divide ratio for the FR output frequency. The FR frequency is given as SCLK/(CNT + 2) where CNT is a binary value of CR23-CR16. The following should be noted when configuring the divide ratio: 1. All ones in CR23-CR16 indicate a degenerative state and should be avoided. 2. Bits are operational whether SCLK is an input or an output. 3. CNT must be greater than seven.
27-24	SCLK prescale control bits. As an output, SCLK is derived from the microcomputer system clock, X2/CLKIN. Prescale divide ratios are selectable through these control bits (see Section 3.9.3 for the available divide ratios).
28	FR pulse-width control. This bit controls the pulse width of the FR output to select data-transfer rates with combo-codec circuits. When set to logic 0, the FR output framing pulse is one SCLK cycle wide for the fixed data-rate mode and appears in the serial-clock cycle preceding the first serial-bit transmission. When set to logic 1, the FR output framing pulse is eight SCLK cycles wide for the variable data-rate mode. In this mode, the framing pulse is active high for the duration of the eight bits transmitted and received.
29	Two's-complement μ-law/A-law conversion enable (TMS320C17/E17). When set to logic 0, sign-magnitude companding is enabled. When set to logic 1, two's-complement companding is enabled.
30	8/16-bit length coprocessor mode select (TMS320C17/E17). When set to logic 0, the 8-bit byte length is used. When set to logic 1, the 16-bit word length is selected.
31	Reserved for future expansion. This bit should be set to zero.

4. Assembly Language Instructions

The instruction set of the TMS320C1x (first-generation TMS320) processors supports numeric-intensive signal processing operations and general-purpose applications, such as high-speed control. 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, the TMS320C1x instruction set 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 in only two cycles. The SUBC (conditional subtract) instruction performs the shifting and conditional branching necessary to implement a divide efficiently and quickly.

This section describes the TMS320C1x 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 TMS320C1x 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 contains 16/128 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 TMS320C1x 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 TMS320C1x 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.

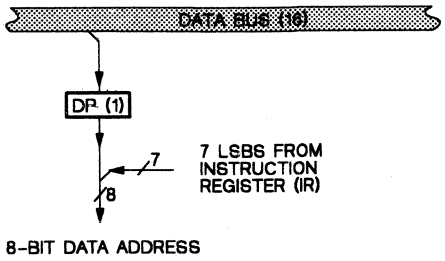


Figure 4-1. Direct Addressing Block Diagram

Direct addressing can be used with all instructions except CALL, the 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 >05 and appears in bits 15 through 8. The notation >nn indicates nn is a hexadecimal number. The shift count of >5 appears in bits 11 through 8 of the opcode. The data memory address >09 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 >08A8, 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 >0888.

Example 3:

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

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 >08A1.

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 >08A0.

4.1.3 Immediate Addressing Mode

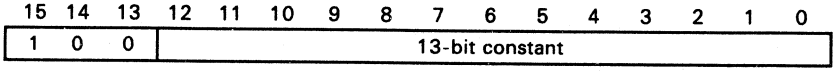
Included in the TMS320C1x instruction set are five immediate operand instructions, in which the immediate operand is contained within the instruction word. These instructions execute within a single instruction cycle. The length of the constant operand is instruction-dependent. The immediate instructions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

LACK Load accumulator immediate short (8-bit constant)
LARK Load auxiliary register immediate short (8-bit constant)
LARP Load auxiliary register pointer (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:

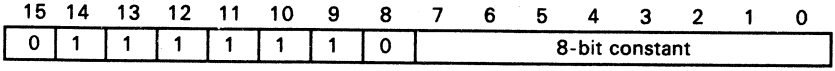
Example 1:

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



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.



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
DP	Data page pointer
I	Addressing mode bit
INTM	Interrupt mode bit
K	Immediate operand field
>nn	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 are multicycle.

Table 4-2. Instruction Set Summary

ACCUMULATOR MEMORY REFERENCE INSTRUCTIONS				
Mnemonic	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	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

T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS				
Mnemonic	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

Table 4-2. Instruction Set Summary (Concluded)

BRANCH/CALL INSTRUCTIONS				
Mnemonic	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 ≥ 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
BIOZ	Branch on I/O status = 0	2	2	1111 0110 0000 0000 0000 BBBB BBBB BBBB
BLEZ	Branch if accumulator ≤ 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 ≠ 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

CONTROL INSTRUCTIONS				
Mnemonic	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

I/O AND DATA MEMORY OPERATIONS				
Mnemonic	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

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

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.

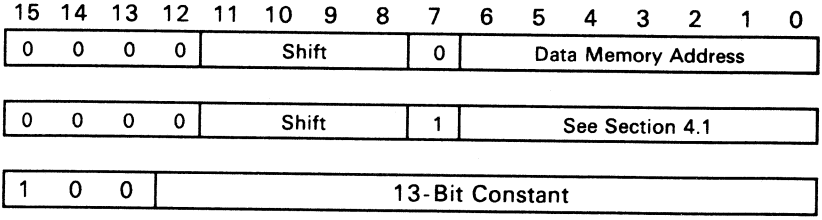
Execution

(PC) + 1 → PC
(ACC) + (dma) × 2^{shift} → ACC

1 → 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



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

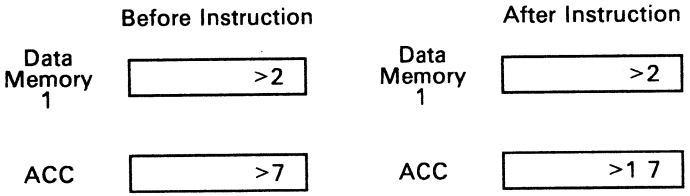
The digit specifies the number of memory words required to store the instruction and its extension words.

Cycles 1

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

Example

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



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

Absolute Value of Accumulator

Syntax [<label>] ABS

Operands None

Execution
(PC) + 1 → PC
If (ACC) < 0:
Then −(ACC) → 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	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 >80000000 is a special case. When the overflow mode is not set, the ABS of >80000000 is >80000000. When in the overflow mode, the ABS of >80000000 is >7FFFFFFF.

Words 1
Cycles 1

Example

```
ABS
```

	Before Instruction		After Instruction
ACC	>1234	ACC	>1234
ACC	>FFFFFFFF	ACC	>1

Syntax

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

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

Execution (PC) + 1 → PC
(ACC) + (dma) × 2^{shift} → ACC

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 1

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

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

Example 2

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

	Before Instruction	After Instruction
Data Memory 2	>8B0E	>8B0E
ACC	>0	>FFF8B0E0

Syntax

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

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

Execution (PC) + 1 → PC
(ACC) + (dma) x 2^{16} → ACC

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

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

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

Add to Accumulator with Sign-Extension Suppressed

Syntax

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

Operands $0 \leq dma \leq 127$
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						

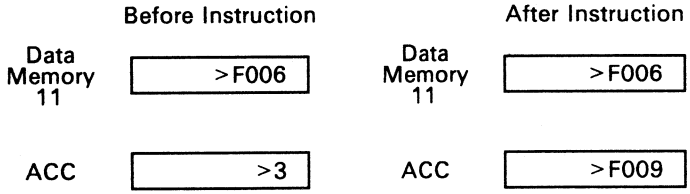
Description Contents of the specified data memory location are added with sign-extension suppressed. The data is treated as a 16-bit unsigned number rather than a two's-complement number. Therefore, there is no sign-extension as with the ADD instruction.

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

Words 1
Cycles 1

Example

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



Syntax

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

Operands $0 \leq dma \leq 127$
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

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

	Before Instruction	After Instruction
Data Memory 16	>FF	>FF
ACC	>12345678	>78

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	>40	P	>40
ACC	>20	ACC	>60

Syntax [<label>] B <pma>

Operands 0 ≤ pma ≤ 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 ≤ pma ≤ 4095

Execution
If (AR bits 8-0) ≠ 0:
Then pma → PC;
Else (PC) + 2 → PC
(AR) - 1 → 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	>1	AR	>0
PC	>46	PC	>35

or

AR	>0	AR	>FFFF
PC	>46	PC	>48

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.

Syntax [<label>] BGEZ <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	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.
```

Syntax [<label>] BGZ <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	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

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

Syntax [<label>] BIOZ <pma>

Operands $0 \leq pma \leq 4095$

Execution If $\overline{BIO} = 0$:
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	0	1	1	0	0	0	0	0	0	0	0	0
Program Memory Address															

Description If the $\overline{BIO}$ pin is active low, then branch to the specified program memory location. 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.

The BIOZ instruction in conjunction with the $\overline{BIO}$ pin can be used to test if a peripheral is ready to send or receive data. Polling the $\overline{BIO}$ pin using BIOZ may be preferable to an interrupt when executing time-critical loops.

Words 2
Cycles 2

Example

```
BIOZ    PRG64    If the BIO- pin is active (low), then
                 a branch to location 64 occurs. Otherwise,
                 the program counter is incremented.
```

Syntax [<label>] BLEZ <pma>

Operands $0 \leq pma \leq 4095$

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	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

```
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

Syntax [<label>] BLZ <pma>

Operands $0 \leq pma \leq 4095$

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	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

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

Syntax [<label>] BNZ <pma>

Operands $0 \leq pma \leq 4095$

Execution If (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.
```

Syntax [<label>] BV <pma>

Operands 0 ≤ pma ≤ 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 $0 \leq pma \leq 4095$

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.
```

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	>25	>83
ACC	>83	>83
Stack	>32 >75 >84 >49	>26 >32 >75 >84

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	>33	>6D
Stack	>71 >48 >16 >80	>35 >71 >48 >16

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. Note that the LST instruction does not affect INTM.

The unmaskable interrupt, $\overline{RS}$, is not disabled by this instruction. Interrupts are also disabled by a reset.

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

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

Execution

(PC) + 1 → PC
(dma) → 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	>43	>43
Data Memory 9	>2	>43

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.
```

Input Data from Port

Syntax

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

Operands

0 ≤ dma ≤ 127
ARP = 0 or 1
0 ≤ port address PA ≤ 7

Execution

(PC) + 1 → PC
Port address → address lines A2/PA2-A0/PA0
0 → address bus A11-A4
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. $\overline{DEN}$ goes low during the same cycle, strobing in the data that the addressed peripheral places on the data bus D15-D0. On the TMS32010/C10/C15, $\overline{MEN}$ remains high when $\overline{DEN}$ is active. On the TMS320C17, the $\overline{MEN}$ signal is not available.

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

Syntax

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

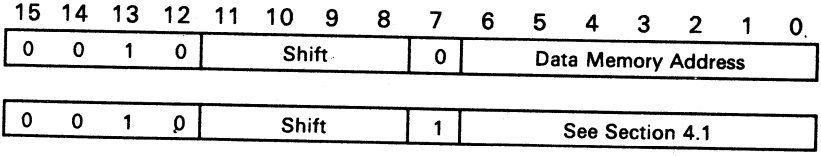
Operands

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

Execution

(PC) + 1 → PC
(dma) x 2^{shift} → ACC

Encoding



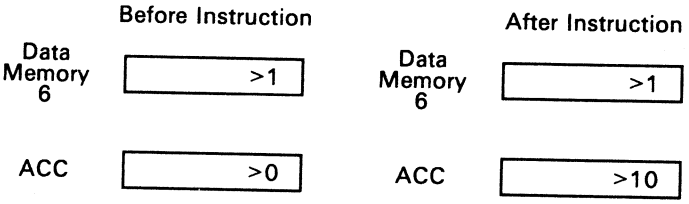
Description

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.
```



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   >15
```

	Before Instruction	After Instruction
ACC	>31	>15

Syntax

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

Operands

0 ≤ dma ≤ 127
auxiliary register AR = 0 or 1
ARP = 0 or 1

Execution

(PC) + 1 → PC
(dma) → 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.

AR0 is not decremented after the LAR instruction. 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	>18	>18
AR0	>6	>18

also,

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

	Before Instruction	After Instruction
Data Memory 7	>32	>32
AR0	>7	>32

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,>21
```

	Before Instruction		After Instruction
AR0	>0	AR0	>21

Syntax [<label>] LARP <constant>

Operands $0 \leq$ constant ≤ 1

Execution (PC) + 1 → PC
Constant → 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.
```

Load Data Memory Page Pointer

Syntax

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

Operands

$0 \leq dma \leq 127$
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 128-143/255. 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	>FEDC	>FEDC
DP	>1	>0

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 128-143/255. 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.
```

Syntax

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

Operands $0 \leq dma \leq 127$
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 $0 \leq dma \leq 127$
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	>62	>62
T	>3	>62

LTA Load T Register and Accumulate Previous Product LTA

Syntax

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

Operands $0 \leq dma \leq 127$
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	>62	>62
T	>3	>62
P	>F	>F
ACC	>5	>14

Load T Register, Accumulate Previous Product, and Move Data

Syntax

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

Operands

$0 \leq dma \leq 127$
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						

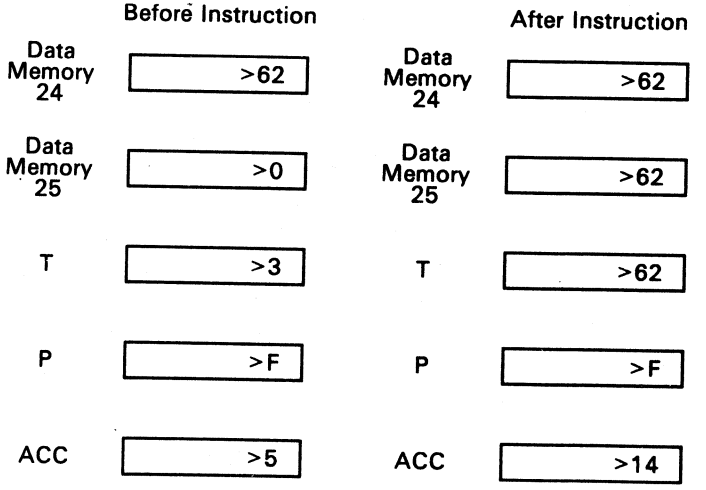
Description

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.
```



Syntax

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

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

Example 2

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

	Before Instruction		After Instruction
AR1	>35	AR1	>34

Example 3

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

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

Syntax

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

Operands $0 \leq$ dma ≤ 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 >8000 x >8000. If this condition arises, the product will be >C0000000.

Words 1
Cycles 1

Example

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

	Before Instruction	After Instruction
Data Memory 13	>7	>7
T	>6	>6
P	>36	>2A

Syntax [<label>] MPYK <constant>

Operands $-2^{12} \leq$ 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	>7	>7
P	>2A	>FFFFFFC1

No Operation

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 $0 \leq dma \leq 127$
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	>F000	>F000
ACC	>100002	>10F002

Syntax

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

Operands

0 ≤ dma ≤ 127
ARP = 0 or 1
0 ≤ port address PA ≤ 7

Execution

(PC) + 1 → PC
Port address PA → address bus A2/PA2-A0/PA0
0 → 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, $\overline{WE}$ goes low and the data word is placed on the data bus D15-D0. On the TMS32010/C10/C15, $\overline{MEN}$ remains high during the first cycle. On the TMS 320C17 , the $\overline{MEN}$ signal is not available.

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.
```

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	>144	P	>144
ACC	>23	ACC	>144

Syntax [<label>] POP

Operands None

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

Encoding

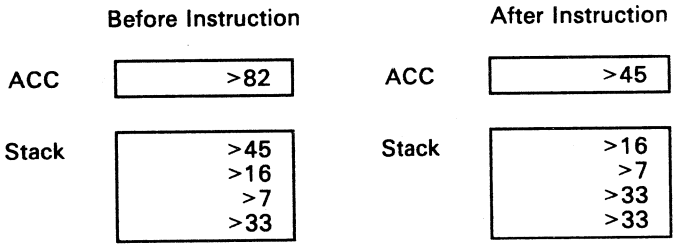
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1

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
```



Push Low Accumulator onto Stack

Syntax [<label>] PUSH

Operands None

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

Encoding

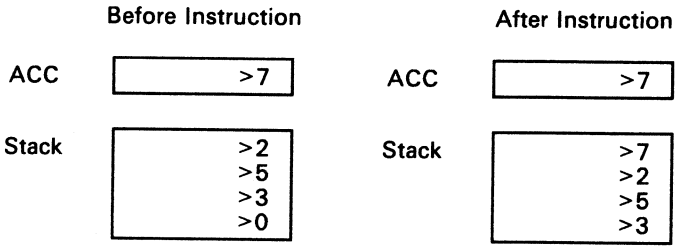
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0

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
```



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	>96	>37
Stack	>37 >45 >75 >75	>45 >75 >75 >75

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.
```

Syntax

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

Operands

$0 \leq dma \leq 127$
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	>4208001	>4208001
Data Memory 70	>0	>841

Syntax

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

Operands

0 ≤ dma ≤ 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	>5	>8421
ACC	>7C638421	>7C638421

Syntax

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

Operands 0 ≤ dma ≤ 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 1

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

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

Example 2

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

AR0	>5	>6
Data Memory 5	>0	>6

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 (>7FFFFFFF) or negative (>80000000) 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

```
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	>24	P	>24
ACC	>3C	ACC	>18

Syntax

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

Operands

0 ≤ dma ≤ 15 (TMS32010/C10)
0 ≤ dma ≤ 127 (TMS320C15/C17)
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	>5EFE	>5EFE
Data Memory 1	>A	>5EFE

Syntax

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

Operands

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

Execution

(PC) + 1 → PC
(ACC) - [(dma) x 2^{shift}] → 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	>24	>13
Data Memory 59	>11	>11

Syntax

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

Operands

$0 \leq dma \leq 127$
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 <u>not</u> 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	>7	Data Memory 2	>7
ACC	>41	ACC	>20009

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 $0 \leq dma \leq 127$
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	1	0	0	Data Memory Address						
Indirect:	0	1	1	0	0	0	1	0	1	See Section 4.1						

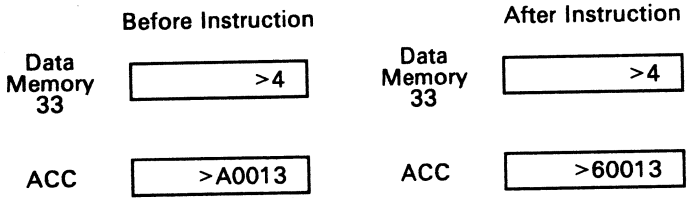
Description The contents of the addressed data memory location are subtracted from the upper 16 bits of the accumulator. The 16 low-order bits of the accumulator are unaffected. The result is stored in the accumulator.

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.
```



Syntax

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

Operands $0 \leq dma \leq 127$
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	>F003	>F003

	Before Instruction	After Instruction
ACC	>F105	>102

Syntax

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

Operands

$0 \leq dma \leq 127$
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

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

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

Syntax

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

Operands

$0 \leq dma \leq 127$
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.

Note that the TBLW and OUT instructions use the same external signals and thus cannot be distinguished when writing to program memory addresses 0 through 7.

Words 1
Cycles 3

Example

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

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

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	>F0F0	>F0F0
ACC	>12345678	>1234A688

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	>A5A5A5A5	ACC	>0

Zero Low Accumulator and Load High Accumulator

Syntax

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

Operands

$0 \leq dma \leq 127$
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	>3F01	>3F01
ACC	>77FFFF	>3F010000

Zero Accumulator, Load Low Accumulator with Sign-Extension Suppressed

Syntax

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

Operands

0 ≤ dma ≤ 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

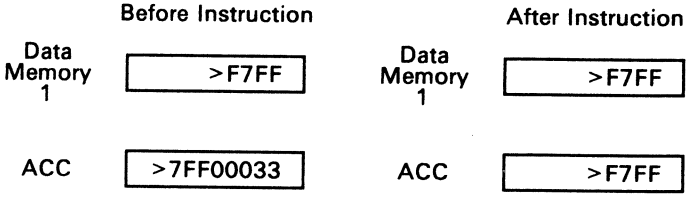
The contents of the addressed data memory location are loaded into the 16 low-order bits of the accumulator. The upper half of the accumulator is zeroed. The data is treated as a 16-bit unsigned number rather than a two's-complement number. Therefore, there is no sign-extension with this instruction.

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.
```



5. Software Applications

The use of various key software-related processor and instruction set features along with assembly language coding examples is explained in this section. TMS320C1x (first-generation TMS320) instructions are tailored to digital signal processing tasks, providing a single-cycle multiply, scaling, convolution, overflow management, and many other features. There is also instruction set support for logical and arithmetic operations.

More information about specific applications can be found in the book, *Digital Signal Processing Applications with the TMS320 Family*. The DSP Software Library contains the major DSP routines and application algorithms presented in the applications book. The TMS320 DSP Bulletin Board Service provides access to code updates and new application reports as they become available. See Appendix E for information about the software library and bulletin board.

Major topics discussed in this section are listed below.

5.1 Processor Initialization

Prior to the execution of a digital signal processing algorithm, it is necessary to initialize the processor. Generally, initialization takes place anytime the processor is reset.

When reset is activated by applying a low level to the $\overline{RS}$ (reset) input for a minimum of five cycles, the TMS320C1x terminates program execution and forces the program counter (PC) to zero. Program memory location 0 normally contains a B (branch) instruction in order to direct program execution to the system initialization routine following the reset. The hardware reset also initializes various registers and status bits.

After reset, the processor should be initialized through software. The initialization routine should set up operational modes, memory pointers, interrupts, and the remaining functions necessary to meet system requirements. This section describes how to configure the TMS320C1x devices after reset and provides code for processor initialization.

5.1.1 TMS32010/C10/C15 Initialization

To configure the TMS32010/C10/C15 processor after reset, the following internal functions should be initialized:

- Interrupt structure
- Overflow mode control (OVM)
- Auxiliary registers and auxiliary register pointer (ARP)
- Data memory page pointer (DP).

Note that the OVM (overflow mode) bit, INTM (interrupt mode) bit, auxiliary register pointer (ARP), and data memory page pointer (DP) are not initialized by reset.

Example 5-1 shows coding for initializing the TMS32010/C10/C15 to the following machine state, in addition to the initialization performed during the hardware reset:

- Interrupt enabled
- Overflow mode (OVM) disabled
- Data memory page pointer (DP) set to zero
- Auxiliary register pointer (ARP) set to zero
- Internal memory filled with zeros.

Example 5-1. TMS32010/C10/C15 Processor Initialization

```
        TITL  'PROCESSOR INITIALIZATION'
        IDT   'EXAMPLE'
        DEF   RESET,INT
        REF   ISR
*
* PROCESSOR INITIALIZATION.
* RESET AND INTERRUPT VECTOR SPECIFICATION.
*
        AORG  >0
RESET   B     INIT     ; RS- BEGINS PROCESSING HERE
INT     B     ISR      ; INT- BEGINS PROCESSING HERE
*
* THE BRANCH INSTRUCTION AT PROGRAM MEMORY LOCATION 0 DIRECTS
* EXECUTION TO BEGIN HERE FOR RESET PROCESSING THAT INITIAL-
* IZES THE PROCESSOR. WHEN RESET IS APPLIED, THE FOLLOWING
* CONDITIONS ARE ESTABLISHED FOR THE STATUS REGISTER:
*
*       OV OVM INTM 12 11 10 9 ARP 7 6 5 4 3 2 DP
* ST:   0   X   1    1  1  1 1  X  1 1 1 1 1 1 X
*
INIT    ROVM           ; DISABLE OVERFLOW MODE
        LDPK  0        ; POINT DP TO DATA PAGE 0
        LARK  0,255    ; SET LOOP COUNT FOR DATA MEM INIT TO
                       ; 143 FOR 32010/11 AND 255 FOR 320C15/17
*
* INTERNAL DATA MEMORY INITIALIZATION.
*
        ZAC            ; CLEAR THE ACCUMULATOR
        LARP  0        ; USE AR0 FOR POINTER AND LOOP CONTROL
LOOP    SACL  *        ; CLEAR DATA MEMORY
        BANZ  LOOP     ; CHECK IF DONE AND DECREMENT AR0
*
* THE PROCESSOR IS INITIALIZED. THE REMAINING APPLICATION-
* DEPENDENT PART OF THE SYSTEM SHOULD NOW BE INITIALIZED.
*
        EINT           ; ENABLE ALL INTERRUPTS
```

5.1.2 TMS320C17 Initialization

To configure the TMS320C17 devices after reset, the following internal functions must be initialized:

- Interrupt structure
- Serial-port framing-pulse generation selection
- Serial-port connection
- Companding hardware
- Serial-port clock
- Auxiliary register pointer
- Data memory page pointer
- Overflow mode.

Two of the I/O ports are dedicated to the serial port and companding hardware, the operation of which is determined by the 32 bits of the system control register. Table 5-1 lists the control register bits with brief definitions.

Table 5-1. Control Register Bit Definitions

CR BIT #	DEFINITION
	PORT 0
CR3 - CR0	Interrupt flags
CR7 - CR4	Interrupt mask bits
CR8	Port 1 configuration control
CR9	External framing enable for serial port transfers
CR10	XF external logic output flag latch
CR11	Serial port companding mode select
CR13 - CR12	Companding hardware enable
CR14	A-law/μ-law conversion select
CR15	Serial clock (SCLK) control
	PORT 1
CR23 - CR16	Frame counter modulus
CR27 - CR24	Serial clock (SCLK) prescale control (divide ratios)
CR28	FR pulse-width control
CR30 - CR29	I/O control on TMS320C17/E17;
CR31	Reserved for future expansion (set to 0)

Example 5-2 shows coding for initializing the TMS 320C17 serial-port and companding hardware for interface to a codec. The following machine state is loaded:

- Set the lower control register bit 8 (CR8) to enable port 1 to access the upper control register. To insure safe system operation, SCLK should be left as an input to the device (CR15 set to logic 0). This prevents any invalid serial-port timing during the initialization routine. The value loaded into the lower control register to accomplish this is >3988.

- The upper control register is set as follows:
  - Long FR pulse (variable data-rate selected)
  - SCLK divide ratio of 10
  - FR frequency at SCLK/256 for an 8-kHz framing pulse
  - The value >1CFE loaded into the upper control register.

- The lower control register is then configured as follows:
  - Interrupt flags cleared
  - Active FR interrupt enabled
  - Port transfers enabled by active FR
  - Serial companding mode selected (see Section 5.6.1)
  - Companding hardware enabled
  - μ-law conversion selected
  - SCLK selected as an output
  - The value >3888 now loaded into the lower control register.

  Note that the interrupt flags are flip-flops. Writing a one to an interrupt flag clears it and sets the corresponding flag to zero; i.e., a write to the flags affects the clear or reset input of the flip-flops.

Example 5-2. TMS320C17 Processor Initialization

```
* A BRANCH INSTRUCTION AT PROGRAM MEMORY LOCATION 0 DIRECTS
* PROCESSOR EXECUTION HERE.  THE CONTROL REGISTER VALUES ARE
* STORED IN ROM STARTING AT LOCATION 4.  THESE VALUES ARE
* THEN READ INTO RAM FOR THE OUT INSTRUCTIONS TO THE CONTROL
* REGISTER.  MEMORY LOCATIONS SET1-SET3 AND ONE ARE LOCATED
* ON RAM PAGE 1. THE PROGRAM MEMORY LOCATION HAS A BRANCH TO
* THE INTERRUPT SERVICE ROUTINE.
*
       DEF   RESET,INT,INIT
       REF   ISR
*
ONE    EQU   1        ; CONSTANT ONE
SET1   EQU   2        ; LOWER CONTROL REGISTER
SET2   EQU   3        ; UPPER CONTROL REGISTER
SET3   EQU   4        ; LOWER CONTROL REGISTER
*
* PROCESSOR INITIALIZATION.
* RESET AND INTERRUPT VECTOR SPECIFICATION.
*
       AORG  >0
RESET  B     INIT     ; RS- BEGINS PROCESSING HERE
INT    B     ISR      ; INT- BEGINS PROCESSING HERE
TABLE  DATA  >3988    ; CONTROL REGISTER DATA
       DATA  >1CFE
       DATA  >3888
*
INIT   DINT           ; DISABLE INTERRUPTS
       SOVM           ; SET OVERFLOW MODE
       LARP  0        ; USE AUXILIARY REGISTER 0
       LDPK  1        ; WORK IN RAM PAGE 1
       LACK  1        ; ACC = 1
       SACL  ONE      ; STORE 1 IN MEMORY LOCATION ONE
       LACK  TABLE    ; START AT LOCATION 4
       TBLR  SET1     ; READ VALUE >3988 TO RAM
       ADD   ONE,0    ; INCREMENT ADDRESS
       TBLR  SET2     ; READ VALUE >1CFE TO RAM
       ADD   ONE,0    ; INCREMENT ADDRESS
       TBLR  SET3     ; READ VALUE >3888 TO RAM
       OUT   SET1,0   ; CONFIGURE LOWER CONTROL REGISTER
       OUT   SET2,1   ; CONFIGURE UPPER CONTROL REGISTER
       OUT   SET3,0   ; CONFIGURE LOWER CONTROL REGISTER
       LDPK  0        ; RESET RAM PAGE TO 0
*
* THE PROCESSOR IS INITIALIZED.  THE REST OF THE SYSTEM THAT
* IS APPLICATION-DEPENDENT SHOULD BE INITIALIZED BEFORE THE
* EINT INSTRUCTION.
*
       EINT           ; ENABLE INTERRUPTS
```

In addition to the above configuration and code, the TMS320C17/E17 requires the following:

- Control register bits CR29 and CR30 must be initialized.
- The data operand of the upper control register is set at >6CBE. This selects two's-complement companding for the serial port and 16-bit length coprocessor mode (i.e., for interface to 16-bit processors). When two's-complement companding is used, there must be at least one instruction between an OUT instruction to the serial port transmit register and an IN instruction from the serial port receive register.
- If the TMS320C17 is programmed to generate the serial port framing pulse using the internal timer (CR9 = 0), the FR interrupt flag will be set regardless of whether or not the FR interrupt has been enabled.

5.2 Interrupt Management

The interrupt function allows the current process to be suspended in order to perform a more critical function. On the TMS32010/C10/C15, processor execution may be suspended on a high-priority basis by using the $\overline{INT}$ pin. Otherwise, a lower priority interrupt can be serviced by using a software ($\overline{BIO}$) polling technique.

The TMS 320C17 has four interrupts maskable via the system control register. These 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 when multiple interrupts are enabled.

Processing in the interrupt service routine (ISR) must assure that the processor context is saved before and during execution and restored when the routine is finished. Descriptions and examples of how to implement interrupt service routines, $\overline{BIO}$ polling, and context switching are provided in this section.

5.2.1 TMS32010/C10/C15 Interrupt Service Routines

The TMS32010/C10 and TMS320C15/E15 devices provide one maskable interrupt ($\overline{INT}$). By using the $\overline{INT}$ pin, the processor's execution can be suspended at any point in the program except after a multiply instruction. The instruction following the MPY and MPYK instructions is always executed.

Interrupt processing on the TMS32010/C10/C15/E15 begins as follows:

1) The EINT (enable interrupt) instruction is executed, which sets the INTM (interrupt mode) bit to 0 so that an interrupt can be received.
2) When an interrupt occurs, the INTF (interrupt flag) bit is set to 1.

As interrupt servicing begins, the following sequence occurs automatically:

1) The interrupt is acknowledged, which clears the INTF (interrupt flag) bit to 0.
2) The INTM (interrupt mode) bit is set to 1 to disable further interrupts.
3) The current PC is pushed onto TOS (top of stack).
4) The new PC is set to 2.

During 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 a branch to the interrupt service routine.
2) The interrupt service routine is executed. The context of the machine can be saved and the source of the interrupt serviced. Then, the context is restored and the interrupts enabled prior to returning from the interrupt routine.
3) The EINT (enable interrupt) instruction is executed, which sets the INTM (interrupt mode) bit to 0.
4) The RET instruction is executed.

The hardware interrupt can be masked at critical points in the program with the DINT instruction. This sets the INTM (disable interrupt mode) bit to logic

one. If an interrupt occurs while INTM equals one, the interrupt will not be serviced until the interrupts are enabled again. However, the INTF (interrupt flag) is set to one, and the interrupt is held pending. The interrupt will be serviced when the INTM bit is set to zero by executing the EINT instruction. If an interrupt is pending when an enable interrupt operation occurs, the interrupt is serviced after the execution of the instruction following the EINT instruction. This allows for a return instruction to be executed before an interrupt is acknowledged.

An interrupt-driven analog input channel can be implemented using the technique described and shown in Example 5-3. However, multiple-level data buffering will impact system I/O overhead. Analog systems supported by first-generation TMS320 devices usually have information bandwidths of less than 20 kHz. The desired sample rate can be generated by dividing the CLKOUT signal from the TMS320. It is advisable to provide at least a one-level data buffer to ensure the integrity of the data read by the processor. If an 8-kHz sample rate is used (for example), the system must then respond to an analog interrupt every 125 μs. The percentage of I/O overhead incurred by this arrangement can be computed by determining the number of clock cycles that the TMS320 will spend in the interrupt routine servicing each sample and dividing by the number of clock cycles available between each sample. Example 5-3 shows a typical interrupt service routine. Note that the memory location flag (FLAG) contains a 1-bit flag to indicate that the required number of samples have been received.

Example 5-3. TMS32010/C10/C15 Interrupt Service Routine

```
* THIS ROUTINE SERVICES AN EXTERNAL INTERRUPT. IT MAY BE
* LOCATED AT PROGRAM MEMORY LOCATION 2, OR A BRANCH AT
* LOCATION 2 DIRECTS PROGRAM EXECUTION HERE. THE ROUTINE
* READS DATA FROM AN EXTERNAL DEVICE (A/D CONVERTER). THE
* NUMBER OF SAMPLES OBTAINED ARE STORED IN MEMORY LOCATION
* COUNT. LIMIT IS THE NUMBER OF SAMPLES NEEDED. MEMORY
* LOCATION ONE CONTAINS THE CONSTANT 1. STATUS IS ALWAYS
* STORED ON DATA PAGE 1 WHEN USING DIRECT MEMORY ADDRESSING.
* ASSUME AR0 POINTS TO THE NEXT EMPTY LOCATION IN THE SAMPLE
* BUFFER.
*
ADC     EQU   0         ; ASSIGN PA0 TO A/D CONVERTER
STATUS  EQU   0
ACCL    EQU   1         ; ASSIGN MEM LOCATION TO SAVE STATUS/ACC
ACCH    EQU   2
SAMP    EQU   3         ; STORE INPUT DATA HERE
COUNT   EQU   4         ; COUNT # OF SAMPLES HERE
FLAG    EQU   5         ; ASSIGN MEM LOCATION TO FLAG
LIMIT   EQU   32        ; ASSIGN TOTAL # OF SAMPLES REQUIRED
*
ISR     SST   STATUS    ; SAVE STATUS
        LDPK  1         ; USE DATA PAGE 1
        SACL  ACCL      ; SAVE ACCUMULATOR LOW
        SACH  ACCH      ; SAVE ACCUMULATOR HIGH
        LARP  0         ; USE AR0
        IN    *-,ADC    ; READ FROM ADC
        LAC   COUNT     ; LOAD SAMPLE COUNTER
        ADD   ONE       ; INCREMENT
        SACL  COUNT     ; STORE UPDATED COUNT
        LACK  LIMIT
        SUB   COUNT     ; CHECK IF LIMIT EXCEEDED
        BGZ   OK
DONE    LACK  1
        SACL  FLAG      ; YES --> SET FLAG
OK      ZALH  ACCH      ; RESTORE ACCUMULATOR HIGH
        ADDS  ACCL      ; RESTORE ACCUMULATOR LOW
        LST   STATUS    ; RESTORE STATUS
        EINT            ; ENABLE SUBSEQUENT INTERRUPTS
        RET
```

If the processor is using a 20-MHz clock, the number of available cycles between each sample is 625. The overhead required to service this system is 18/625 = 2.9 percent. This overhead burden can be reduced by using a FIFO (first in, first out) to buffer the data. In this case, the TMS320 need only be interrupted when the buffer has filled. If a 16-level FIFO is used in the example above, this interrupt will occur every 2 ms, and the overhead burden will be reduced to about 0.5 percent.

If two different kinds of devices are being serviced by the same interrupt routine, the $\overline{BIO}$ pin can be used to determine which device needs to be serviced (see Section 5.2.3 for $\overline{BIO}$ polling).

5.2.2 TMS320C17 Interrupt Service Routines

The TMS320C17/E17 have four maskable interrupts: $\overline{\text{EXINT}}$ (TMS320C17), $\overline{\text{FSR}}$, $\overline{\text{FSX}}$ and FR. The interrupts are maskable via the system control register bits CR7-CR4. Bits CR3-CR0 serve as the interrupt flags for the four interrupts. An active signal on any of these pins sets the corresponding interrupt flag to one. Since all four interrupts activate a single master interrupt flag, the interrupt service routine (ISR) should poll all four interrupt flags and check for the corresponding interrupt source. The ISR may also need to poll the individual mask bits (CR7-CR4) before recognizing the interrupt flag.

Interrupt processing on the TMS 320C17/E17 begins as follows:

1) The EINT (enable interrupt) instruction is executed, which sets the INTM (interrupt mode) bit to 0 so that an interrupt can be received.
2) When an interrupt occurs, the INTF (interrupt flag) bit is set to 1.

As interrupt servicing begins, the following sequence occurs automatically:

1) The interrupt is acknowledged, which clears the INTF (interrupt flag) bit to 0.
2) The INTM (interrupt mode) bit is set to 1 to disable further interrupts.
3) The current PC is pushed onto TOS (top of stack).
4) The new PC is set to 2.

During 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 a branch to the interrupt service routine.
2) The interrupt service routine is executed. The context of the machine may be stored and 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 four flags has been set in the control register.
   b) Check for corresponding mask bits before proceeding (optional).
   c) Clear that flag (set to 0) and service the source of that flag. There must be an interval of at least four clock cycles after the flag has been set before clearing it.
3) The EINT (enable interrupt) instruction is executed, which sets the INTM (interrupt mode) bit to 0.
4) The RET instruction is executed.

All interrupts are synchronized and multiplexed into the master interrupt circuitry and have the same priority. However, interrupt priorities in polling the interrupt flags can be established by the user. The ISR should clear the interrupt flag before executing an EINT instruction or enabling the interrupts. Note that writing a one to an interrupt flag will clear it, i.e., set the corresponding flag to zero. In the coprocessor mode on the TMS320C17, the $\overline{\text{BIO}}$ and $\overline{\text{EXINT}}$ lines cannot be driven externally, but are reserved for transfers to/from the coprocessor port. An example interrupt service routine for a system with three active interrupts enabled is given in Example 5-4. Polling is also included in the code example.

Example 5-4. TMS320C17 Interrupt Service Routine

```
* THIS ROUTINE MAY BE LOCATED AT PROGRAM MEMORY LOCATION 2,
* OR A BRANCH INSTRUCTION AT LOCATION 2 DIRECTS PROGRAM
* EXECUTION HERE.  MEMORY LOCATION ONE CONTAINS THE
* CONSTANT 1. STATUS IS ALWAYS STORED ON DATA PAGE 1 WHEN
* DIRECT MEMORY ADDRESSING IS USED.
*
* RECV IS THE SERVICE ROUTINE FOR THE RECEIVE INTERRUPT.
* XINT IS THE SERVICE ROUTINE FOR THE EXTERNAL INTERRUPT.
* TRANS IS THE SERVICE ROUTINE FOR THE TRANSMIT INTERRUPT.
*
       DEF   ISR,RECV
       REF   XINT,TRANS
*
STATUS EQU   0
ACCL   EQU   1        ; ASSIGN MEM LOCATION TO SAVE STATUS/ACC
ACCH   EQU   2
RBUF   EQU   3        ; STORE RECEIVE DATA HERE
CREG   EQU   4        ; TEMP LOCATION TO STORE CONTROL REG
*
ISR    SST   STATUS   ; SAVE STATUS
       LDPK  1        ; USE DATA PAGE 1
       SACL  ACCL     ; SAVE LOW ACCUMULATOR
       SACH  ACCH     ; SAVE HIGH ACCUMULATOR
*
* THIS ROUTINE CHECKS FOR THREE ACTIVE INTERRUPTS OCCURRING
* AND SERVICES THEM ACCORDINGLY. IT IS ASSUMED THAT ONE OF
* THREE IS THE SOURCE OF THE INTERRUPT. AFTER AN INTERRUPT
* FLAG IS SET, IT MUST BE RESET BY THE INTERRUPT SERVICE
* ROUTINE TO AVOID BEING INTERRUPTED AGAIN ON THE RETURN
* FROM THE SUBROUTINE.
*
     IN    CREG,PA0 ; READ LOWER CONTROL REGISTER
     LAC   ONE,0    ; LOAD INT- INTERRUPT MASK
     AND   CREG     ; INT FLAG SET?
     BNZ   XINT     ; GO TO INT SERVICE ROUTINE
     LAC   ONE,2    ; LOAD FSX- INTERRUPT MASK
     AND   CREG     ; FSX FLAG SET?
     BNZ   TRANS    ; GO TO TRANSMIT SERVICE ROUTINE
*
* INTERRUPT MUST BE FSR-.
*
RECV SACL  CREG     ; CLEAR FSR INTERRUPT FLAG
     OUT   CREG,PA0 ; RESTORE CONTROL REGISTER
     IN    RBUF,PA1 ; READ REC DATA FROM PORT 1
*
* RESTORE STATUS.
*
     ZALH  ACCH     ; RESTORE HIGH ACCUMULATOR
     ADDS  ACCL     ; RESTORE LOW ACCUMULATOR
     LST   STATUS   ; RESTORE STATUS
     EINT           ; ENABLE INTERRUPTS
     RET
```

5.2.3 $\overline{\text{BIO}}$ Polling

A low priority interrupt can be serviced by using $\overline{\text{BIO}}$ polling. The BIOZ instruction can be used to poll (or test) the $\overline{\text{BIO}}$ pin to see if a device needs to be serviced. This method allows a critical loop or set of instructions to be executed without a variation in execution time. Because the test for the $\overline{\text{BIO}}$ pin occurs at defined points in the program, context saves are minimal.

The $\overline{\text{BIO}}$ pin can be used to monitor the status of a peripheral. If the FIFO (first in, first out) full status line is connected to the $\overline{\text{BIO}}$ pin, the FIFO is serviced only when the FIFO is full. In the following code segment, the FIFO contains 16 data words. The $\overline{\text{BIO}}$ pin is tested after each time-critical function has been executed.

```
        BIOZ    SKIP
        CALL    SERVE
SKIP     .
         .
         .
```

The subroutine does not have to save the registers or the status, because a new procedure will be executed after the device is serviced, as shown below.

```
SERVE   LARK    AR0,15
        LARK    AR1,TABLE
LOOP    LARP    1
        IN      *+,PA0,AR0
        BANZ    LOOP
        RET
```

The FIFO must be serviced before another word is input or data may be lost. This fact determines the frequency at which the polling must take place.

5.2.4 Context Switching

Context switching, commonly required when processing a subroutine call or interrupt, may be quite extensive or simple, depending on system requirements such as the use made of the stack or auxiliary registers. Unless the interrupt service routine (ISR) is a simple I/O handler, the processing in the ISR generally must assure that the processor context is preserved during execution. The context must be saved before executing the routine itself and restored when the routine is finished. A common routine may be used to secure the context of the processor during interrupt processing.

The TMS320C1x program counter is stored automatically on the hardware stack. If there is any important information in the other TMS320C1x registers, such as the status or auxiliary registers, these must be saved by user software. A stack in data memory, identified by an auxiliary register, is useful for storing the machine state when processing interrupts.

During an interrupt, all registers except the P register can be saved and restored directly. However, the TMS320C1x 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 the MPY and MPYK instructions with LTA, LTD, PAC, APAC, or SPAC instructions that transfer data from the P register to the accumulator.

Examples of saving and restoring the state of the TMS320C1x processor are given in Example 5-5 and Example 5-6. Auxiliary register 1 (AR1) is used in both examples as the stack pointer. As the stack grows, it expands into lower memory addresses. The registers saved are the ST status register, accumulator (ACC), P register, T register, all four levels of the hardware stack, and auxiliary registers AR0 and AR1.

The routines in Example 5-5 and Example 5-6 are protected against interrupts, allowing context switches to be nested. This is accomplished by the use of the MAR *- and MAR *+ instructions at the beginning of the context save and context restore routines, respectively. Note that the last instruction of the context save decrements AR1 while the context restore is completed with an additional increment of AR1. This prevents the loss of data if a context save or restore routine is interrupted.

Example 5-5. Context Save

```
        TITL 'CONTEXT SAVE'
        DEF  SAVE
*
* CONTEXT SAVE ON SUBROUTINE CALL OR INTERRUPT. ASSUME THAT
* AR1 IS THE STACK POINTER AND AR1 = 128.
*
SAVE    LARP AR1      ; CHANGE POINTER TO AR1   AR1 = 128
        MAR  *-       ;                         AR1 = 127
*
* SAVE THE STATUS REGISTER.
*
        SST  *-       ; ST  --> (127),          AR1 = 126
*
* SAVE THE ACCUMULATOR.
*
        SACH *-       ; ACCH --> (126),         AR1 = 125
        SACL *-       ; ACCL --> (125),         AR1 = 124
*
* SAVE THE P REGISTER.
*
* THE P REGISTER CANNOT BE EASILY RESTORED FROM MEMORY. ON
* TMS320C1X DEVICES, IT IS ASSUMED THAT THE MPY AND MPYK
* INSTRUCTIONS HAVE BEEN FOLLOWED BY AN APAC, PAC, SPAC,
* LTA, OR LTD INSTRUCTION. HENCE, SAVING THE ACCUMULATOR
* HAS ALSO SAVED THE P REGISTER.
*
* SAVE THE T REGISTER.
*
        MPYK 1        ; T --> P
        PAC           ; T --> ACC
        SACL *-       ; T --> (124),            AR1 = 123
*
* SAVE ALL FOUR LEVELS OF THE HARDWARE STACK.
*
        POP           ; TOS       --> ACC,
        SACL *-       ; TOS  (4)  --> (123),    AR1 = 122
        POP           ; STACK(3)  --> ACC,
        SACL *-       ; STACK(3)  --> (122),    AR1 = 121
        POP           ; STACK(2)  --> ACC,
        SACL *-       ; STACK(2)  --> (121),    AR1 = 120
        POP           ; BOS  (1)  --> ACC,
        SACL *-       ; BOS  (1)  --> (120),    AR1 = 119
*
* SAVE AUXILIARY REGISTERS.
*
        SAR  AR0,*-   ; AR0 --> (119),          AR1 = 118
        SAR  AR1,*-   ; AR1 --> (118),          AR1 = 117
*
* SAVE IS COMPLETE.
```

Example 5-6. Context Restore

```
        TITL  'CONTEXT RESTORE'
        DEF   RESTOR
*
* CONTEXT RESTORE AT THE END OF A SUBROUTINE OR INTERRUPT.
* ASSUME THAT AR1 IS THE STACK POINTER AND AR1 = 117.
*
RESTOR LARP  AR1      ; CHANGE POINTER TO AR1,  AR1 = 117
        MAR   *+      ;                         AR1 = 118
*
* RESTORE AUXILIARY REGISTERS.
*
        LAR   AR1,*+  ; (118) --> AR1,          AR1 = 119
        LAR   AR0,*+  ; (119) --> AR0,          AR0 = 120
*
* RESTORE ALL FOUR LEVELS OF THE HARDWARE STACK.
*
        ZALS  *+      ; (120) --> ACC,          AR1 = 121
        PUSH          ; (120) --> BOS  (1),
        ZALS  *+      ; (121) --> ACC,          AR1 = 122
        PUSH          ; (121) --> STACK(2),
        ZALS  *+      ; (122) --> ACC,          AR1 = 123
        PUSH          ; (122) --> STACK(3),
        ZALS  *+      ; (123) --> ACC,          AR1 = 124
        PUSH          ; (123) --> TOS  (4),
*
* RESTORE THE T REGISTER.
*
        LT    *+      ; (124) --> T,            AR1 = 125
*
* RESTORE THE ACCUMULATOR.
*
        ZALS  *+      ; (125) --> ACCL,         AR1 = 126
        ADDH  *+      ; (126) --> ACCH,         AR1 = 127
*
* RESTORE THE STATUS REGISTER.
*
        LST   *+      ; (127) -> ST,            AR1 = 128
*
* RESTORE IS COMPLETE.
*
        EINT          ; ENABLE INTERRUPTS
        RET           ; RETURN TO CALLING ROUTINE
```

5.3 Program Control

To facilitate the use of the TMS320C1x in general-purpose high-speed processing, a variety of instructions are provided for software stack expansion, implementation of subroutine calls, addressing and loop control with auxiliary registers, and external branch control. Descriptions and examples of how to use these features are given in this section.

5.3.1 Software Stack Expansion

The TMS320C1x has a 12-bit Program Counter (PC) and a four-level hardware stack for PC storage. Provisions have been made on the TMS320C1x for extending the hardware stack into data memory. This is useful for deep subroutine nesting or stack overflow protection.

The hardware stack is accessible via the accumulator using the PUSH and POP instructions. The PUSH instruction pushes the 12 LSBs of the accumulator onto the top of stack (TOS). The POP instruction pops the TOS into the 12 LSBs of the accumulator. 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 the data RAM. From data RAM, it can easily be copied into off-chip program RAM using the TBLW instruction. In this way, the stack can be expanded to very large levels.

When the stack has four values stored on it and one or more values are to be put on the stack before any other values are popped off, a subroutine can be used to perform software stack expansion. Such a routine is illustrated in Example 5-7. In this example, the main program stores the stack starting location in memory in the auxiliary register and indicates to the subroutine whether to push data from memory onto the stack or pop data from the stack to memory. If a zero is loaded into the accumulator before calling the subroutine, the subroutine pushes data from memory to the stack. If a one is loaded into the accumulator, the subroutine pops data from the stack to memory.

A CALL instruction should be used to initiate execution of the software stack expansion routine. Since the CALL instruction uses the stack to save the program counter, the subroutine pops this value into the accumulator and saves it in a memory location. Then at the end of the subroutine, this value is reloaded into the accumulator, and the main program is reentered using the RET instruction. This prevents the calling routine program counter from being stored into a memory location. The subroutine in Example 5-7 uses the BANZ (branch on auxiliary register not zero) instruction to control all of its loops.

Example 5-7. Software Stack Expansion

```
* THIS ROUTINE EXPANDS THE STACK WHILE LETTING THE MAIN
* PROGRAM DETERMINE WHERE TO STORE THE STACK CONTENTS OR
* FROM WHERE TO RECOVER THEM.
*
LOC1    EQU   0
*
STACK   LARK  AR1,3       ; LOAD COUNTER
        LDPK  1           ; USE PAGE 1
        BNZ   PO          ; IF POPD IS NEEDED, GOTO PO
        POP               ; LOAD PC INTO ACCUMULATOR
        SACL  LOC1        ; STORE PC AT MEM LOCATION LOC1
*
P       LARP  0           ; USE AR0
        LAC   *+,AR1      ; LOAD ACCUMULATOR INTO MEMORY
        PUSH              ; PUT MEMORY ON STACK
        BANZ  P           ; BRANCH TO P UNTIL STACK IS FULL
        LAC   LOC1        ; LOAD PC INTO ACCUMULATOR
        PUSH              ; PUT RETURN ADDRESS ON STACK
        RET               ; RETURN TO MAIN PROGRAM
*
PO      POP               ; LOAD PC INTO ACCUMULATOR
        SACL  LOC1        ; SAVE PC INTO MEMORY
        MAR   *-          ; ALIGN STACK POINTER
*
PO1     LARP  0           ; USE AR0
        POP               ; PUT STACK IN ACCUMULATOR
        SACL  *-,0,AR1    ; STORE STACK IN MEMORY
        BANZ  PO1         ; BRANCH TO PO1 UNTIL SAVED
        MAR   *+          ; REALIGN STACK POINTER
        LAC   LOC1        ; LOAD ACCUMULATOR WITH PC
        PUSH              ; PUT RETURN ADDRESS ON STACK
        RET               ; RETURN TO MAIN PROGRAM
```

5.3.2 Subroutine Calls

When a subroutine call is made using the CALL or CALA instruction, the current contents of the program counter are stored on the top of the stack. At the end of the subroutine, a RET (return from subroutine) instruction pops the top of the stack to the program counter. The program then resumes execution at the instruction following the subroutine call.

In two circumstances, a level of stack must be reserved for the machine's use. First, the TBLR and TBLW instructions use one level of stack. Second, when interrupts are enabled, the PC is saved on the stack during the interrupt routine. If a system is designed to use both interrupts and a TBLR or TBLW instruction, only two levels of stack are available for nesting subroutine calls.

Subroutine calls can be nested deeper than two levels if the return address is removed from the stack and saved in data memory. The POP instruction moves the top of stack (TOS) into the accumulator and pops the stack up one level. The return address can then be stored in data memory until the end of the subroutine when it is put back into the accumulator. The PUSH instruction pushes the stack down one level and then moves the accumulator onto the TOS. Therefore, when the RET instruction is executed, the PC is updated with the return address. This procedure allows a second subroutine to be called inside the first subroutine without using another level of stack.

The POP and PUSH instructions can also be used to pass arguments to a subroutine. DATA directives following the subroutine call can be used to create a list of constants and/or variables to be passed to the subroutine. After the subroutine is called, the TOS points to the list of arguments following the CALL instruction. By moving the argument pointer from the TOS to the accumulator, the list of arguments can be read into data memory using the TBLR instruction. Between each TBLR instruction, the accumulator must be incremented by one to point to the next argument in the list. To create the return address, the argument pointer is incremented past the last element in the argument list. The PUSH instruction moves the return address onto the TOS, and the RET instruction updates the PC. Example 5-8 illustrates a call that passes two arguments to a subroutine.

Example 5-8. Two Arguments Passed to a Subroutine

```
* CLEAR BITS
*
* THIS ROUTINE CLEARS THE BITS OF A DATA WORD DESIGNATED BY
* A MASK. THE BITS SET TO ONE IN THE MASK INDICATE THE BITS
* IN THE DATA WORD TO BE CLEARED. ALL OTHER BITS REMAIN
* UNCHANGED. LOCATION ONE CONTAINS THE CONSTANT 1. MINUS
* CONTAINS A MASK INVERTER -1 OR >FFFF. TWO ARGUMENTS ARE
* PASSED TO THIS SUBROUTINE. THE CALLING SEQUENCE IS AS
* FOLLOWS:
*
*       CALL  CBITS
*       DATA  VALUE     ; 1ST ARGUMENT = ADDRESS OF DATA WORD
*       DATA  >0081     ; 2ND ARGUMENT = MASK
*
STATUS  EQU   0         ; STORE STATUS REGISTER HERE
XR0     EQU   126       ; TEMPORARY LOCATIONS
XR1     EQU   127
*
CBITS   SST   STATUS    ; SAVE STATUS
        LDPK  0         ; USE DATA PAGE 0
        SAR   AR0,XR0   ; SAVE AR0 IN TEMPORARY LOCATION
*
        POP             ; GET ADDRESS OF 1ST ARGUMENT IN ACC
        TBLR  XR1       ; STORE 1ST ARGUMENT IN TEMP LOCATION
        LAR   AR0,XR1   ; PUT 1ST ARGUMENT INTO AR0
        ADD   ONE       ; POINT TO 2ND ARGUMENT
        TBLR  XR1       ; 2ND ARGUMENT = MASK
        ADD   ONE       ; POINT TO RETURN ADDRESS
        PUSH            ; PUT RETURN ADDRESS ON TOS
*
        LARP  0
        LAC   XR1       ; LOAD MASK INTO ACCUMULATOR
        XOR   MINUS     ; INVERT MASK
        AND   *         ; CLEAR BITS
        SACL  *         ; STORE MODIFIED VALUE
*
        LAR   AR0,XR0   ; RESTORE AR0
        LDPK  1         : USE DATA PAGE 1
        LST   STATUS    ; RESTORE STATUS REGISTER
        RET             ; RETURN TO MAIN PROGRAM
```

Hardware stack allocation involves allocating the usage of the various stack levels for interrupts, subroutine calls, pipelined instructions, and the emulator (XDS). The TMS320C1x disables all interrupts when taking an interrupt trap. If interrupts are enabled more than one instruction before the return of the

interrupt service routine, the routine can also be interrupted, thus using another level of the hardware stack. This should be taken into consideration when managing the use of the stack.

When nesting subroutine calls, each call uses a level of the stack. The number of levels used by interrupts must be considered as well as the depth of the nesting of subroutines. Two possible allocations of the hardware stack levels are:

- 1 level reserved for interrupt service routines (ISR)
- 3 levels available for subroutine calls.

or:

- 1 level reserved for interrupt service routines (ISR)
- 2 levels available for subroutine calls
- 1 level available for TBLR/TBLW instructions.

5.3.3 Addressing and Loop Control with Auxiliary Registers

The two auxiliary registers on the TMS320C1x can be used either as pointers for indirect addressing or as loop counters. In the indirect addressing mode, the auxiliary register pointer (ARP) is used to determine which auxiliary register is selected. The LARP instruction sets the ARP equal to the value of the immediate operand. The value of the ARP can also be changed in the indirect addressing mode; the ARP is updated after the instruction has been executed.

The contents of the auxiliary register are interpreted as a data memory address when the indirect addressing mode is used. A sequential list of data can easily be accessed in the indirect mode by using the autoincrement/decrement feature of the auxiliary registers. The auxiliary register can also be used as a 9-bit counter (see Section 3.4.5). The MAR (modify auxiliary register and pointer) instruction allows the auxiliary register selected by the ARP to be incremented or decremented without implementing any other operation in parallel.

Three instructions (LARK, LAR, and SAR) either load or store a value into an auxiliary register, independent of the value of the ARP. The first operand in each of these instructions determines which auxiliary register is to be either loaded or stored. This operand does not affect the value of the ARP for subsequent instructions.

Example 5-9 illustrates using an auxiliary register in the indirect addressing mode to input data into a block of memory.

Example 5-9. Auxiliary Register Indirect Addressing

```
* THIS ROUTINE USES AN AUXILIARY REGISTER IN THE INDIRECT
* ADDRESSING MODE TO INPUT DATA INTO A BLOCK OF MEMORY.
*
       LARK  AR0,DATBLK ; INIT AR0 AS A POINTER TO DATBLK
*                       ; (AREA OF 8 WORDS IN DATA MEMORY)
       LARP  0          ; SELECT AR0
       LACK  8          ; INIT ACCUMULATOR AS A COUNTER
*
LOOP   IN    *+,PA0     ; INPUT DATA
       SUB   ONE        ; DECREMENT COUNTER (ONE = VALUE 1)
       BNZ   LOOP       ; REPEAT UNTIL COUNT = 0
```

An auxiliary register can also be used as a loop counter. The BANZ instruction tests and then decrements the auxiliary register selected by ARP. Because the test for zero occurs before the auxiliary register is decremented, the value loaded into the auxiliary register must be one less than the number of times the loop should be executed. The maximum number of loops that can be counted is 512, because only 9 bits of each auxiliary register are implemented as counters. A routine that inputs data and calculates a sum while the auxiliary register is used to count the number of loops is shown in Example 5-10. The accumulator contains the result.

Example 5-10. Auxiliary Register Loop Counting

```
* THIS ROUTINE USES AN AUXILIARY REGISTER TO COUNT THE
* NUMBER OF LOOPS.
*
       LARK  AR0,3      ; INITIALIZE AR0 AS A COUNTER
       LARP  0          ; SELECT AR0
       ZAC              ; CLEAR ACCUMULATOR
*
LOOP   IN    DATA1,PA2  ; INPUT DATA VALUE
       ADD   DATA1      ; ADD DATA TO ACCUMULATOR
       BANZ  LOOP       ; REPEAT LOOP FOUR TIMES
```

Both indirect addressing and loop counting can be performed at the same time to implement loops efficiently. If the data block is defined to start at location 0 in data memory, the same auxiliary register that is counting the number of loops can also be the pointer for indirect addressing, as shown below. Note that data locations 0 through 7 are loaded with input data.

```
        LARK  AR0,7    ; AR0 POINTS TO END OF DATA BLOCK
LOOP    IN    *,PA2    ; INPUT DATA VALUE
        BANZ  LOOP     ; REPEAT LOOP 8 TIMES
```

The data block does not have to start at zero if one auxiliary register is used for counting and the other register is used as a pointer. Example 5-11 illustrates how both auxiliary registers can be used at once.

Example 5-11. Auxiliary Register Pointing and Loop Counting

```
* THIS ROUTINE USES ONE AUXILIARY REGISTER FOR POINTING AND
* THE OTHER REGISTER FOR LOOP COUNTING.
*
        LARK  AR0,7        ; INITIALIZE AR0 AS A COUNTER
        LARK  AR1,DATBLK   ; AR0 POINTS TO START OF DATBLK
*                          ; (DATA MEMORY AREA)
        ZAC                ; CLEAR ACCUMULATOR
*
LOOP    LARP  1            ; POINT TO AR1
        ADD   *+,AR0       ; CALCULATE SUM OF DATA IN BLOCK
*                          ; POINT TO AR0
        BANZ  LOOP         ; REPEAT LOOP 8 TIMES
```

5.3.4 Computed GOTOs

Processing may be executed in a time-dependent (interrupt-driven) or a process-dependent (user-selected) way. Selecting the processing mode may depend on the result of a particular computation. A simple computed GOTO can be programmed in the TMS320C1x by using the CALA instruction. This instruction uses the contents of the accumulator as the direct address of the call. The address of the subroutine can be computed from a data value to determine which one of several routines will be executed. The return at the end of each of these routines causes program execution to resume with the instruction following the CALA command. Note that the CALA instruction uses a level of stack, because it is an indirect subroutine call, not just an indirect branch.

Example 5-12 illustrates how to compute a call to one of several routines. The subroutines are defined first, and then a table of branches to each subroutine is created. The main part of the program inputs a data value of 0, 1, or 2. The appropriate address in the table is calculated in the accumulator. An indirect subroutine call causes the proper branch in the table to be executed.

Example 5-12. Computed GOTO

```
* THIS ROUTINE COMPUTES AND EXECUTES A SUBROUTINE CALL.
*
ONE     EQU   126          ; STORE CONSTANT 1
VALUE   EQU   127          ; VALUE READ FROM PORT 4.
*
SUB1    IN    DAT1,PA0     ; INPUT DATA VALUE FROM PORT 0
        RET
*
SUB2    IN    DAT1,PA1     ; INPUT DATA VALUE FROM PORT 1
        RET
*
SUB3    IN    DAT1,PA2     ; INPUT DATA VALUE FROM PORT 2
        RET
*
TBL1    B     SUB1         ; CREATE TABLE OF BRANCHES TO EACH
        B     SUB2         ; SUBROUTINE DEFINED
        B     SUB3
*
START   LDPK  0            ; USE PAGE 0
        LACK  1            ; ACC = 1
        SACL  ONE          ; STORE 1 IN LOCATION ONE
        LT    ONE          ; LOAD T REGISTER WITH VALUE OF 1
        MPYK  TBL1         ; GET ADDRESS OF TABLE
        PAC
        IN    VALUE,PA4    ; INPUT DATA VALUE OF 0, 1, OR 2
        LT    VALUE        ; LOAD T REG WITH VALUE FROM PA4
        MPYK  2            ; CALCULATE OFFSET
        APAC
        CALA               ; GO TO DESIGNATED SUBROUTINE
        LAC   DAT1         ; RETURN HERE AFTER SUBROUTINE
         .
         .
         .
```

5.4 Memory Management

The TMS320C1x has a modified Harvard architecture in which program memory and data memory reside in two separate spaces. Therefore, the next instruction fetch can occur while the current instruction is fetching data and executing the operation. The concept of the Harvard architecture increases the speed of the device, but it requires the use of instructions to transfer a word between data memory and program memory.

Data memory consists of 144/256 words of 16-bit on-chip RAM. All non-immediate data operands reside within this RAM. Program memory consists of 1.5K/4K words of 16-bit on-chip ROM, of which 1524/4000 words are available for program use. On the EPROM versions, all 4096 words are available. Since there is no microprocessor mode of operation on the TMS 320C17 , all program memory resides on-chip in the ROM.

The TMS320C1x uses three forms of data memory addressing: direct, indirect, and immediate. Direct addressing uses the seven 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 auxiliary registers as the data memory address. Immediate addressing uses part of the instruction word for data rather than data RAM.

The structure of the TMS320C1x memory map can vary for each application (see Section 3.4.4 for memory maps). Instructions are provided for moving data and for moving constants into data memory. Explanations and examples are provided in this section.

5.4.1 Moving Data

The DMOV (data move) instruction allows a data word to be written into the next higher memory location in a single cycle without affecting the accumulator. If variables are placed in consecutive locations, a DMOV instruction can be used to move each of the variables before the next calculation is performed. For example, when implementing a digital filter, the variables in the equation represent the inputs and outputs at discrete times. This type of data structure is typically implemented as a shift register when the data at time t is shifted to the position previously occupied by the data at time t-1. If consecutive addresses in data memory correspond to consecutive time increments, then shifts can be accomplished simply by using the DMOV instruction to move the data item at location d to that corresponding to d+1.

The LTD instruction combines the data move operation with the LTA (load T register and accumulate previous product) instruction operations, performing the three operations in parallel. The operand of the instruction is loaded into the T register; the operand is also written into the next higher memory location; and the P register is added to the accumulator. When using the LTD instruction, the order of the multiply and accumulate operations becomes important because the data is being moved while the calculation is being performed. The oldest input variable must be multiplied by its constant and loaded into the accumulator first. Then the input, which is one time-unit delay less, is multiplied and accumulated. This process is repeated until the entire equation has been computed.

Example 5-13 illustrates the use of the LTD instruction to move input variables in memory as the results are calculated.

Example 5-13. Moving Data Using the LTD Instruction

```
* THE FOLLOWING EQUATION WILL BE IMPLEMENTED TO DEMONSTRATE
* THE USE OF THE LTD INSTRUCTION. AT THE END OF THE SUB-
* ROUTINE, LOCATION X1 IS AVAILABLE TO INPUT THE NEW SAMPLE.
*
*                Y = A*X3 + B*X2 + C*X1
*
* WHERE A, B, C, X1, X2, AND X3 ARE VALUES STORED AT THESE
* ADDRESSES.
*
X1      EQU   0         ; USE THESE MEMORY LOCATIONS
X2      EQU   1
X3      EQU   2
Y       EQU   3
A       EQU   127
B       EQU   126
C       EQU   125
*
START   ZAC             ; CLEAR ACCUMULATOR
        LDPK  0         ; USE PAGE 0
        LT    X3
        MPY   A         ; P = A*X3
        LTD   X2        ; T = X2, X2 --> X3, ACC = A*X3
        MPY   B         ; P = B*X2
        LTD   X1        ; T = X1, X1 --> X2, ACC = A*X3 + B*X2
        MPY   C         ; P = C*X1
        APAC            ; ACC = A*X3 + B*X2 + C*X1
        SACH  Y,1       ; Y = ACCH
```

The table below illustrates the effect on data memory after execution of the code in Example 5-13.

Data Memory	Before Code Execution	After Code Execution
>0	X1	X1
>1	X2	X1
>2	X3	X2

The DMOV feature is useful in implementing filters and convolution algorithms.

5.4.2 Moving Constants into Data Memory

Most signal processors have a separate memory space for storing constants. By allowing communication between data and program memory, the TMS320C1x is able to incorporate a constant memory capability with its program memory, thus allowing an efficient use of memory space. The portion of memory not used for storing constants is available for use as program space.

Five immediate instructions provide an efficient way to execute operations using constants. The LARP instruction changes the auxiliary register pointer, and the LDPK instruction changes the data page pointer. The LACK, LARK, and MPYK instructions allow constants to be used in calculations. LACK and LARK both require an unsigned operand with a magnitude no greater than eight bits. The MPYK instruction allows a 13-bit signed number as an operand.

A 16-bit value can be moved from program memory to data memory using the TBLR instruction. TBLR requires that the program memory address (the source) be in the accumulator, while the data memory address (the destination) is obtained from the operand of the instruction. This instruction is commonly used to look up values in a table in program memory. The address of the value in the table is computed in the accumulator before executing the instruction. TBLR then moves the value into data memory. TBLR is a three-cycle instruction and, therefore, takes longer than an immediate instruction. However, it has more flexibility since it operates on 16-bit constants.

Sometimes it is convenient to store data operands in program ROM or external memory, and then read them into the on-chip RAM as they are needed. Two means are available for doing this. First, the TBLR (table read) instruction can be used to transfer data from on-chip program ROM to on-chip data RAM. Second, off-chip data RAM can be addressed via the IN and OUT instructions. With some extra hardware, the IN and OUT instructions can be used to read and write from data RAM to large amounts of external storage addressed as a peripheral.

Data may also be transferred from data memory to program memory by means of the TBLW instruction. The IN and OUT instructions can be used to transfer data between the on-chip data memory and the I/O space (see Section 6.1).

Note that the TBLW (table write) instruction should not be used on the TMS 320C17 since this instruction transfers data from on-chip data RAM to external memory. The TMS 320C17 does not directly interface to external memory since the port address bits (PA2-PA0) are the only address lines external to the device.

Example 5-14 illustrates bringing the cosine value of a variable into data memory using the TBLR instruction. Note that if the address of COSINE is greater than 255, the address can be loaded into the accumulator by loading the T register with a one, multiplying by the constant COSINE, and transferring it from the P register into the accumulator.

Example 5-14. Moving a Constant into Data Memory Using the TBLR Instruction

```
* THIS ROUTINE USES THE TBLR INSTRUCTION TO BRING THE COSINE
* VALUE OF A VARIABLE INTO DATA MEMORY. A TABLE CONTAINING
* THE COSINE VALUES IS FIRST CREATED IN PROGRAM MEMORY.
*
COSINE      DATA
        .
        .
        .
START   IN    X,PA0
        LACK  COSINE    ; LOAD TABLE ADDRESS
        ADD   X         ; CALCULATE PROGRAM MEMORY ADDRESS
        TBLR  COSX      ; MOVE VALUE INTO DATA MEMORY
```

The following table shows the effect on data memory after the TBLR instruction has been executed in Example 5-14.

	Program Memory	Before TBLR Execution	After TBLR Execution
>(COSINE + X)		>02FF	>02FF
	Data Memory		
>COSX		>71F2	>02FF

Another method for transferring data from program memory into data memory uses the TBLR instruction. By using the TBLR instruction, a calculated, rather than predetermined, location of data in program memory may be specified for transfer. A routine using this approach is shown in Example 5-15.

Example 5-15. Moving Program Memory to Data Memory with TBLR

```
* THIS ROUTINE USES THE TBLR INSTRUCTION TO MOVE DATA VALUES
* FROM PROGRAM MEMORY INTO DATA MEMORY. BY USING THIS ROUTINE,
* THE PROGRAM MEMORY LOCATION IN THE ACCUMULATOR FROM WHICH
* DATA IS TO BE MOVED TO A SPECIFIC DATA MEMORY LOCATION CAN
* BE SPECIFIED. ASSUME THAT THE ACCUMULATOR CONTAINS THE
* ADDRESS IN PROGRAM MEMORY FROM WHICH TO TRANSFER THE DATA.
*
TABLE  LARP  1          ; USE AR1
       LARK  AR1,63     ; START FROM ADDRESS 63
*
LOOP   TBLR  *          ; MOVE DATA INTO DATA RAM
       BANZ  LOOP       ; TRANSFER 64 VALUES
       RET              ; RETURN TO CALLING PROGRAM
```

In cases where systems require that temporary storage be allocated in the program memory, TBLW can be used to transfer data from internal data memory to external program memory. The code in Example 5-16 demonstrates how this may be accomplished.

Example 5-16. Moving Internal Data Memory to Program Memory with TBLW

```
* THIS ROUTINE USES THE TBLW INSTRUCTION TO MOVE DATA VALUES
* FROM INTERNAL DATA MEMORY TO EXTERNAL PROGRAM MEMORY. THE
* CALLING ROUTINE MUST SPECIFY THE DESTINATION PROGRAM MEMORY
* ADDRESS IN THE ACCUMULATOR. ASSUME THAT THE ACCUMULATOR
* CONTAINS THE ADDRESS IN PROGRAM MEMORY INTO WHICH THE DATA
* IS TRANSFERRED.
*
TABLE  LARK  AR1,63    ; LOAD LOOP COUNT OF 64
       LARK  AR0,DAT1  ; LOAD STARTING ADDRESS
*
LOOP   LARP  AR0       ; USE AR0
       TBLW  *+,AR1    ; MOVE DATA TO EXTERNAL PROGRAM RAM
       BANZ  LOOP      ; DECREMENT AND CHECK IF DONE
       RET             ; RETURN TO CALLING PROGRAM
```

After the execution of the TBLW instruction, the following effect has occurred on program memory:

Program Memory	Before TBLW Execution	After TBLW Execution
>PROG1	>FF10	>1234
Data Memory		
>DAT1	>1234	>1234

The IN and OUT instructions are used to transfer data between the data memory and the I/O space, as shown in Example 5-17 and Example 5-18.

Example 5-17. Moving Data from I/O Space into Data Memory with IN

```
* THIS ROUTINE USES THE IN INSTRUCTION TO MOVE DATA VALUES
* FROM THE I/O SPACE INTO DATA MEMORY. DATA ACCESSED FROM
* I/O PORT 7 IS TRANSFERRED TO SUCCESSIVE MEMORY LOCATIONS
* ON DATA PAGE 0.
*
INPUT  LARK  AR0,32      ; SET UP LOOP COUNT
       LARK  AR1,DAT1    ; SET UP DESTINATION ADDRESS
*
LOOP   LARP  AR1         ; USE AR1
       IN    *+,PA7,AR0  ; MOVE DATA INTO DATA RAM
       BANZ  LOOP        ; DECREMENT AND CHECK IF DONE
       RET               ; RETURN TO CALLING PROGRAM
```

Example 5-18. Moving Data from Data Memory to I/O Space with OUT

```
* THIS ROUTINE USES THE OUT INSTRUCTION TO MOVE DATA VALUES
* FROM THE DATA MEMORY TO THE I/O SPACE. DATA IS TRANSFERRED
* TO I/O PORT 7 FROM SUCCESSIVE MEMORY LOCATIONS ON DATA
* PAGE 0.
*
OUTPUT LARK  AR0,32       ; SET UP LOOP COUNT
       LARK  AR1,DAT1     ; SET UP STARTING ADDRESS
*
LOOP   LARP  AR1          ; USE AR1
       OUT   *+,PA7,AR0   ; MOVE DATA INTO I/O SPACE
       BANZ  LOOP         ; DECREMENT AND CHECK IF DONE
       RET                ; RETURN TO CALLING PROGRAM
```

5.5 Logical and Arithmetic Operations

Although the TMS320C1x 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. 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.

5.5.1 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 hardware shifter and the logic instructions, AND, OR, and XOR. In Example 5-19, 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 5-19. Single-Bit Manipulation

```
*
* CLEAR BIT 5 OF DATA MEMORY LOCATION VALUE. MEMORY LOCATION
* ONE CONTAINS CONSTANT 1. MEMORY LOCATION MINUS CONTAINS -1
* OR >FFFF.
*
       LAC   ONE,5    ; ACC = >00000020
       XOR   MINUS    ; INVERT ACCUMULATOR; ACC = >0000FFDF
       AND   VALUE    ; BIT 5 OF VALUE IS ZEROED
       SACL  VALUE
*
* SET BIT 12 OF VALUE.
*
       LAC   ONE,12   ; ACC = >00001000
       OR    VALUE    ; BIT 12 OF VALUE
       SACL  VALUE
*
* TEST BIT 3 OF VALUE.
*
       LAC   ONE,3    ; ACC = >00000008
       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 5-20, the six low-order bits in the word VALUE are cleared if MASK contains the value 63.

Example 5-20. 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 >FFFF.
*
      LAC   MASK    ; ACC = >0000003F
      XOR   MINUS   ; INVERT ACCUMULATOR; ACC = >0000FFC0
      AND   VALUE   ; CLEAR LOWER SIX BITS
      SACL  VALUE
```

5.5.2 Overflow Management

The TMS320C1x 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 simulates 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 (>7FFFFFFF) if the overflowed number is positive, or the largest negative value (>80000000) 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 5-21, the accumulator saturates to >7FFFFFFF or the largest positive value. The BV instruction also clears the OV bit.

Example 5-21. Overflow Management

```
* THE ACCUMULATOR WILL SATURATE TO THE HIGHEST POSITIVE VALUE
* WHEN OVERFLOW OCCURS. THE ACCUMULATOR CONTAINS >7FFFF423.
* MEMORY LOCATION A CONTAINS >74ED. MEMORY LOCATION B
* CONTAINS >67AF.
*
      SOVM          ; SET OVERFLOW MODE
      LT    A       ; T = >74ED
      MPY   B       ; P = >2F5B4903
      APAC          ; ACC = >7FFFFFFF
      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	>7FFFF423	>7FFFFFFF

5.5.3 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 TMS320C1x 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 5.5.5).

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 5-22 performs an arithmetic right-shift of 7 on a 16-bit number in the accumulator.

Example 5-22. 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	>FFFFA452	>FFFFFF48

A logical right-shift of 4 on a 32-bit number stored in the accumulator is shown in Example 5-23. The 32-bit results of the shift are then stored in data memory. In this example, the accumulator initially contains the hexadecimal number, >9D84C1B2. The variables, SHIFTH and SHIFTL, will receive the high word (>09D8) and low word (>4C1B) of the shifted results.

Example 5-23. Logical Right-Shift

```
*
* SHIFT THE LOWER WORD. MEMORY LOCATION MINUS CONTAINS -1
* OR >FFFF.
*
       SACH  SHIFTH     ; SHIFTH = >9D84    INITIAL VALUES
       SACL  SHIFTL     ; SHIFTL = >C1B2
       LAC   SHIFTL,12  ; ACC = >FC1B2000
       SACH  SHIFTL     ; SHIFTL = >FC1B
       LAC   MINUS,12   ; ACC = >FFFFF000
       XOR   MINUS      ; ACC = >FFFF0FFF
       AND   SHIFTL     ; ACC = >00000C1B
*
* SHIFT THE UPPER WORD.
*
       ADD   SHIFTH,12  ; ACC = >F9D84C1B
       SACL  SHIFTL     ; SHIFTL = >4C1B    FINAL LOW VALUE
       SACH  SHIFTH     ; SHIFTH = >F9D8
       LAC   MINUS,12   ; ACC = >FFFFF000
       XOR   MINUS      ; ACC = >FFFF0FFF
       AND   SHIFTH     ; ACC = >000009D8
       SACL  SHIFTH     ; SHIFTH = >09D8    FINAL HIGH VALUE
```

The accumulator is affected before and after the code execution as follows:

	Before Code Execution	After Code Execution
ACC	>9D84C1B2	>09D84C1B

An arithmetic right-shift of 4 can be implemented using the same routine as shown above, except with the last four lines omitted.

5.5.4 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.

5.5.5 Multiplication

The TMS320C1x 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 TMS320C1x 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 5-24, Example 5-25, and Example 5-26.

Example 5-24. Fraction × Fraction (Q15 × Q15 = Q30)

```
                    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 = >4000 (0.5 in Q15)
MPY     OP2   ; OP2 = >4000 (0.5 in Q15)
PAC
SACH    ANS,1 ; ANS = >2000 (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 = >4000 (0.5  in Q15)
MPYK    2048  ; OP2 = >0800 (0.5  in Q12)
PAC
SACH    ANS,4 ; ANS = >2000 (0.25 in Q15)
```

Example 5-25. Integer × Integer (Q0 × Q0 = Q0)

```
                    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 = >0011 ( 17 in Q0)
MPY     OP2   ; OP2 = >FFFB ( -5 in Q0)
PAC
SACH    ANS   ; ANS = >FFAB (-85 in Q0)
```

Example 5-26. Mixed Notation (Q14 x Q14 = Q28)

```
                    0110000000000000  = 1.50 in Q14
                x   0011000000000000  = 0.75 in Q14
-------------------------------------------------
0001 001000000000  0000000000000000  = 1.125 in Q28
     └ binary point
```

The maximum magnitude of a Q14 number is just under 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 = >6000 (1.5   in Q14)
MPY     OP2       ; OP2 = >3000 (0.75  in Q14)
PAC
SACH    ANS,1     ; ANS = >2400 (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 5-27. The maximum error generated in this example is plus one-half LSB whether ANS is positive or negative.

Example 5-27. Rounding Technique for Multiplication

```
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 program memory value. Example 5-28 shows an implementation of multiplies and accumulates using the LTA-MPY instruction pair.

Example 5-28. Multiply and Accumulate Using the LTA-MPY Instruction Pair

```
*
*                    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                         1
      .                      2N                          2N
      .                                                  
      .                                                  
     LTA   DN         1                         1
     MPY   CN         1                         1
     APAC             1        2 + 2N           1          2 + 2N
```

5.5.6 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. For each subtract not producing a negative answer, a one is put in the LSB of the quotient and then shifted. 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 )00000000000100001
                        -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 5-29 and Example 5-30). Note that the next instruction after SUBC cannot use the accumulator.

Example 5-29. Using SUBC Where Numerator < Denominator

```
* THIS ROUTINE DIVIDES TWO BINARY, TWO'S-COMPLEMENT NUMBERS
* OF ANY SIGN WHERE THE NUMERATOR IS LESS THAN THE
* DENOMINATOR.
*
*                BEFORE        AFTER
*             INSTRUCTION  INSTRUCTION
*
* 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 5-30. Using SUBC Where Accuracy of Quotient Specified

```
* THIS ROUTINE DIVIDES TWO BINARY, TWO'S-COMPLEMENT NUMBERS
* OF ANY SIGN, SPECIFYING THE FRACTIONAL ACCURACY OF THE
* QUOTIENT (FRAC).
*
*                BEFORE        AFTER
*             INSTRUCTION  INSTRUCTION
*
* 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
```

5.5.7 Addition

Both operands in division 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 5-31 and Example 5-32 illustrate two approaches to adding 16-bit numbers.

Example 5-31. 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 5-32. Adjusted Binary Point to Maintain 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.

5.5.8 Floating-Point Arithmetic

Although the TMS320C1x 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 TMS320C1x 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 TMS320C1x.

Operations in the TMS320C1x 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 used in Q15 format. Q format is a number representation commonly used when performing operations on non-integer 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.

5.6 Application-Oriented Operations

The TMS320C1x has been designed to provide efficient implementations of many common digital signal processing algorithms. Its features provide solutions to numerically intensive problems usually characterized by multiply and accumulate operations. Some device-specific features that aid in the implementation of specific algorithms on the TMS320C1x include companding, filtering, Fast Fourier Transforms (FFT), and PID control. These applications require I/O performed either in parallel or serial.

5.6.1 Companding

In the area of telecommunications, one of the primary concerns is the I/O bandwidth in the communications channel. One way to minimize this bandwidth is by companding (COMpress/exPAND). Companding is defined by two international standards, A-law and μ-law, both based on the compression of the equivalent of 13 bits of dynamic range into an 8-bit code. The standard employed in the United States and Japan is μ-law companding. The European standard is referred to as A-law companding. Detailed descriptions and code examples of μ-law and A-law companding are presented in "Companding Routines for the TMS32010/TMS32020," an application report included in the book, *Digital Signal Processing Applications with the TMS320 Family*.

The technique of companding allows the digital sample information corresponding to a 13-bit dynamic range to be transmitted as 8-bit data. For processing in the TMS320C1x, it is necessary to convert the 8-bit logarithmic data to a 16-bit linear format. Prior to output, the linear result must be converted to the compressed or companded format. On the TMS32010/C10/C15, companding must be performed in software using conversion routines. On-chip companding hardware on the TMS320C17/E17 implements these functions.

Software routines for μ-law and A-law companding, flowcharts, companding algorithms, and detailed descriptions are provided in the application report on companding routines in the book, *Digital Signal Processing Applications with the TMS320 Family*. The algorithm space and time requirements for μ-law and A-law companding on the TMS32010/C10/C15 are given in Table 5-2.

Table 5-2. Program Space and Time Requirements for μ-/A-Law Companding

FUNCTION	WORDS OF MEMORY		PROGRAM CYCLES		TIME REQD†
	Program	Data	Initialization	Loop‡	μs
μ-Law:					
Compression	105	13	17	40	8.0
Expansion	46	8	6	23	4.6
A-Law:					
Compression	97	11	14	36	7.2
Expansion	48	7	4	25	5.0

†Assuming initialization
‡Worst case

Four modes are available for the on-chip companding hardware operation on the TMS320C17/E17: serial encode, serial decode, parallel encode, and parallel decode. On the TMS320C17/E17, the companding hardware converts between two's-complement or sign-magnitude format and the companded format.

In the serial encode mode, transmitted data is encoded according to either μ-law or A-law format. In the serial decode mode, received data is decoded to sign-magnitude format according to the specified companding law.

In the parallel modes, either the encoder or decoder is enabled, and then data written to port 1 is compressed or expanded. To convert sign-magnitude linear PCM to 8-bit log PCM, the encoder is enabled for parallel operation, and the sample is written to port 1. An IN instruction from port 1 returns the converted 8-bit log PCM value. To convert 8-bit log PCM to sign-magnitude linear PCM, the decoder is enabled for parallel operation, and the 8-bit sample is written to port 1. The expanded sign-magnitude value is returned on the IN instruction from port 1. Enabling both the encoder and decoder is an undesirable state and should be avoided for the parallel mode. Care should be taken to have one OUT-IN instruction sequence to port 1 for each data sample, because the execution of two OUT instructions to port 1 in succession pushes the first sample into the transmit register TR1, preventing access for read purposes. OUT instructions to port addresses 2 through 7 do not affect the serial-port operation.

When the companding hardware converts to sign-magnitude data, it must be converted to two's-complement notation for computation in the microcomputer. Sign-magnitude notation consists of a sign bit in the MSB: a zero indicating a positive value, and a one indicating a negative number. All bits between the sign bit and the MSB of the data value are set to zero. For conversions between μ-law and sign-magnitude linear PCM, the hexadecimal value >1FFF represents the most positive value of 8191 and the value >9FFF represents the most negative value of -8191. For conversions between A-law and sign-magnitude linear PCM, the hexadecimal value >0FFF represents the most positive value of 4095 and the value >8FFF represents the most negative value of -4095.

5.6.2 FIR/IIR Filtering

Digital filters are a common requirement for digital signal processing systems. The filters fall into two basic categories: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. For either category of filter, the coefficients of the filter (weighting factors) may be fixed or adapted during the course of the signal processing. The theory and implementation of digital filters has been presented and discussed in an application report, "Implementation of FIR/IIR Filters with the TMS32010/TMS32020," included in the book, *Digital Signal Processing Applications with the TMS320 Family*.

IIR filters benefit from the fast instruction cycle time of the TMS320C1x. IIR filters typically require fewer multiply/accumulates. Correspondingly, the amount of data memory for samples and coefficients is not usually the limiting factor. Because of sensitivity to quantization of the coefficients themselves, IIR filters are usually implemented in cascaded second-order sections. This translates to code consisting of LTD-MPY instruction pairs. Example 5-35 provides an implementation of a second-order IIR filter.

Example 5-35. Implementing an IIR Filter

```
*
* THE FOLLOWING EQUATIONS ARE USED TO IMPLEMENT AN IIR FILTER:
*
*     d(n) = x(n)    + d(n-1)a1 + d(n-2)a2
*     y(n) = d(n)b0 + d(n-1)b1 + d(n-2)b2
*
START  IN    XN,PA0  ; INPUT NEW VALUE XN
       LAC   XN,15   ; LOAD ACCUMULATOR WITH XN
*
       LT    DNM1
       MPY   A1
*
       LTD   DNM2
       MPY   A2
*
       APAC
       SACH  DN,1    ; d(n) = x(n) + d(n-1)a1 + d(n-2)a2
       ZAC
       MPY   B2
*
       LTD   DNM1
       MPY   B1
*
       LTD   DN
       MPY   B0
*
       APAC
       SACH  YN,1    ; y(n) = d(n)b0 + d(n-1)b1 + d(n-2)b2
       OUT   YN,PA1  ; YN IS THE OUTPUT OF THE FILTER
```

FIR filters also benefit from the fast instruction cycle time. In addition, an FIR filter requires many more multiply/accumulates than does the IIR filter with equivalent sharpness at the cutoff frequencies and with distortion and attenuation in the passbands and stopbands. The TMS320C1x helps solve this problem by making longer filters feasible to implement. The TMS320C15/C17 has expanded data memory of 256 words, thus allowing additional coeffi-

cients and samples to be stored for longer-length filters. Example 5-36 provides an implementation of a fourth-order (4 taps) FIR filter. Each tap consists of a LTD-MPY instruction pair, uses two data memory locations, and takes two instruction cycles to execute.

Example 5-36. Implementing an FIR Filter

```
*
* THE FOLLOWING EQUATION IS USED TO IMPLEMENT AN FIR FILTER:
*
*    y(n)=[Ax(n-1)+Cx(n-3)+Dx(n-4)]* 2**-16
*
START  IN    X1,PA0  ; INPUT SAMPLE
       ZAC
*
       LT    X4      ; x(n-4)
       MPY   D
*
       LTD   X3      ; ACC=Dx4; x(n-4))=x(n-3)
       MPY   C
*
       LTD   X2      ; ACC=Dx4+Cx3; x(n-3))=x(n-2)
       MPY   B
*
       LTD   X1      ; ACC=Dx4+Cx3+Bx2; x(n-2))=x(n-1)
       MPY   A
*
       APAC          ; ACC=Dx4+Cx3+Bx2+Ax1
       SACH  Y,1
       OUT   Y,PA1   ; OUTPUT RESULTS
       B     START
```

An implementation of an FIR filter using straightline code was shown in Example 5-36. For longer-length FIR filters, straightline code may require larger program memory size. Depending on the system constraints, the designer may choose to reduce program memory size by using looped code. However, straightline code will run much faster than looped versions. The design tradeoff should be carefully considered by the design engineer.

5.6.3 Adaptive Filtering

With FIR or IIR filtering, the filter coefficients may be fixed or adapted. If the coefficients are adapted or updated with time, then another factor impacts the computational capacity. This factor is the requirement to adapt each of the coefficients, usually with each sample. A means of adapting the coefficients is the Least-Mean-Square (LMS) algorithm given by the following equation:

$$b_k(i+1) = b_k(i) + 2B\ e(i)\ x(i-k)$$

$$\text{where } e(i) = x(i) - y(i)$$

$$\text{and } y(i) = \sum_{k=0}^{N-1} b_k\ x(i-k)$$

Quantization errors in the updated coefficients can be minimized if the result is obtained by rounding rather than truncating. For each coefficient in the filter at a given point in time, the factor 2B e(i) is a constant. This factor can then be computed once and stored in the T register for each of the updates. Thus, the computational requirement has become one multiply/accumulate plus rounding. The adaptation of each coefficient is five instructions corresponding to five clock cycles. This is shown in the instruction sequence as follows:

```
LARK  AR0,LASTAP    ; POINT TO DATA SAMPLE
LARK  AR1,COEFFD    ; POINT TO COEFFICIENTS
LARP  AR0
LT    ERRF          ; errf = 2B*e(i)
 .
 .
 .
MPY   *-,AR1        ; P = 2B*e(i)*X(i-0)
ZALH
APAC                ; b0(i+1) = b0(1) + P
ADD   ONE,15        ; ROUND
SACH  *+,0,AR0      ; STORE b0(i+1)
 .
 .
 .
```

Example 5-37 shows a routine to filter a signal and update the coefficients. The total execution time of the routine is 30 + 7n where n is the filter length. Data and program memory requirements are 5 + 2n words and 28 + 7n words, respectively. The filter length for adaptive filters is restricted both by execution time and memory. There is obviously more processing to be completed per sample due to the adaptation, and the size of the on-chip data RAM limits the number of coefficients and data samples that can be stored.

Another way to perform adaptive filtering is discussed in an application report, "Digital Voice Echo Canceller with a TMS32020," included in the book, *Digital Signal Processing Applications with the TMS320 Family*.

Example 5-37. 32-Tap Adaptive Filter

```
        TITL  'ADAPTIVE FILTER'
        DEF   ADPFIR
        DEF   X,Y
*
* THIS 32-TAP ADAPTIVE FILTER USES PAGE 0 FOR COEFFICIENTS
* AND DATA SAMPLES. THE NEWEST INPUT SHOULD BE IN MEMORY
* LOCATION X WHEN CALLED. THE OUTPUT WILL BE IN MEMORY
* LOCATION Y WHEN RETURNED.
*
ONE     EQU   120          ; CONSTANT ONE
BETA    EQU   121          ; ADAPTATION CONSTANT * 2
ERR     EQU   122          ; SIGNAL ERROR
ERRF    EQU   123          ; ERROR FUNCTION
Y       EQU   124          ; FILTER OUTPUT
X       EQU   125          ; NEWEST DATA SAMPLE
FRSTAP  EQU    32          ; NEXT NEWEST DATA SAMPLE
LASTAP  EQU    63          ; OLDEST DATA SAMPLE
COEFFD  EQU     0          ; START OF COEFFICIENT TABLE
*
* FINITE IMPULSE  RESPONSE (FIR) FILTER.
*
ADPFIR  LDPK  0            ; USE DATA PAGE 0
        LARK  AR1,COEFFD   ; LOAD POINTER FOR COEFF TABLE
        LARK  AR0,LASTAP   ; LOAD POINTER FOR DATA SAMPLES
        MPYK  0            ; CLEAR THE P REGISTER
        LAC   ONE,14       ; LOAD OUTPUT ROUNDING BIT
        LARP  AR0
*
* DO 32 TAPS.
*
FIR     LT    *-,AR1       ; LOAD T REG WITH OLDEST SAMPLE
        MPY   *+,AR0       ; MULTIPLY WITH LAST COEFFICIENT
*
        LTD   *-,AR1       ; LOAD NEXT SAMPLE
        MPY   *+,AR0       ; MULTIPLY WITH NEXT COEFFICIENT
*
        LTD   *-,AR1       ; LOAD NEXT SAMPLE
        MPY   *+,AR0       ; MULTIPLY WITH NEXT COEFFICIENT
         .
         .
         .
        LTD   *-,AR1       ; LOAD LAST SAMPLE
        MPY   *+,AR0       ; MULTIPLY WITH LAST COEFFICIENT
*
        APAC
        SACH  Y,1          ; STORE FILTER OUTPUT
        ZAC
        SUB   Y            ; ACC = -y(i)
        ADD   X            ; ADD THE NEWEST INPUT
        SACL  ERR          ; err(i) = x(i) - y(i)
*
* LMS ADAPTATION OF FILTER COEFFICIENTS.
*
        LT    ERR
        MPY   BETA
        PAC                ; errf(i) = 2*beta*err(i)
        ADD   ONE,14       ; ROUND THE RESULT
        SACH  ERRF,1
        LAC   X
        SACL  FRSTAP       ; INCLUDE NEWEST SAMPLE
```

```
*
        LARK  AR0,LASTAP  ; POINT TO DATA SAMPLE
        LARK  AR1,COEFFD  ; POINT TO COEFFICIENTS
*
        LT    ERRF        ; KEEP ERRF IN T REGISTER
*
ADAPT   MPY   *-,AR1      ; P = 2*beta*err(i)*x(i-31)
        ZALH  *
        APAC              ; b31(i+1) = b31(i) + P
        ADD   ONE,15      ; ROUND
        SACH  *+,0,AR0    ; STORE b31(i+1)
*
        MPY   *-,AR1      ; P = 2*beta*err(i)*x(i-30)
        ZALH  *
        APAC              ; b30(i+1) = b30(i) + P
        ADD   ONE,15      ; ROUND
        SACH  *+,0,AR0    ; STORE b30(i+1)
*
        MPY   *-,AR1      ; P = 2*beta*err(i)*x(i-29)
        ZALH  *
        APAC              ; b29(i+1) = b29(i) + P
        ADD   ONE,15      ; ROUND
        SACH  *+,0,AR0    ; STORE b29(i+1)
         .
         .
         .
        MPY   *-,AR1      ; P = 2*beta*err(i)*x(i-0)
        ZALH  *
        APAC              ; b0(i+1) = b0(i) + P
        ADD   ONE,15      ; ROUND
        SACH  *+,0,AR0    ; STORE b0(i+1)
*
        RET               ; RETURN TO MAIN PROGRAM
```

5.6.4 Fast Fourier Transforms (FFT)

Fourier transforms are another important tool often used in digital signal processing systems. The purpose of the transform is to convert information from the time domain to the frequency domain. The inverse Fourier transform converts information back to the time domain from the frequency domain. Implementations of Fourier transforms that are computationally efficient are known as Fast Fourier Transforms (FFTs). The theory and implementation of FFTs has been discussed in the book, *DFT/FFT and Convolution Algorithms*, by Burrus and Parks, published by John Wiley and Sons. The book also contains a large number of sample TMS32010 and FORTRAN programs to implement DFT/FFT algorithms. The TMS320C1x reduces the execution time of all FFTs by virtue of its single-cycle instruction time.

Example 5-38 consists of some of the macros used in the implementation of FFTs. Example 5-39 provides the code for an 8-point DIT (decimation in time) FFT. The code has been structured into a number of macro calls, including a macro for bit reversal.

Example 5-38. FFT Macros

```
COMBO   $MACRO R1,I1,R2,I2,R3,I3,R4,I4
*
* CALCULATE PARTIAL TERMS FOR R3, R4, I3, AND I4.
*
        LAC    :R3:,14    ACC    := (1/4)(R3)
        ADD    :R4:,14    ACC    := (1/4)(R3+R4)
        SACH   :R3:,1     R3     := (1/2)(R3+R4)
        SUB    :R4:,15    ACC    := (1/4)(R3+R4)-(1/2)(R4)
        SACH   :R4:,1     R4     := (1/2)(R3-R4)
        LAC    :I3:,14    ACC    := (1/4)(I3)
        ADD    :I4:,14    ACC    := (1/4)(I3+I4)
        SACH   :I3:,1     I3     := (1/2)(I3+I4)
        SUB    :I4:,15    ACC    := (1/4)(I3+I4)-(1/2)(I4)
        SACH   :I4:,1     I4     := (1/2)(I3-I4)
*
* CALCULATE PARTIAL TERMS FOR R2, R4, I2, AND I4.
*
        LAC    :R1:,14    ACC    := (1/4)(R1)
        ADD    :R2:,14    ACC    := (1/4)(R1+R2)
        SACH   :R1:,1     R1     := (1/2)(R1+R2)
        SUB    :R2:,15    ACC    := (1/4)(R1+R2)-(1/2)(R2)
        ADD    :I4:,15    ACC    := (1/4)[(R1-R2)+(I3-I4)]
        SACH   :R2:       R2     := (1/4)[(R1-R2)+(I3-I4)]
        SUBH   :I4:       ACC    := (1/4)[(R1-R2)-(I3-I4)]
        DMOV   :R4:       I4     := R4 = (1/2)(R3-R4)
        SACH   :R4:       R4     := (1/4)[(R1-R2)-(I3-I4)]
        LAC    :I1:,14    ACC    := (1/4)(I1)
        ADD    :I2:,14    ACC    := (1/4)(I1+I2)
        SACH   :I1:,1     I1     := (1/2)(I1+I2)
        SUB    :I2:,15    ACC    := (1/4)(I1+I2)-(1/2)(I2)
        SUB    :I4:,15    ACC    := (1/4)[(I1-I2)-(I3-I4)]
        SACH   :I2:       I2     := (1/4)[(I1-I2)-(I3-I4)]
        ADDH   :I4:       ACC    := (1/4)[(I1-I2)+(I3-I4)]
        SACH   :I4:       I4     := (1/4)[(I1-I2)+(I3-I4)]
*
* CALCULATE PARTIAL TERMS FOR R1, R3, I1, AND I3.
*
        LAC    :R1:,15    ACC    := (1/4)(R1+R2)
        ADD    :R3:,15    ACC    := (1/4)[(R1+R2)+(R3+R4)]
        SACH   :R1:       R1     := (1/4)[(R1+R2)+(R3+R4)]
        SUBH   :R3:       ACC    := (1/4)[(R1+R2)-(R3+R4)]
        SACH   :R3:       R3     := (1/4)[(R1+R2)-(R3+R4)]
        LAC    :I1:,15    ACC    := (1/4)(I1+I2)
        ADD    :I3:,15    ACC    := (1/4)[(I1+I2)+(I3+I4)]
        SACH   :I1:       I1     := (1/4)[(I1+I2)+(I3+I4)]
        SUBH   :I3:       ACC    := (1/4)[(I1+I2)-(I3+I4)]
        SACH   :I3:       I3     := (1/4)[(I1+I2)-(I3+I4)]
        $END
*
* MACRO FOR INPUT BIT REVERSAL.
*
BITREV  $MACRO PR,PI,QR,QI
        ZALH   :PR:
        ADDS   :QR:
        SACL   :PR:
        SACH   :QR:
        ZALH   :PI:
        ADDS   :QI:
        SACL   :PI:
        SACH   :QI:
        $END
```

```
*
ZERO    $MACRO PR,PI,QR,QI
*
* CALCULATE Re(P+Q) AND Re(P-Q)
*
        LAC     :PR:,15     ACC     := (1/2)(PR)
        ADD     :QR:,15     ACC     := (1/2)(PR+QR)
        SACH    :PR:        PR      := (1/2)(PR+QR)
        SUBH    :QR:        ACC     := (1/2)(PR+QR)-(QR)
        SACH    :QR:        QR      := (1/2)(PR-QR)
*
* CALCULATE Im(P+Q) AND Im(P-Q)
*
        LAC     :PI:,15     ACC     := (1/2)(PI)
        ADD     :QI:,15     ACC     := (1/2)(PI+QI)
        SACH    :PI:        PR      := (1/2)(PI+QI)
        SUBH    :QI:        ACC     := (1/2)(PI+QI)-(QI)
        SACH    :QI:        QR      := (1/2)(PI-QI)
        $END
*
PIBY4   $MACRO PR,PI,QR,QI,W
*
        LT      :W:         T REG   := W=COS(PI/4)=SIN(PI/4)
        LAC     :QI:,14     ACC     := (1/4)(QI)
        SUB     :QR:,14     ACC     := (1/4)(QI-QR)
        SACH    :QI:,1      QI      := (1/2)(QI-QR)
        ADD     :QR:,15     ACC     := (1/4)(QI+QR)
        SACH    :QR:,1      QR      := (1/2)(QI+QR)
        LAC     :PR:,14     ACC     := (1/4)(PR)
        MPY     :QR:        P REG   := (1/4)(QI+QR)*W
        APAC                ACC     := (1/4)[PR+(QI+QR)*W]
        SACH    :PR:,1      PR      := (1/2)[PR+(QI+QR)*W]
        SPAC                ACC     := (1/4)(PR)
        SPAC                ACC     := (1/4)[PR-(QI+QR)*W]
        SACH    :QR:,1      QR      := (1/2)[PR-(QI+QR)*W]
        LAC     :PI:,14     ACC     := (1/4)(PI)
        MPY     :QI:        P REG   := (1/4)(QI-QR)*W
        APAC                ACC     := (1/4)[PI+(QI-QR)*W]
        SACH    :PI:,1      PI      := (1/2)[PI+(QI-QR)*W]
        SPAC                ACC     := (1/4)(PI)
        SPAC                ACC     := (1/4)[PI-(QI-QR)*W]
        SACH    :QI:,1      QI      := (1/2)[PI-(QI-QR)*W]
        $END
*
PIBY2   $MACRO PR,PI,QR,QI
*
* CALCULATE Re(P+jQ) AND Re(P-jQ)
*
        LAC     :PI:,15     ACC     := (1/2)(PI)
        SUB     :QR:,15     ACC     := (1/2)(PI-QR)
        SACH    :PI:        PI      := (1/2)(PI-QR)
        ADDH    :QR:        ACC     := (1/2)(PI-QR)+(QR)
        SACH    :QR:        QR      := (1/2)(PI+QR)
```

```
*
* CALCULATE Im(P+jQ) AND Im(P-jQ)
*
        LAC     :PR:,15     ACC     := (1/2)(PR)
        ADD     :QI:,15     ACC     := (1/2)(PR+QI)
        SACH    :PR:        PR      := (1/2)(PR+QI)
        SUBH    :QI:        ACC     := (1/2)(PR+QI)-(QI)
        DMOV    :QR:        QR      --> QI
        SACH    :QR:        QR      := (1/2)(PR-QI)
        $END
*
PI3BY4  $MACRO PR,PI,QR,QI,W
*
        LT      :W:         T REG   := W=COS(PI/4)=SIN(PI/4)
        LAC     :QI:,14     ACC     := (1/4)(QI)
        SUB     :QR:,14     ACC     := (1/4)(QI-QR)
        SACH    :QI:,1      QI      := (1/2)(QI-QR)
        ADD     :QR:,15     ACC     := (1/4)(QI+QR)
        SACH    :QR:,1      QR      := (1/2)(QI+QR)
        LAC     :PR:,14     ACC     := (1/4)(PR)
        MPY     :QI:        P REG   := (1/4)(QI-QR)*W
        APAC                ACC     := (1/4)[PR+(QI-QR)*W]
        SACH    :PR:,1      PR      := (1/2)[PR+(QI-QR)*W]
        SPAC                ACC     := (1/4)(PR)
        SPAC                ACC     := (1/4)[PR-(QI-QR)*W]
        MPY     :QR:        P REG   := (1/4)(QI+QR)*W
        SACH    :QR:,1      QR      := (1/2)[PR-(QI-QR)*W]
        LAC     :PI:,14     ACC     := (1/4)(PI)
        SPAC                ACC     := (1/4)[PI-(QI+QR)*W]
        SACH    :PI:,1      PI      := (1/2)[PI-(QI+QR)*W]
        APAC                ACC     := (1/4)(PI)
        APAC                ACC     := (1/4)[PI+(QI+QR)*W]
        SACH    :QI:,1      QI      := (1/2)[PI+(QI+QR)*W]
        $END
```

Example 5-39. An 8-Point DIT FFT

```
* THIS ROUTINE IMPLEMENTS AN 8-POINT DIT FFT. ASSUME THAT
* TWIDDLE FACTOR = W VALUE STORED IN MEMORY LOCATION W.
*
X0R     EQU     00
X0I     EQU     01
X1R     EQU     02
X1I     EQU     03
X2R     EQU     04
X2I     EQU     05
X3R     EQU     06
X3I     EQU     07
X4R     EQU     08
X4I     EQU     09
X5R     EQU     10
X5I     EQU     11
X6R     EQU     12
X6I     EQU     13
X7R     EQU     14
X7I     EQU     15
W       EQU     16
WVALUE  EQU     >5A82       ; VALUE FOR SIN(45) OR COS(45)
*
* INITIALIZE FFT PROCESSING. ASSUME TWIDDLE FACTOR =
* W VALUE STORED IN MEMORY LOCATION W.
*
FFT     ROVM                ; RESET OVERFLOW MODE
        LDPK    0           ; SET DATA PAGE POINTER TO 0
*
* BIT-REVERSED INPUT SAMPLES.
*
        BITREV  X1R,X1I,X4R,X4I
        BITREV  X3R,X3I,X6R,X6I
*
* FIRST AND SECOND STAGES COMBINED WITH DIVIDE-BY-4
* INTERSTAGE SCALING.
*
        COMBO   X0R,X0I,X1R,X1I,X2R,X2I,X3R,X3I,
        COMBO   X4R,X4I,X5R,X5I,X6R,X6I,X7R,X7I.
*
* THIRD STAGE WITH DIVIDE-BY-2 INTERSTAGE SCALING.
*
        ZERO    X0R,X0I,X4R,X4I
        PIBY4   X1R,X1I,X5R,X5I,W
        PIBY2   X2R,X2I,X6R,X6I
        PI3BY4  X3R,X3I,X7R,X7I,W
```

5.6.5 PID Control

Control systems are concerned with regulating a process and achieving a desired behaviour 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 behaviour. 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

$$u(t) = K_p\, e(t) + K_i \int e\,dt + K_d\, de/dt$$

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

$$u(n) = u(n-1) + K_1\, e(n) + K_2\, e(n-1) + K_3\, e(n-2)$$

This algorithm is implemented in Example 5-40.

Example 5-40. PID Control

```
        TITL  'PID CONTROL'
        DEF   PID
*
* THIS ROUTINE IMPLEMENTS A PID ALGORITHM.
*
UN      EQU   0         ; OUTPUT OF CONTROLLER
E0      EQU   1         ; LATEST ERROR SAMPLE
E1      EQU   2         ; PREVIOUS ERROR SAMPLE
E2      EQU   3         ; OLDEST ERROR SAMPLE
K1      EQU   4         ; GAIN CONSTANT
K2      EQU   5         ; GAIN CONSTANT
K3      EQU   6         ; GAIN CONSTANT
*
* ASSUME DATA PAGE 0 IS SELECTED.
*
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 2.6 μs at a 20-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.

5.6.6 Selftest Routines

A selftest program can effectively perform incoming quality verification or be used as a powerup device verification tool. Texas Instruments has developed a selftest program to check out the functionality of a TMS320C1x device before branching to the user code. This program is not intended to provide a means of logic debug but rather to indicate device pass/fail from which it can be determined whether or not the TMS320C1x is still functional.

When designing a DSP device, Texas Instruments runs very thorough patterns through the logic to test all the stages. In these patterns, worst-case conditions and transitions are forced in order to verify logic design prior to manufacturing. Likewise, the speed and electrical specifications are thoroughly tested. In production manufacturing, every TMS320C1x is tested to meet the functionality, speed, and power specifications of the device before it is shipped. The drive levels and loading of lines are checked at full speed and over varying temperature.

The 460-word selftest program for the TMS320C1x exercises most of the on-chip resources of the device with a minimal amount of external circuitry.

Note that this code is intended for testing on-chip resources and will not exercise the external interface lines.

Example 5-41 contains a small portion of this selftest program, which checks out the ALU section. The ALU test is designed to validate the basic operation of the circuit. It consists of a series of subtests to verify addition and subtraction operations of both halves of the 32-bit operation as well as carry and overflow calculations, absolute value, and SUBC operation. A failure in any of these tests will set the error code in the accumulator to >100X where X is the number of the subtest that has failed.

Other sections of this selftest check the auxiliary registers, on-chip data RAM, on-chip program ROM (longitudinal redundancy test), status register and branches, pre- and post-scaling shifters, multiplier, and the instruction set.

An applications brief is available which discusses the code segments that comprise the TMS320C1x selftest program as well as how to link and execute this code. The applications brief and selftest code are available via the TMS320 DSP Bulletin Board Service (see Appendix E).

Example 5-41. Selftest Routine

```
* THIS PROGRAM EXECUTES AN INTERNAL SELFTEST OF THE TMS320C1X
* MICROCOMPUTER ALU. A FAILURE IN ANY OF THESE TESTS WILL SET
* THE ERROR CODE IN THE ACCUMULATOR TO >100X WHERE X IS THE
* NUMBER OF THE SELFTEST THAT HAS FAILED.
*
      RORG 0
*
* RESET AND INTERRUPT VECTORS.
*
BEGIN  B     START   ; RESET SOFT VECTOR
       B     INTRPT  ; INTERRUPT SOFT VECTOR
*
* REQUIRED DATA VALUES FOR TEST PROGRAMS.
*
       DATA >FFFF    ; RAM TEST PATTERN 1
       DATA >AAAA    ; RAM TEST PATTERN 2
       DATA >5555    ; RAM TEST PATTERN 3
       DATA >0       ; RAM TEST PATTERN 4
*
* PROGRAM INITIALIZATION DP = 0 AND DISABLE INTERRUPTS.
*
START  EQU  $        ; START INITIALIZATION ROUTINE
       LDPK 0        ; START IN ZERO DATA PAGE
       DINT          ; DISABLE EXTERNAL INTERRUPTS
*
* ARITHMETIC LOGIC UNIT TEST.
*
ALU    EQU  $
       LACK 1        ; GET INCREMENT VALUE
       SACL 8        ; STORE IT IN REG8
       LACK 4        ; POINT ACC TO PATTERNS TABLE
       TBLR 4        ; PUT TABLE VALUE IN REG4
       ADD  8        ; INCREMENT TABLE ADDRESS
       TBLR 5        ; PUT TABLE VALUE IN REG5
       ADD  8        ; INCREMENT TABLE ADDRESS
       TBLR 6        ; PUT TABLE VALUE IN REG6
       ADD  8        ; INCREMENT TABLE ADDRESS
       TBLR 7        ; PUT TABLE VALUE IN REG7
       LACK >10      ; SET ERROR CODE VALUE
       SACL 2        ; STORE CODE IN REG2
*
ALU1   ZAC           ; CLEAR OUT ACCUMULATOR
       ADDS 5        ; ADD IN >AAAA PATTERN
       AND  5        ; AND WITH >AAAA PATTERN
       OR   6        ; OR WITH >5555 PATTERN
       SUBS 4        ; SUBTRACT -1 FROM PATTERN
       BZ   ALU2     ; IF ACC CLEARED, GO TO NEXT TEST
*
       LACK 1        ; IF NOT, THEN SET TEST 1 CODE
       ADD  2,8      ; ADD IN ERROR CODE
       B    ERROR    ; EXIT TO ERROR ROUTINE
*
ALU2   ZALH 5        ; ADD HIGH THE >AAAA PATTERN
       ADDH 6        ; SUBTRACT HIGH THE >5555 PATTERN
       SACH 0        ; SAVE THE VALUE
       ZALH 0        ; RESTORE THE VALUE
       ABS           ; TAKE ABSOLUTE VALUE
       SUBH 8        ; SUBTRACT HIGH >10000
       BZ   ALU3     ; IF ACC CLEARED, GO TO NEXT TEST
```

```
*
        LACK  2         ; IF NOT, THEN SET TEST 2 CODE
        ADD   2,8       ; ADD IN ERROR CODE
        B     ERROR     ; EXIT TO ERROR ROUTINE
*
ALU3    LAC   4,12      ; LOAD ACC WITH >FFFFF000 PATTERN
        ADD   8,12      ; ADD >00001000 TO IT
        BZ    ALU4      ; IF ACC CLEARED, GO TO NEXT TEST
*
        LACK  3         ; IF NOT, THEN SET TEST 3 CODE
        ADD   2,8       ; ADD IN ERROR CODE
        B     ERROR     ; EXIT TO ERROR ROUTINE
*
ALU4    ADD   4         ; LOAD ACC WITH >FFFFFFFF PATTERN
        ABS             ; TAKE ABSOLUTE VALUE
        SUB   8         ; SUBTRACT >00000001
        BZ    ALU5      ; IF ACC CLEARED, GO TO NEXT TEST
*
        LACK  4         ; IF NOT, THEN SET TEST 4 CODE
        ADD   2,8       ; ADD IN ERROR CODE
        B     ERROR     ; EXIT TO ERROR ROUTINE
*
ALU5    LACK  >40       ; GET DIVISOR = 64
        SACL  0         ; SAVE IN REG0
        LACK  >FF       ; GET DIVIDEND = 255
        SUBC  0         ; 1ST STAGE OF DIVIDE
        NOP             ; REQUIRED NOP
        SUBC  0         ; 2ND STAGE OF DIVIDE
        NOP             ; REQUIRED NOP
         .
         .
         .
        SUBC  0         ; 16TH STAGE OF DIVIDE
        NOP             ; REQUIRED NOP
        SACH  1         ; SAVE REMAINDER
        SACL  2         ; SAVE QUOTIENT
        LACK  3         ; GET QUOTIENT COMPARISON MASK
        XOR   2         ; COMPARE WITH CALCULATED ANSWER
        BZ    ALU6      ; IF ACC CLEARED, GO TO NEXT TEST
*
        LACK  5         ; IF NOT, THEN SET TEST 5 CODE
        ADD   2,8       ; ADD IN ERROR CODE
        B     ERROR     ; EXIT TO ERROR ROUTINE
*
ALU6    LACK  >3F       ; GET REMAINDER COMPARISON MASK
        XOR   1         ; COMPARE WITH ANSWER
        BZ    STATUS    ; IF ACC CLEARED, GO TO NEXT TEST
*
        LACK  6         ; IF NOT, THEN SET TEST 6 CODE
        ADD   2,8       ; ADD IN ERROR CODE
        B     ERROR     ; EXIT TO ERROR ROUTINE
```

6. Hardware Applications

Information and examples on how to interface the TMS320C1x (first-generation TMS320) to external devices are presented in this section. The examples given are general enough in nature that they may be easily adapted to fit a particular system requirement.

The following buses, ports, and control signals provide system interface to the TMS320C1x processor:

- 12-bit address bus (A11-A0)
- 16-bit data bus (D15-D0)
- 3-bit port address bus
- Memory control signals (MC/$\overline{MP}$ or MC/$\overline{PM}$)
- Reset ($\overline{RS}$)
- Interrupt ($\overline{INT}$) and branch control ($\overline{BIO}$)
- Enable signals ($\overline{DEN}$, $\overline{MEN}$, and $\overline{WE}$)
- External flag (XF))
- Serial port clock (SCLK)
- Serial port receive/transmit channel inputs/outputs (DR/DX)
- Serial port framing inputs and output ($\overline{FSR}$, $\overline{FSX}$, and FR)
- Coprocessor port read/write signals ($\overline{RD}$/$\overline{WR}$)
- Coprocessor latch signals ($\overline{TBLF}$/$\overline{RBLE}$).

Major hardware applications discussed in this section are listed below.

6.1 Expansion Memory Interface

The TMS320C1x can be interfaced to a wide variety of memory and I/O devices. The TMS32010/C10 and TMS320C15/E15 devices can be interfaced to up to 4K words of external program memory. Expansion of program memory is accomplished directly through the use of the $\overline{MEN}$ (memory enable) and $\overline{WE}$ (write enable) control lines, with memory accesses occurring in a single cycle.

6.1.1 Program ROM Expansion

Twelve TMS32010 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 $\overline{MEN}$ (memory enable) strobe is generated by driving $\overline{MEN}$ low to enable external memory. The instruction word is then transferred to the TMS32010 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 form data bus valid prior to CLKOUT↓

serves as the timing constraint used when calculating $t_{c(C)}$.

When only external program ROM is required, a minimum system can consist of a TMS320C10/C15 and up to 4K words of external program memory (TMS27C292), as shown in Figure 6-1. The $\overline{MEN}$ signal and the address (A11-A0) and data (D15-D0) lines on the TMS320C10/C15 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 TMS320C10/C15.

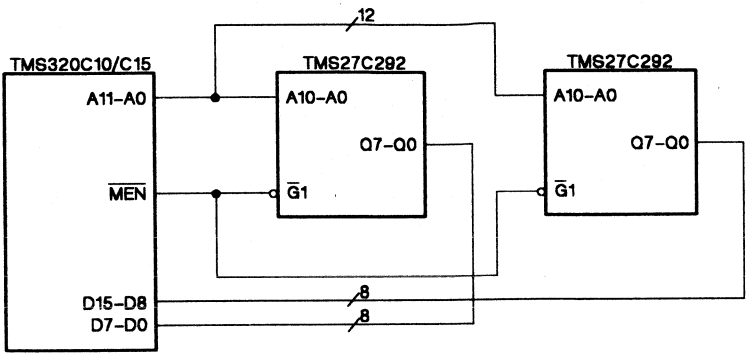


Figure 6-1. Minimum Program ROM Expansion

A very low chip-count system can result when using the low-cost TMS32010-16. The use of EPROMs in an external program memory interface to the TMS32010-16 allows the implementation of 4K words of non-volatile program memory along with the added flexibility of reprogrammability, thus providing for system development, future program expansion, and/or upgrade modification. Single-cycle memory access using a direct memory interface requires no additional external interface logic.

On the TMS32010-16, t_{d1} with a maximum value of 50 ns and $t_{su(D)}$ with a minimum value of 50 ns are both constants; therefore, $t_{a(A)}$ is the only remaining variable used in determining the minimum clock cycle time of the system. For the circuit shown in Figure 6-2 (with $t_{a(A)}$ = 170 ns), inserting these values into the equation yields $t_{c(C)}$ min = 250 ns.

The memories used in Figure 6-2 are a pair of Texas Instruments TMS2732A-17 4K x 8 EPROMs, configured in parallel for a direct 16-bit interface to the TMS32010-16. These EPROMs display a 170-ns access time. However, other EPROMs may be used with access times best suited to a particular application as long as the TMS32010-16 clock frequency has been selected to allow for the access time of the EPROMs chosen.

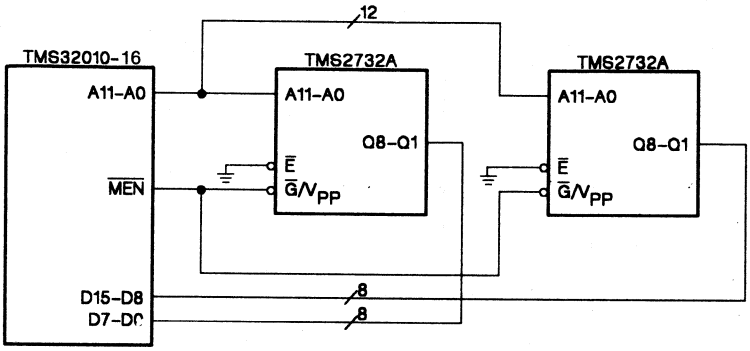


Figure 6-2. EPROM Interface to the TMS32010-16

Contention for the data bus is not a concern in this memory configuration. Therefore, the $\overline{E}$ (chip enable) pin for the EPROM pair has been tied to ground to avoid unnecessary switching transients that could be induced if the chip enables were toggled upon memory access.

6.1.2 Data RAM Expansion

No direct memory expansion is provided on the TMS320C1x. 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 TMS320C1x 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 6-3 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.

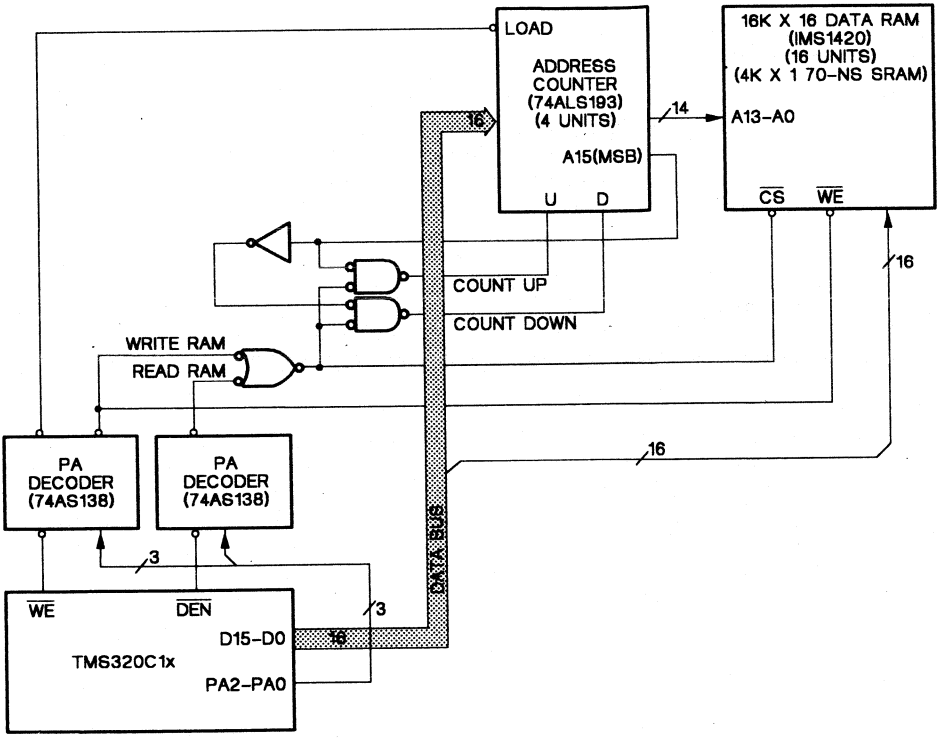


Figure 6-3. Data RAM Expansion

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 TMS320C1x does not implement "wait" states, memory refresh must be generated transparent to the processor.

For additional information regarding interfacing to TMS320C1x devices, refer to the book, *Digital Signal Processing Applications with the TMS320 Family*.

6.2 Codec Interface

In areas of telecommunications, speech processing, and other applications that require low-cost analog I/O devices, a combo-codec may be useful. A combo-codec consists of nonlinear A/D and D/A converters with antialiasing and smoothing filters and data storage registers. For additional information on combo-codecs, refer to the *TCM29C13/C14/C16/C17 Combined Single-Chip PCM Codec and Filter Data Sheet.*

The TMS 320C17/E17 is capable of direct interface to serial devices such as combo-codecs, thus reducing chip count and improving system throughput. These TMS320 devices can also compand (COMpress and exPAND) a PCM (Pulse Code Modulation) data stream, acquired by the codec, through the use of on-chip companding hardware.

Figure 6-4 shows the TMS 320C17/E17 interfaced to a TCM29C13 combo-codec to demonstrate direct serial-port interface capability. A standalone full-duplex serial interface is shown, in which the TMS 320C17/E17 provides the serial clock for bit transmission. The codec is sampled every 125 μs (8-kHz frequency), at which time an 8-bit PCM byte is exchanged between the two devices. A second port can also be interfaced to the TMS 320C17/E17 with no additional logic or interconnections since these devices implement two independent serial ports.

Timing for the serial interface system is controlled by the serial-port clock (SCLK). SCLK is configured as an output from the TMS 320C17/E17 , and its frequency is set to 2.048 MHz (see Section 3.9). A 20.48-MHz crystal is input to the TMS320 as its system clock. The SCLK frequency is derived from this system clock by a divide-by-10 in the SCLK prescale control logic, initialized through control register 1. SCLK is connected to CLKR/CLKX on the TCM29C13 to provide the transmit and receive master clock. CLKSEL on the codec is tied to V_{CC} to select the 2.048-MHz master clock mode.

Framing pulses are generated by the TMS 320C17/E17 on the FR output pin. The frequency of these pulses is set to 8 kHz by dividing the serial clock (SCLK) by 256. This value is also initialized through control register 1. The short FR framing pulses provide the codec with framing pulses for the fixed data-rate mode. FR is input to both the FSX and FSR inputs on the codec. The FR output causes simultaneous transmit and receive operations from the serial port. The FSX input on the codec causes the device to transmit PCM data on the next eight consecutive positive transitions of the serial-port clock (SCLK). The FSR input on the codec causes the device to receive PCM data on the next eight consecutive negative transitions of the serial-port clock (SCLK). With this timing, the codec transmits and receives one 8-bit PCM sample every 125 μs.

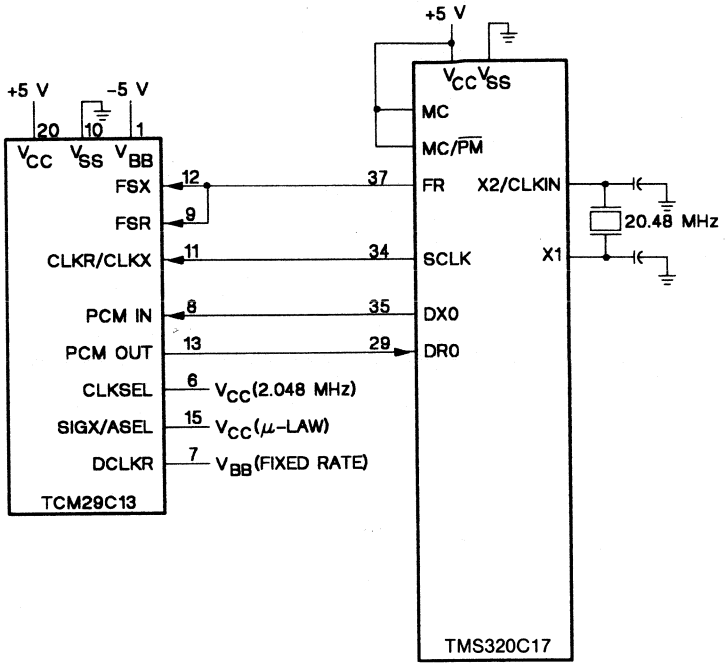


Figure 6-4. Codec Interface for Standalone Serial Operation

The TMS 320C17/E17 transmits its PCM sample via the DX0 pin. The sample is received by the TCM29C13 on the PCM IN pin. The TMS320 receives PCM samples on its DR0 pin, which is the output of the PCM OUT pin of the TCM29C13. With this setup, single-channel operation is realized with the TMS 320C17/E17 . All data transmission occurs on channel 0, requiring one IN instruction from port 1 to receive the PCM sample and one OUT instruction to port 1 to send a sample to the codec.

In the serial interface configuration, μ-255 law companding is selected by setting system control register bit 14 (CR14) to logic 0. The TCM29C13 is put into the μ-law companding mode by connecting the SIGX/ASEL pin to V_{CC}.

Linear A/D and D/A converters may also be interfaced to the TMS 320C17/E17 through its parallel ports instead of using the serial port.

6.3 A/D and D/A Interface

The TMS320C10/C15 can be interfaced to A/D (analog-to-digital) and D/A (digital-to-analog) converters to perform the necessary conversions. A minimum of external circuitry is required.

Figure 6-5 shows an interface of the TLC0820 8-bit A/D converter to the TMS320C10/C15. Since the control circuitry of the TLC0820 operates much more slowly that the TMS320C10/C15, 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
74LS32	Quad two-input OR-gate.

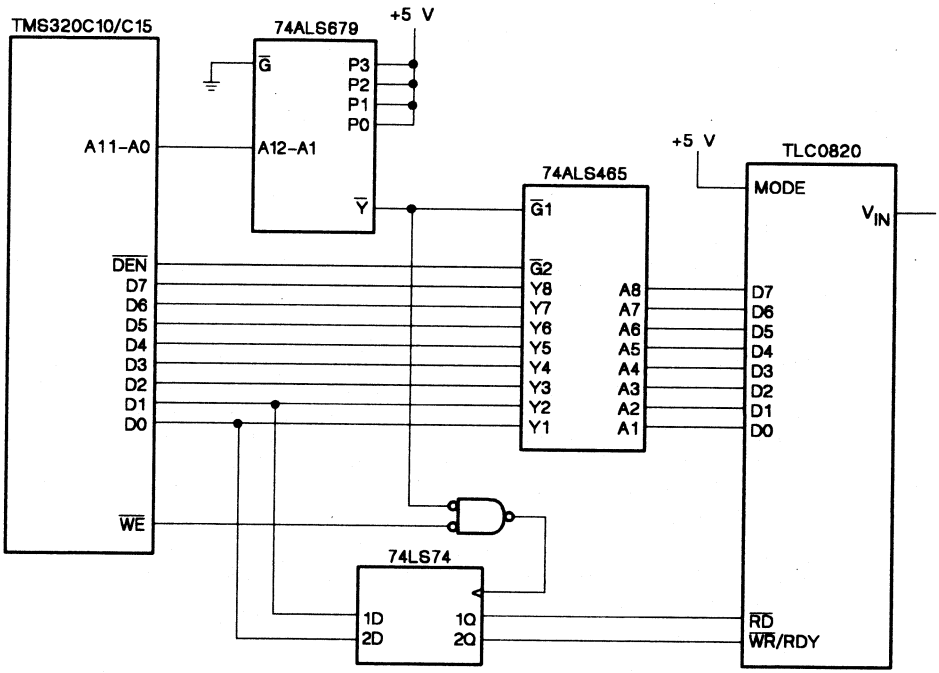
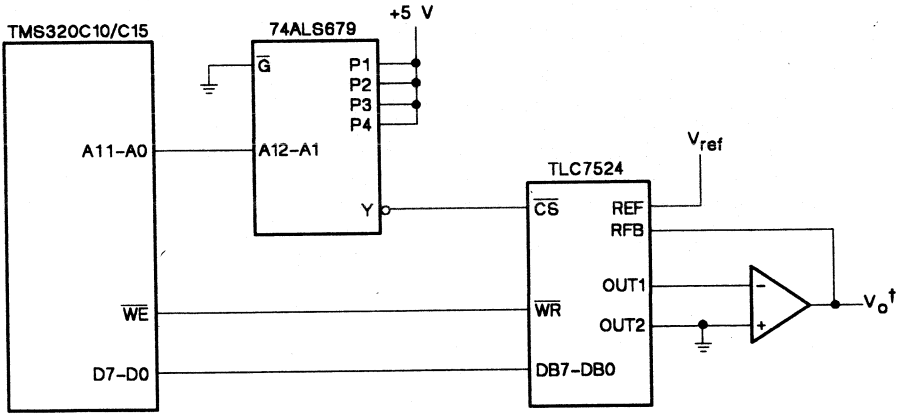


Figure 6-5. A/D Converter to TMS320C10/C15 Interface

An interface of the TLC7524 8-bit D/A converter to the TMS320C10/C15 is shown in Figure 6-6. Due to the high-speed operation of the internal logic circuitry of the TLC7524, the interface to the TMS320C10/C15 requires external logic circuitry to decode the address of the peripheral. Here a 74ALS679 12-bit address comparator is used.



†$V_o = -V_{ref} \frac{D}{256}$, where D = digital input

Figure 6-6. D/A Converter to TMS320C10/C15 Interface

For further information about the A/D and D/A converters shown in the figures, refer to the *Linear Data Book*.

6.4 I/O Ports

The TMS320C1x devices interface to input/output (I/O) devices through the eight 16-bit parallel ports (see Section 3.7 for I/O functions). The I/O space is selected by the $\overline{DEN}$ signal for reads and the $\overline{WE}$ signal for writes. Each of the eight I/O ports is addressed by the three LSBs of the address bus with all other address lines held low. The I/O ports share the 16 data lines.

The I/O ports may be used for interfacing external circuitry such as data memory expansion devices (see Section 6.1), A/D and D/A converters, synchronization latches, or memory-mapped peripheral devices. Figure 6-7 shows a circuit that can be used to generate device select lines for each of the individual port writes. A similar circuit may be used to enable I/O port reads.

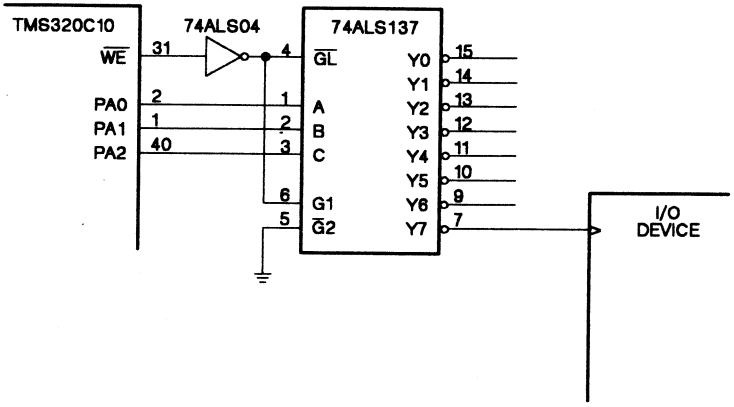


Figure 6-7. I/O Port Interface Circuit

When interfacing the TMS320C1x to slower devices, a handshake interface used in conjunction with the I/O port interface may be desirable. Data to be transferred may be stored in latches to be read by the TMS320C1x at a later time. Handshaking may then be established using the interrupt, $\overline{BIO}$, and XF (TMS 320C17/E17) signals.

6.5 Coprocessor Interface

The TMS320C17/E17 includes an option to use the parallel I/O interface exclusively as a coprocessor interface. This option includes both the buffer logic to communicate between two processors asynchronously, and the protocol logic to protect against miscommunication. This port allows the TMS320C17/E17 to act as either a master processor or a slave processor in a multiprocessing system. The circuit also allows data to be transferred as either 8 or 16-bit values.

As a master processor, the TMS320C17/E17 writes to and reads from the coprocessor interface at will. This requires that the slave processor keep the receive buffer full and the transmit buffer empty. Figure 6-8 shows the TMS320C17 as a master processor to a TMS70C42 (8-bit microcomputer). As the internal CPU writes to the coprocessor interface, the $\overline{\text{TBLF}}$ (transmit buffer latch full) signal is driven active low. This signals the TMS70C42 that there is data to be read and that the 8-bit microcomputer must read that data before the next write by the internal CPU. In Figure 6-8, the $\overline{\text{TBLF}}$ signal is tied to an I/O bit on the 8-bit microcomputer so that the microcomputer can poll the signal and act accordingly. This signal could also be tied to an interrupt on the 8-bit microcomputer if this better suited system requirements. When the internal CPU reads its buffer, it signals the 8-bit microcomputer that the read buffer is empty by generating the $\overline{\text{RBLE}}$ (read buffer latch empty) signal. This signals the microcomputer that it must reload the receive latch before the next internal CPU access.

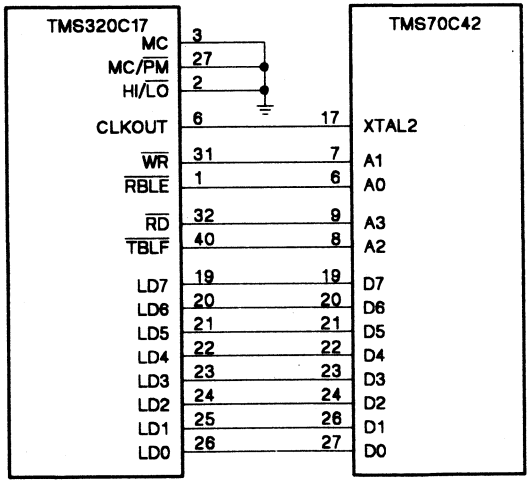


Figure 6-8. TMS320C17 to TMS70C42 Interface

When the TMS320C17/E17 serves as a slave processor, data transfers are controlled by the master processor. Figure 6-9 shows the TMS320C17 as a slave to a TMS320C25 (a 16-bit microprocessor). When the TMS320C25 writes to the TMS320C17, it causes an interrupt to the internal CPU. The CPU must then read the information stored in the coprocessor interface before the next write from the TMS320C25. When the TMS320C25 reads the transfer latch of the coprocessor port, the internal CPU receives an active low $\overline{BIO}$ signal. When transferring information to the master processor, the internal CPU monitors the $\overline{BIO}$ line (using the BIOZ instruction) to determine when it can reload the transmit latch. Note that a wait state may be required when interfacing to the TMS320C25.

To support mixed 8/16-bit operation, the read buffer latch is cleared to 0 when read by the internal CPU.

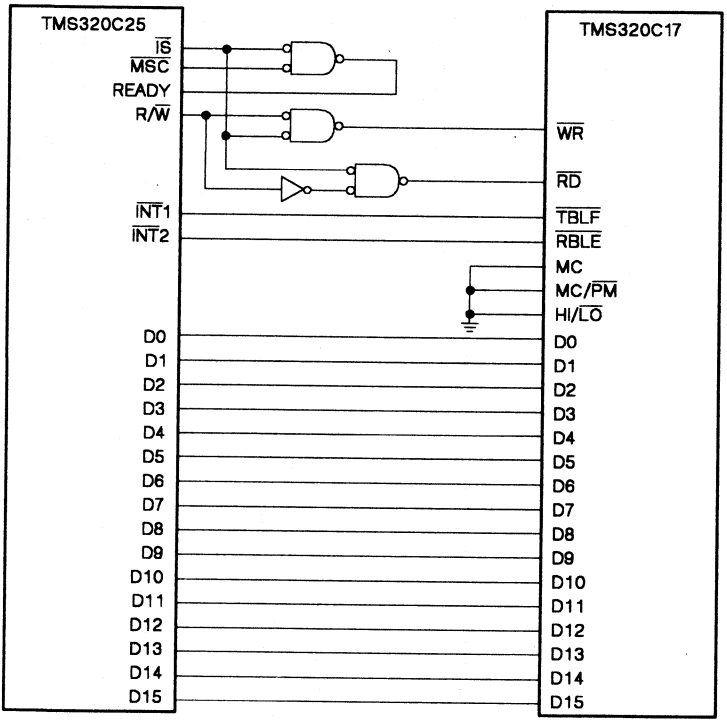


Figure 6-9. TMS320C17 to TMS320C25 Interface

6.6 System Applications

The TMS320C1x devices are commonly used in many system applications. Several of these system applications are presented in this section, in a general form, to illustrate basic approaches to system design using the TMS320C1x. These applications include a 2400 bps modem, a speech synthesis system, and a voice store-and-forward message center.

6.6.1 2400 bps Modem

The implementation of a 2400 bps modem is shown in Figure 6-10. This system implements the functions of a V.22 bis modem using a TMS320A2400 and a TMS70A2400, which are masked ROM versions of the TMS320C17 and TMS7042, respectively. The TMS320A2400 performs all of the signal processing functions, and the TMS70A2400 performs all of the interface protocol and control functions. The remaining system components perform analog-to-digital (A/D) and digital-to-analog (D/A) conversions, PC bus interface, telephone line interface, and filtering functions.

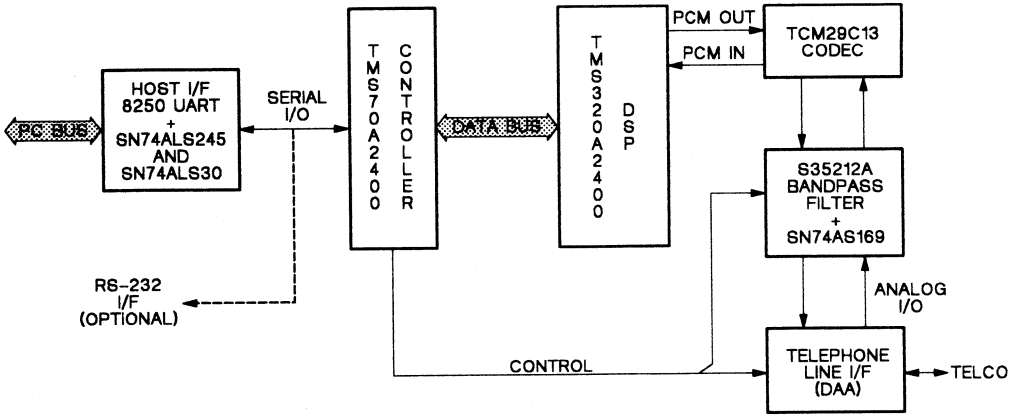


Figure 6-10. 2400 bps Modem

6.6.2 Speech Synthesis System

The system design for speech applications consists of a codec, a digital signal processor supported with program and data memory, a speech data memory, and an optional host processor. A block diagram of this system, shown in Figure 6-11, consists of the following components:

- Codec (TCM29C18)
- Digital signal processor (TMS320C17)
- Speech data ROM (TMS60C20) or EPROM (TMS27C56)
- Microcomputer host (TMS70C42).

The actual speech system is composed of the digital signal processor and the codec. The microcomputer host is used to perform an end-product application that calls upon the speech subsystem when needed, such as in the case of a minicomputer and array processor system. The speech system can be used to perform speech synthesis, vocoding, speech recognition, speaker verification, DTMF decoding/encoding as well as many other algorithmically intensive applications.

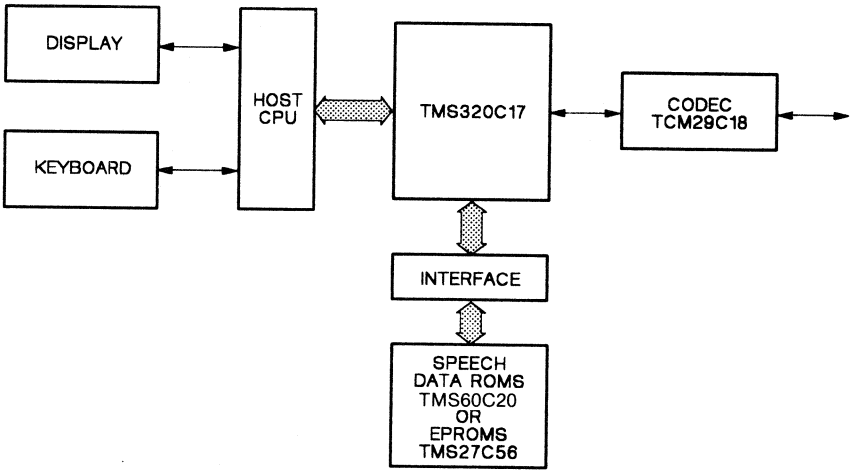


Figure 6-11. Speech Synthesis System

6.6.3 Voice Store-and-Forward Message Center

The voice store-and-forward message center consists of a TMS320C17-based system interfaced to a phone line and a large storage area either on DRAMs or computer disks depending on the application. Some applications of the message center are: voice mail for a computer network, answering machines for home use (see Figure 6-12), and a hand-held battery-operated voice message pad for personal use. Typical algorithms required to perform the task are: half-duplex ADPCM or subband coder, LPC synthesis, and DTMF encoder/decoder. A combination of these algorithms will fit into the 4K on-chip program ROM of the TMS320C17, requiring no external data memory. Because the CPU utilization is less than 100 percent when performing any of these tasks, other operations can also be done by the TMS320C17, such as digital volume control, noise filtering, etc. A masked ROM version of the TMS320C17 can provide a cost-effective solution.

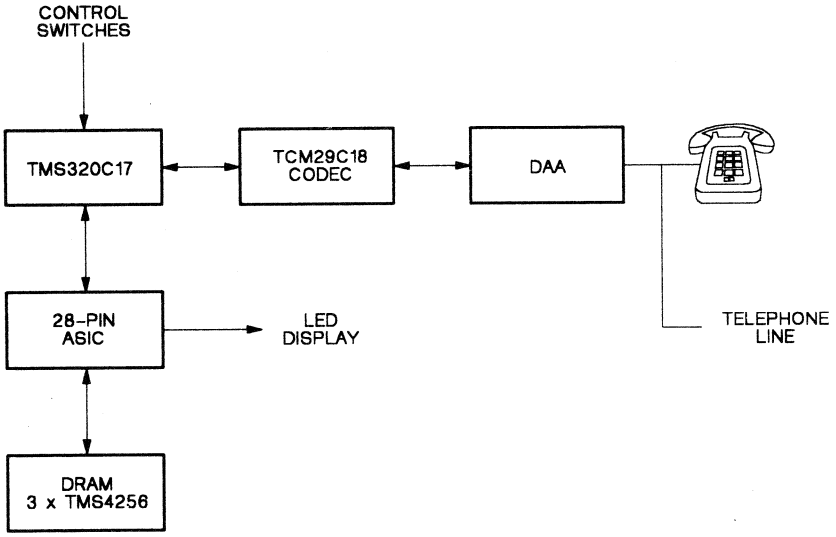
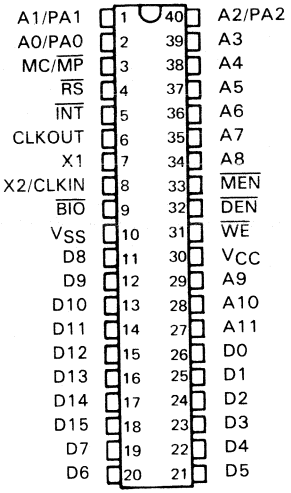


Figure 6-12. Answering Machine

JANUARY 1987

- 160-ns Instruction Cycle
- 144/256-Word On-Chip Data RAM
- 1.5K/4K-Word On-Chip Program ROM
- 4K-Word On-chip Program EPROM (TMS320E15/E17)
- EPROM Code Protection for Copyright Security
- 4K-Word Total External Memory at Full Speed
- 32-Bit ALU/Accumulator
- 16 x 16-Bit Multiplier with a 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- Eight Input and Eight Output Channels
- Dual-Channel Serial Port (TMS320C17/E17)
- 16-Bit Bidirectional Data Bus with 50-Mbps Transfer Rate
- Single 5-V Supply
- Packaging: 40-Pin DIP and 44-Pin PLCC
- Commercial and Military Versions Available
- NMOS Technology:
  - TMS32010-25 . . . . . 160-ns cycle time
  - TMS32010 . . . . . 200-ns cycle time
  - TMS32010-16 . . . . . 250-ns cycle time

TMS32010, TMS320C10
N PACKAGE
(TOP VIEW)



- CMOS Technology:
  - TMS320C10-25 . . . . . 160-ns cycle time
  - TMS320C10 . . . . . 200-ns cycle time
  - TMS320C15-25 . . . . . 160-ns cycle time
  - TMS320C15 . . . . . 200-ns cycle time
  - TMS320E15 (EPROM) . 200-ns cycle time
  - TMS320C17-25 . . . . . 160-ns cycle time
  - TMS320C17 . . . . . 200-ns cycle time
  - TMS320E17 (EPROM) . 200-ns cycle time

This data sheet provides complete design documentation for all the first-generation devices of the TMS320 family. This facilitates the selection of the devices best suited for user applications by providing all specifications and special features for each TMS320 member. This data sheet is divided into four major sections: architecture, electrical specifications (NMOS and CMOS), timing diagrams, and mechanical data. In each of these sections, generic information is presented first, followed by specific device information. An index is provided for quick reference to specific information about a device.

escription

The TMS320 family of 16/32-bit single-chip digital signal processors combines the flexibility of a high-speed controller with the numerical capability of an array processor, thereby offering an inexpensive alternative to multichip bit-slice processors. The highly paralleled architecture and efficient instruction set provide speed and flexibility to produce a MOS microprocessor family capable of executing 6.4 MIPS (million instructions per second). The TMS320 family optimizes speed by implementing functions in hardware that other processors implement through microcode or software. This hardware-intensive approach provides the design engineer with processing power previously unavailable on a single chip.

description (continued)

The TMS320 family consists of two generations of digital signal processors. The first generation contair the TMS32010 and its spinoffs, as described in this data sheet. The TMS32020 and TMS320C25 ar the second-generation processors, designed for higher performance. Many features are common amon the TMS320 processors. Specific features are added in each processor to provide different cost/performanc tradeoffs. Software compatibility is maintained throughout the family to protect the user's investmer in architecture. Each processor has software and hardware tools to facilitate rapid design.

introduction

The TMS32010, the first NMOS digital signal processor in the TMS320 family, was introduced in 198 Its powerful instruction set, inherent flexibility, high-speed number-crunching capabilities, and innovativ architecture have made this high-performance, cost-effective processor the ideal solution to mar telecommunications, computer, commercial, industrial, and military applications. Since that time, th TMS320C10, a low-power CMOS version of the industry-standard TMS32010, and other spinoff device have been added to the first generation of the TMS320 family.

The TMS32010 microprocessor is available in three speed versions: TMS32010 (20 MHz), TMS32010-2 (25 MHz), and TMS32010-16 (16 MHz). These devices are capable of executing a 16 x 16-bit multip with a 32-bit result in a single instruction cycle. On-chip data RAM of 144 words and on-chip progra ROM of 1.5K words are available. Full-speed execution of 4K words of off-chip program memory is als possible. The TMS32010-25, a 160-ns instruction cycle time version of the TMS32010, is intended f higher-performance applications that use off-chip program memory and require faster processor throughpu (6.25 MIPS). The TMS32010-16 provides a low-cost alternative for DSP applications not requiring th maximum operating frequency of the TMS32010. The device provides a direct EPROM interface for cos effective system development and modification. All of these devices are pin-for-pin and object-coc compatible with the TMS32010 and its development tools.

The TMS320C10 is object-code and pin-for-pin compatible with the TMS32010. It is processed in CMO technology, achieving a power dissipation less than one-sixth that of the NMOS device. The lower powe dissipation makes the TMS320C10 ideal for power-sensitive applications such as digital telephony an portable products. The TMS320C10-25, a 25-MHz version of the TMS320C10, has a 160-ns instructio cycle time and is well suited for high-performance DSP applications.

The TMS320C15 and TMS320E15 CMOS devices are object-code and pin-for-pin compatible with th TMS32010 and offer expanded on-chip RAM of 256 words and on-chip program ROM or EPROM of 4 words. These devices allow the capability of upgrading performance and reducing power, board space and system cost without hardware redesign. The TMS320C15 is also available in a 160-ns version, th TMS320C15-25.

troduction (continued)

The TMS320C17 and TMS320E17 dedicated microcomputers also offer expanded on-chip RAM of 256 words and on-chip program ROM or EPROM of 4K words. These devices provide a dual-channel serial interface, on-chip μ-law/A-law companding hardware, and a serial port timer. In addition, a 16-bit coprocessor interface provides a direct communication channel to common 4/8-bit microcomputers (no glue logic required), and minimal logic interface to most common 16/32-bit microprocessors. The devices are object-code compatible with the TMS32010 and processed in CMOS technology. The TMS320C17 is also available in a 160-ns version, the TMS320C17-25.

Table 1 provides an overview of the first generation of TMS320 processors with comparisons of memory, I/O, cycle timing, power, package type, technology, and military support. For specific availability, contact the nearest TI sales office.

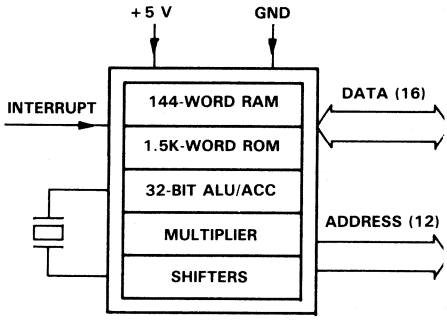
TABLE 1. TMS320 FIRST-GENERATION DEVICE OVERVIEW

DEVICE		MEMORY				I/O†			CYCLE TIME	TYP POWER	PACKAGE TYPE	
		ON-CHIP			OFF-CHIP							
		RAM	ROM	EPROM	EXPANSION	SER	PAR	CPX	(ns)	(mW)	DIP	PLCC
TMS32010-25	(NMOS)	144	1.5K	—	4K	—	8 x 16	—	160	900	40	—
TMS32010‡	(NMOS)	144	1.5K	—	4K	—	8 x 16	—	200	900	40	—
TMS32010-16	(NMOS)	144	1.5K	—	4K	—	8 x 16	—	250	900	40	—
TMS320C10-25	(CMOS)	144	1.5K	—	4K	—	8 x 16	—	160	200	40	44
TMS320C10§	(CMOS)	144	1.5K	—	4K	—	8 x 16	—	200	165	40	44
TMS320C15-25	(CMOS)	256	4K	—	4K	—	8 x 16	—	160	250	40	44
TMS320C15§	(CMOS)	256	4K	—	4K	—	8 x 16	—	200	225	40	44
TMS320E15§	(CMOS)	256	—	4K	4K	—	8 x 16	—	200	300	40	—
TMS320C17-25	(CMOS)	256	4K	—	—	2	6 x 16	YES	160	275	40	44
TMS320C17	(CMOS)	256	4K	—	—	2	6 x 16	YES	200	250	40	44
TMS320E17	(CMOS)	256	—	4K	—	2	6 x 16	YES	200	325	40	—

SER = serial; PAR = parallel; CPX = coprocessor interface.
Military version available.
Military version planned; contact nearest TI sales office for availability.

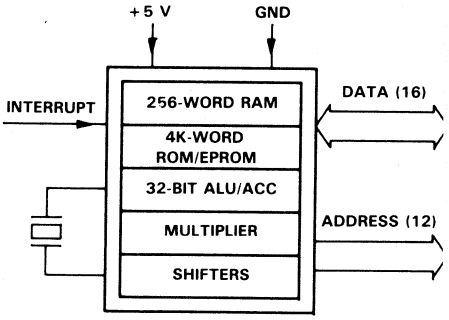
Key Features: TMS32010/C10

- Instruction Cycle Timing:
  - 160 ns (TMS32010-25/C10-25)
  - 200 ns (TMS32010/C10)
  - 250 ns (TMS32010-16)
- 144 Words of On-Chip Data RAM
- 1.5K Words of On-Chip Program ROM
- External Memory Expansion up to 4K Words at Full Speed
- 16 x 16-Bit Multiplier with 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- On-Chip Clock Oscillator
- Single 5-V Supply
- Device Packaging:
  - 40-Pin DIP (all devices)
  - 44-Lead PLCC (CMOS only)
- Technology
  - NMOS: TMS32010/10-16/10-25
  - CMOS: TMS320C10/C10-25



Key Features: TMS320C15/E15

- Instruction Cycle Timing:
  - 160 ns (TMS320C15-25)
  - 200 ns (TMS320C15/E15)
- 256 Words of On-Chip Data RAM
- 4K Words of On-Chip Program ROM (TMS320C15/C15-25)
- 4K Words of On-Chip Program EPROM (TMS320E15)
- EPROM Code Protection for Copyright Security
- External Memory up to 4K Words at Full Speed
- Object-Code and Pin-For-Pin Compatible with TMS32010
- 16 x 16-Bit Multiplier with 32-Bit Product
- 0 to 16-Bit Barrel Shifter
- On-Chip Clock Oscillator
- Single 5-V Supply
- Device Packaging:
  - 40-Pin DIP (all devices)
  - 44-Lead PLCC† (TMS320C15/C15-25)
- CMOS Technology

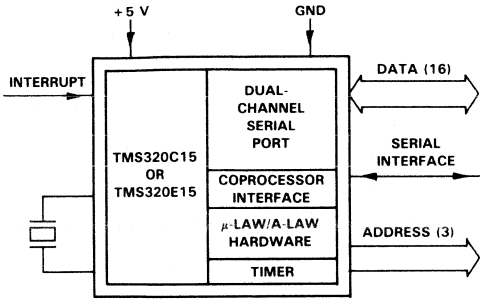


†PLCC version planned, contact nearest TI sales office for availability.

Key Features: TMS320C17/E17

- Instruction Cycle Timing:
  - 160 ns (TMS320C17-25)
  - 200 ns (TMS320C17/E17)
- 256 Words of On-Chip Data RAM
- 4K Words of On-Chip Program ROM (TMS320C17/C17-25)
- 4K Words of On-Chip Program EPROM (TMS320E17)
- EPROM Code Protection for Copyright Security
- Object-Code Compatible with TMS32010
- Dual-Channel Serial Port for Full-Duplex Serial Communication
- Serial Port Timer for Standalone Serial Communications
- On-Chip Companding Hardware for μ-law/A-law PCM Conversions
- 16-Bit Coprocessor Interface for Common 4/8/16/32-Bit Microcomputers/Microprocessors
- Device Packaging:
  - 40-Pin DIP (all devices)
  - 44-Lead PLCC† (TMS320C17/C17-25)
- CMOS Technology

†PLCC version planned; contact nearest TI sales office for availability.



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 TMS320 first-generation 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 TMS320 devices use a Harvard architecture, data and program memory reside in two separate spaces. The first-generation devices have 144 or 256 words of on-chip data RAM and 1.5K or 4K words of on-chip program ROM. On-chip program EPROM of 4K words is provided on the TMS320E15/E17. The EPROM cell utilizes standard PROM programmers and is programmed 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. The TMS320C17/E17 provides no memory expansion capability.

microcomputer/microprocessor operating modes (TMS32010/C10/C15)

The TMS32010/C10 and TMS320C15 devices offer two modes of operation defined by the state of the MC/$\overline{MP}$ pin: the microcomputer mode (MC/$\overline{MP}$ = 1) or the microprocessor mode (MC/$\overline{MP}$ = 0). In the microcomputer mode, on-chip ROM is mapped into the memory space with up to 4K words of memory available. In the microprocessor mode, all 4K words of memory are external.

interrupts and subroutines

The TMS320 first-generation 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. The interrupts used in these devices are maskable.

input/output

The 16-bit parallel data bus can be utilized to perform I/O functions in two cycles. The I/O ports are addressed by the three LSBs on the address lines. In addition, a polling input for bit test and jump operations ($\overline{BIO}$) and an interrupt pin ($\overline{INT}$) have been incorporated for multitasking.

serial port (TMS320C17/E17)

Two of the I/O ports on the TMS320C17/E17 are dedicated to the serial port and companding hardware. I/O port 0 is dedicated to control register 0, which controls the serial port, interrupts, and companding hardware. I/O port 1 accesses control register 1, as well as both serial port channels, and the companding hardware. The six remaining I/O ports are available for external parallel interfaces.

The dual-channel serial port is capable of full-duplex serial communication and offers direct interface to combo-codecs. Receive and transmit registers that operate with 8-bit data samples are I/O-mapped. Either internal or external framing signals for serial data transfers are selected through the system control register. The serial port clock provides the bit timing for transfers with the serial port, and may be either an input or output. A framing pulse signal provides framing pulses for combo-codec circuits, an 8-kHz sample clock for voice-band systems, or a timer for control applications.

companding hardware (TMS320C17/E17)

On-chip hardware enables the TMS320C17/E17 to compand (COMpress/exPAND) data in either μ-law or A-law format. The companding logic operation is configured via the system control register.
Data may be companded in either a serial mode for operation on serial port data (converting between sign-magnitude linear and logarithmic PCM) or a parallel mode for computation inside the device. The TMS320C17/E17 allows the hardware companding logic to operate with either sign-magnitude or two's-complement numbers.

coprocessor port (TMS320C17/E17)

The coprocessor port on the TMS320C17/E17 provides a direct connection to most 4/8-bit microcomputers and 16/32-bit microprocessors. The port is accessed through I/O port 5 using IN and OUT instructions. The coprocessor interface allows the device to act as a peripheral (slave) microcomputer to a microprocessor, or as a master to a peripheral microcomputer. The 16 data lines are used for the 6 parallel 16-bit I/O ports. In the coprocessor mode, the 16-bit parallel port is reconfigured to operate as a 16-bit latched bus interface. For peripheral transfer, an 8-bit or 16-bit length of the coprocessor port can be selected.

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.

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 the first page contains 128 words, and the second page contains up to 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 2 lists the symbols and abbreviations used in Table 3, the instruction set summary. Table 3 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 2. 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 3. 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 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 3. 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 →
BIOZ	Branch on $\overline{BIO}$ = 0	2	2	1 1 1 1 0 1 1 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 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 3. 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

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 4 lists the development support products for the first-generation TMS320 devices.

System development begins with the use of the Evaluation Module (EVM) or Emulator (XDS). These hardware tools allow 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 simulator, XDS, or EVM. 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* for further information about TMS320 development support products and DSP workshops.

When technical questions arise regarding the TMS320, contact the Texas Instruments Regional Technology Centers (see last pages).

TABLE 4. TMS320 FIRST-GENERATION SOFTWARE AND HARDWARE SUPPORT

SOFTWARE TOOLS	PART NUMBER
Macro Assembler/Linker	
VAX VMS	TMDS3240210-08
TI/IBM MS/PC-DOS	TMDS3240810-02
Simulator	
VAX VMS	TMDS3240211-08
TI/IBM MS/PC-DOS	TMDS3240811-02
Digital Filter Design Package (DFDP)	
TI PC MS-DOS	DFDP-TI001
IBM PC PC-DOS	DFDP-IBM001
DSP Software Library	
VAX VMS	TMDC3240212-18
TI/IBM MS/PC-DOS	TMDC3240812-12
HARDWARE TOOLS	**PART NUMBER**
Evaluation Module (EVM)	RTC/EVM320A-03
Analog Interface Board (AIB)	RTC/EVM320C-06
XDS/22 Emulator	TMDS3262211
XDS/22 Upgrade	
Factory Upgrade	TMDS3282215
Customer Upgrade	TMDS3282216
EPROM Programmer Adaptor Socket	RTC/PGM320A-06
TMS320 Design Kit	TMS320DDK

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.*

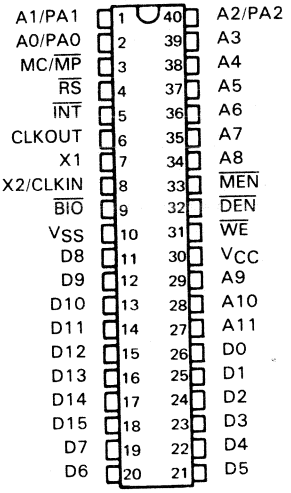
A series of DSP textbooks is being published by Prentice-Hall and John Wiley & Sons to support digital signal processing research and education. The TMS320 newsletter, *Details on Signal Processing,* is published quarterly and distributed to update TMS320 customers on product information. The TMS320 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 Regional Technology Centers (see last pages).

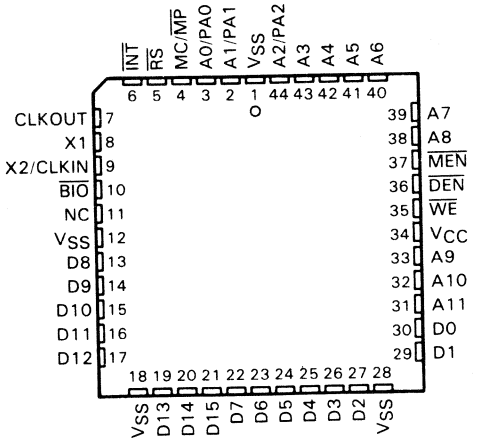
description

Since the TMS32010 was the first digital signal processor in the TMS320 family, its architecture has served as the basis from which first-generation spinoff devices have evolved. The TMS320C10 is a low-power CMOS version of the TMS32010 and identical to it. The TMS320C15/E15 is object-code and pin-for-pin compatible with the TMS32010 and offers expanded on-chip RAM and ROM or EPROM.

TMS32010, TMS320C10
TMS320C15, TMS320E15
N/JD PACKAGE
(TOP VIEW)



TMS320C10, TMS320C15
FN PACKAGE
(TOP VIEW)

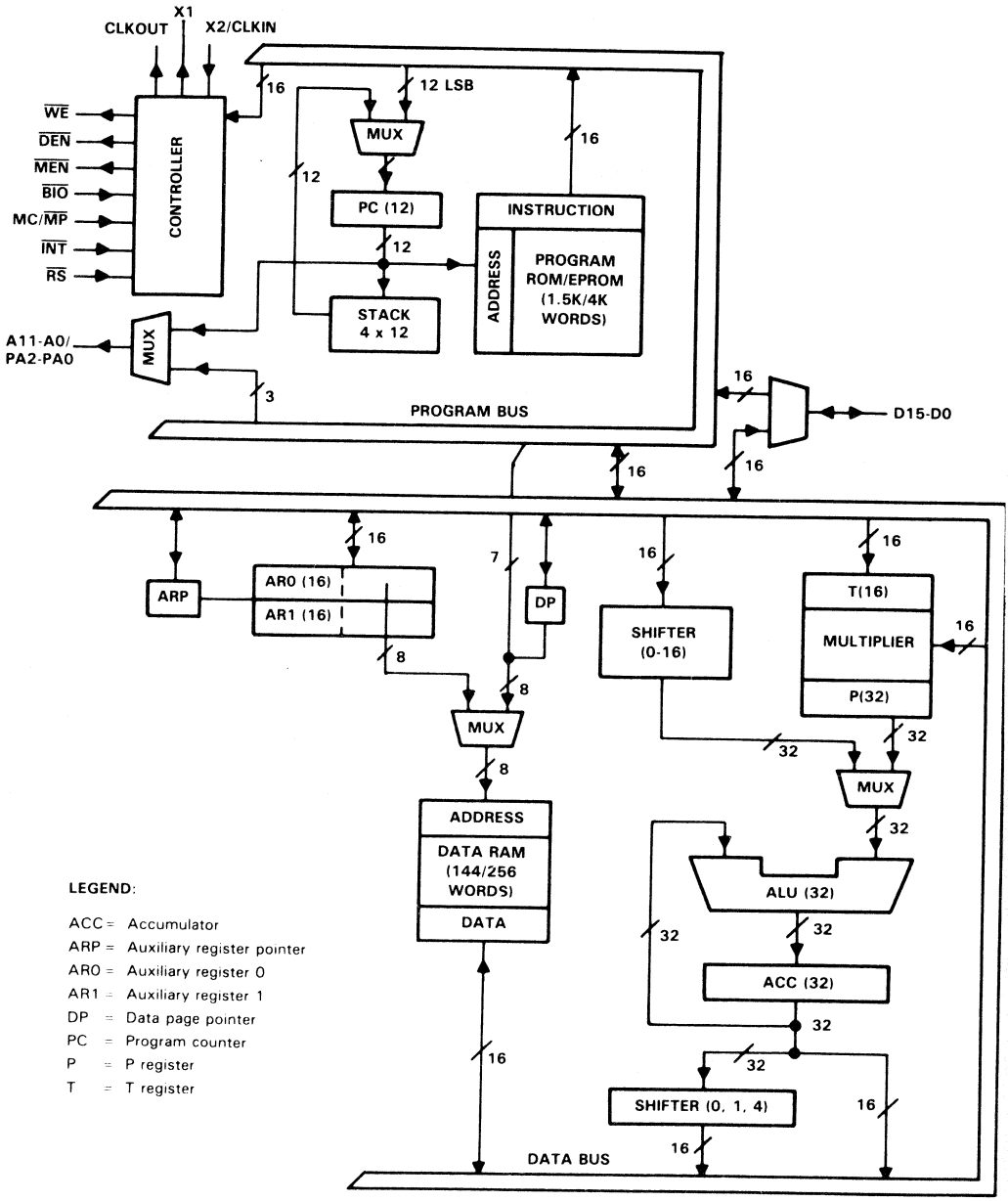


PIN NOMENCLATURE (TMS32010, TMS320C10, TMS320C15, TMS320E15†)

NAME	I/O/Z‡	DEFINITION
A11-A0/PA2-PA0	O	External address bus. I/O port address multiplexed over PA2-PA0.
BIO	I	External polling input
CLKOUT	O	System clock output, ¼ crystal/CLKIN frequency
D15-D0	I/O/Z	16-bit parallel data bus
DEN	O	Data enable for device input data on D15-D0
INT	I	External interrupt input
MC/MP	I	Memory mode select pin. High selects microcomputer mode. Low selects microprocessor mode.
MEN	O	Memory enable indicates that D15-D0 will accept external memory instruction.
NC	-	No connection; make no external connection to this pin.
RS	I	Reset for initializing the device
VCC	I	+5 V supply
VSS	I	Ground
WE	O	Write enable for device output data on D15-D0
X1	O	Crystal output for internal oscillator
X2/CLKIN	I	Crystal input for internal oscillator or external system clock input

†See EPROM programming section.
‡Input/Output/High-impedance state.

functional block diagram (TMS32010, TMS320C10, TMS320C15, TMS320E15)



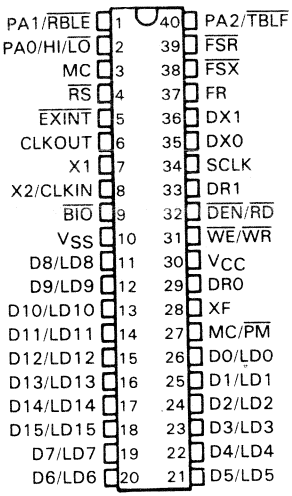
description

The TMS320C17, like the TMS320C15, has 256 words of on-chip data RAM and 4K words of on-chip program ROM. The TMS320C17 is object-code compatible with the TMS32010 The TMS320C17 provides a dual-channel serial port designed specifically to interface to two combo-codecs. A 16-bit coprocessor interface is also provided for interfacing to common 4/8/16/32-bit microcomputers/microprocessors.

architecture

The TMS320C17 consists of five major functional units: the TMS320C15 microcomputer, a system control register, a full-duplex dual-channel serial port, companding hardware, and a coprocessor port.

TMS320C17, TMS320E17
N/JD PACKAGE
(TOP VIEW)

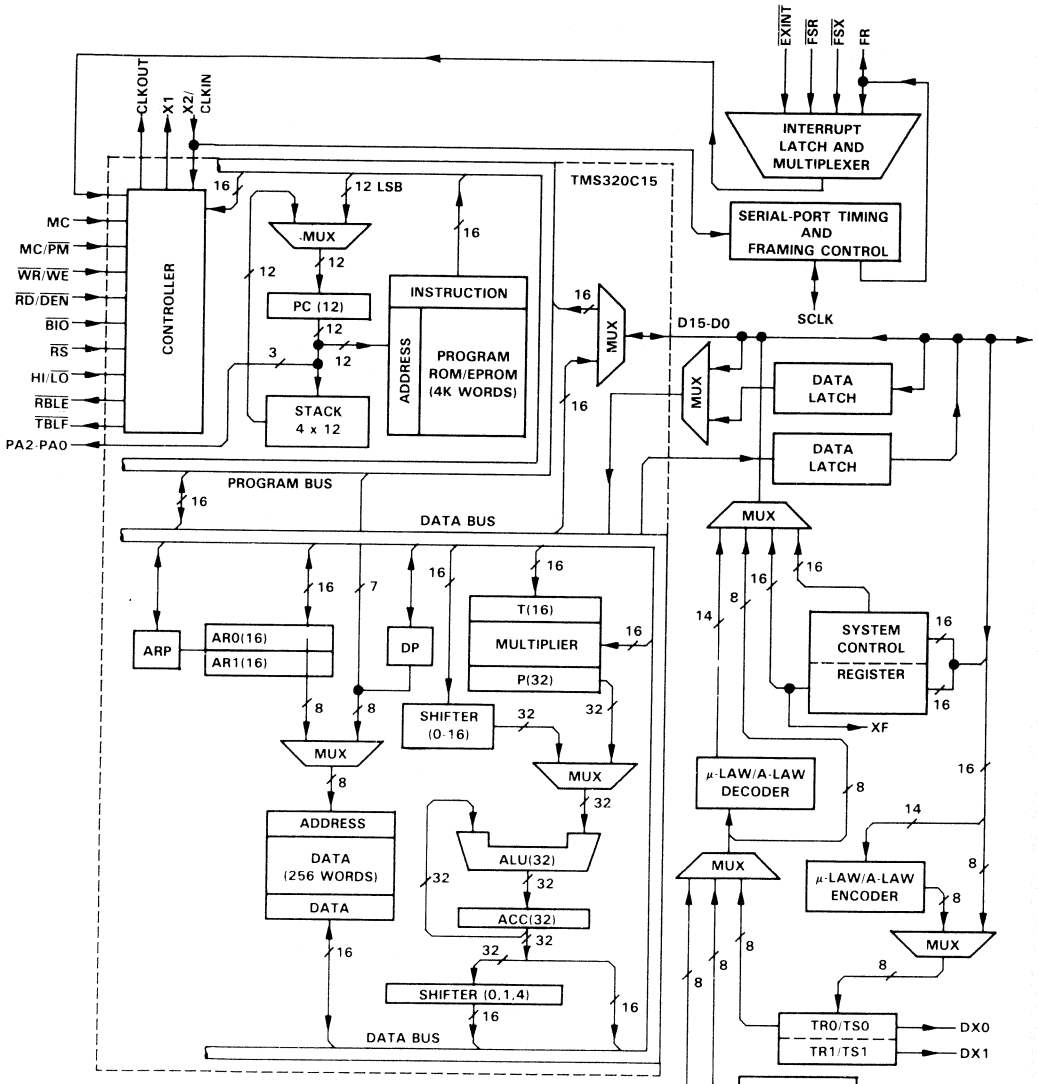


PIN NOMENCLATURE (TMS320C17, TMS320E17†)

NAME	I/O/Z‡	DEFINITION
$\overline{BIO}$	I	External polling input
CLKOUT	O	System clock output, ¼ crystal/CLKIN frequency
D15/LD15-D0/LD0	I/O/Z	16-bit parallel data bus/data lines for coprocessor latch
$\overline{DEN}/\overline{RD}$	I/O/Z	Data enable for device input data/external read for output latch
DR1, DR0	I	Serial-port receive-channel inputs
DX1, DX0	O/Z	Serial-port transmit-channel outputs
$\overline{EXINT}$	I	External interrupt input
FR	O	Internal serial-port framing output
$\overline{FSR}$	I	External serial-port receive framing input
$\overline{FSX}$	I	External serial-port transmit framing input
MC	I	Microcomputer select (must be same state as MC/$\overline{PM}$)
MC/$\overline{PM}$	I	Microcomputer/peripheral coprocessor select (must be same state as MC)
PA0/HI/$\overline{LO}$	I/O/Z	I/O port address output/latch byte select pin
PA1/$\overline{RBLE}$	O	I/O port address output/receive buffer latch empty flag
PA2/$\overline{TBLF}$	O	I/O port address output/transmit buffer latch full flag
$\overline{RS}$	I	Reset for initializing the device
SCLK	I/O/Z	Serial-port clock
V_{CC}	I	+5 V Supply
V_{SS}	I	Ground
$\overline{WE}/\overline{WR}$	I/O	Write enable for device output data/external write for input latch
X1	O	Crystal output for internal oscillator
X2/CLKIN	I	Crystal input for internal oscillator or external oscillator system clock input
XF	O	External-flag output pin

†See EPROM programming section.
‡Input/Output/High impedance state.

functional block diagram (TMS320C17, TMS320E17)



LEGEND:

ACC = ACCUMULATOR	PC = PROGRAM COUNTER
ARP = AUXILIARY REGISTER POINTER	P = P REGISTER
AR0 = AUXILIARY REGISTER 0	T = T REGISTER
AR1 = AUXILIARY REGISTER 1	TR = TRANSMIT REGISTER
DP = DATA PAGE POINTER	RR = RECEIVE REGISTER

architecture (continued)

Three of the I/O ports are used by the serial port, companding hardware, and the coprocessor port. Their operation is determined by the 32 bits of the system control register (see Table 7 for the TMS320C17 control register definitions). Control register 0, accessed through port 0, consists of the lower 16 register bits (CR15-CR0), and is used to control the interrupts, serial port connections, and companding hardware operation. Port 1 accesses control register 1, consisting of the upper 16 control bits (CR31-CR16), as well as both serial port channels, the companding hardware, and the coprocessor port channels. Communication with the control register is via IN and OUT instructions to ports 0 and 1.

Interrupts fully support the TMS320C17 serial port interface. Four maskable interrupts ($\overline{\text{EXINT}}$, FR, $\overline{\text{FSX}}$, and $\overline{\text{FSR}}$) are mapped into I/O port 0 via control register 0. When disabled, these interrupts may be used as single-bit logic inputs polled by software.

serial port

The dual-channel serial port is capable of full-duplex serial communication and offers direct interface to two-combo-codecs. Two receive and two transmit registers are mapped into I/O port 1, and operate with 8-bit data samples. Internal and external framing signals for serial port transfers (MSB first) are selected via the system control register. The serial port clock, SCLK, provides the bit timing for transfers with the serial port, and may be either an input or output. As an input, an external clock provides the timing for data transfers and framing pulse synchronization. As an output, SCLK provides the timing for standalone serial communication and is derived from the TMS320C17 system clock, X2/CLKIN and system control register bits CR27-CR24. See Table 6 for the available divide ratios.
The internal framing (FR) pulse frequency is derived from the serial port clock (SCLK) and system control register bits CR23-CR16. This framing pulse signal provides framing pulses for combo-codecs, for a sample clock for voice-band systems, or for a timer used in control applications.

TABLE 6. SERIAL CLOCK (SCLK) DIVIDE RATIOS (X2/CLKIN = 20.48 MHZ)

CR27	CR26	CR25	CR24	DIVIDE RATIO	SCLK FREQUENCY	UNIT
0	0	0	0	32	0.640	MHz
0	0	0	1	28	0.731	MHz
0	0	1	0	24	0.853	MHz
0	1	0	0	20	1.024	MHz
1	0	0	0	16	1.280	MHz
1	0	0	1	14	1.463	MHz
1	0	1	0	12	1.706	MHz
1	1	0	0	10	2.048	MHz

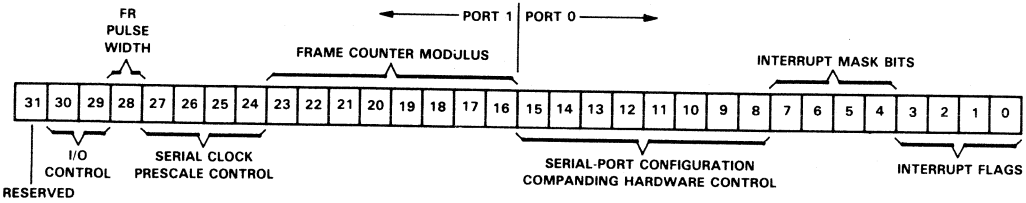
μ-law/A-law companding hardware

The TMS320C17 features hardware companding logic than can operate in either μ-law or A-law format with either sign-magnitude or two's-complement numbers. Data may be companded in either a serial mode for operation on serial port data or a parallel mode for computation inside the device. The companding logic operation is selected through CR14. No bias is required when operating in two's complement. A bias of 33 is required for sign magnitude in μ-law companding. Upon reset, the device is programmed to operate in sign-magnitude mode. This mode can be changed by modifying control bit 29 (CR29) in control register 1.

In the serial mode, sign-magnitude linear PCM (13 magnitude bits plus 1 sign bit for μ-law format or 12 magnitude bits plus 1 sign bit for A-law format) is compressed to 8-bit sign-magnitude logarithmic PCM by the encoder and sent to the transmit register for transmission on an active framing pulse. The decoder converts 8-bit sign-magnitude log PCM from the serial port receive registers to sign-magnitude linear PCM.

In the parallel mode, the serial port registers are disabled to allow parallel data from internal memory to be encoded or decoded for computation inside the device. In the parallel encode mode, the encoder is enabled and a 14-bit sign-magnitude value written to port 1. The encoded value is returned with an IN instruction from port 1. In the parallel decode mode, the decoder is enabled and an 8-bit sign-magnitude log PCM value written to port 1. On the successive IN instruction from port 1, the decoded value is returned.

TABLE 7. CONTROL REGISTER CONFIGURATION

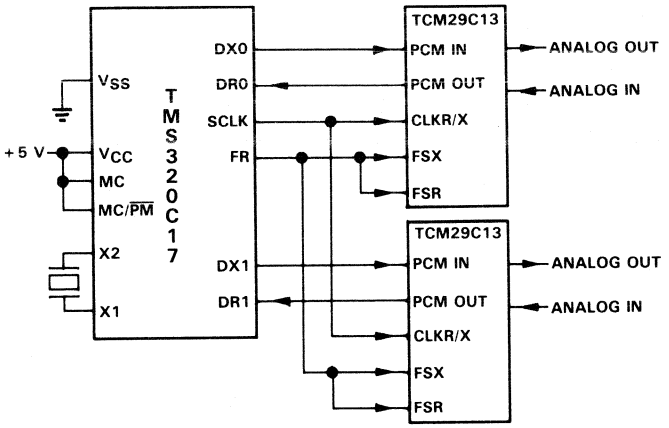


BIT	DESCRIPTION AND CONFIGURATION
0	$\overline{\text{EXINT}}$ interrupt flag†
1	$\overline{\text{FSR}}$ interrupt flag†
2	$\overline{\text{FSX}}$ interrupt flag†
3	FR interrupt flag†
4	$\overline{\text{EXINT}}$ interrupt enable mask. When set to logic 1, an interrupt on $\overline{\text{EXINT}}$ activates device interrupt circuitry.
5	$\overline{\text{FSR}}$ interrupt enable mask. Same as $\overline{\text{EXINT}}$ control.
6	$\overline{\text{FSX}}$ interrupt enable mask. Same as $\overline{\text{EXINT}}$ control.
7	FR interrupt enable mask. Same as $\overline{\text{EXINT}}$ control.
8	Port 1 configuration control: 0 = port 1 connects to either serial-port registers or companding hardware. 1 = port 1 accesses CR31-CR16.
9	External framing enable: 0 = serial-port data transfers controlled by active FR. 1 = serial-port data transfers controlled by active $\overline{\text{FSX}}$/$\overline{\text{FSR}}$.
10	XF external logic output flag latch
11	Serial-port enable: 0 = parallel companding mode; serial port disabled. 1 = serial companding mode; serial port registers enabled.
12	μ-law/A-law encoder enable: 0 = disabled. 1 = data written to port 1 is μ-law or A-law encoded.
13	μ-law/A-law decoder enable: 0 = disabled. 1 = data read from port 1 is μ-law or A-law decoded.
14	μ-law or A-law encode/decode select: 0 = companding hardware performs μ-law conversion. 1 = companding hardware performs A-law conversion.
15	Serial clock control: 0 = SCLK is an output, derived from the prescaler in timing logic. 1 = SCLK is an input that provides the clock for serial port and frame counter in timing logic.
23-16	Frame counter modulus. Controls FR frequency = SCLK/(CNT + 2) where CNT is binary value of CR23-CR16.‡
27-24	SCLK prescale control bits. (See Table 6 for divide ratios.)
28	FR pulse-width control: 0 = fixed-data rate; FR is 1 SCLK cycle wide. 1 = variable-data rate; FR is 8 SCLK cycles wide.
29	Two's-complement μ-law/A-law conversion enable: 0 = sign-magnitude companding 1 = two's-complement companding
30	8/16-bit length coprocessor mode select: 0 = 8-bit byte length 1 = 16-bit word length
31	Reserved for future expansion. Should be set zero.

† Interrupt flag is cleared by writing a logic 1 to the bit with an OUT instruction to port 0.
‡ All ones in CR23-CR16 indicate a degenerative state and should be avoided. Bits are operational whether SCLK is an input or an output. CNT must be greater than 7.

μ-law/A-law companding hardware (continued)

The following diagram shows a TMS320C17 interface to two codecs as used for μ-law or A-law companding format.



coprocessor port

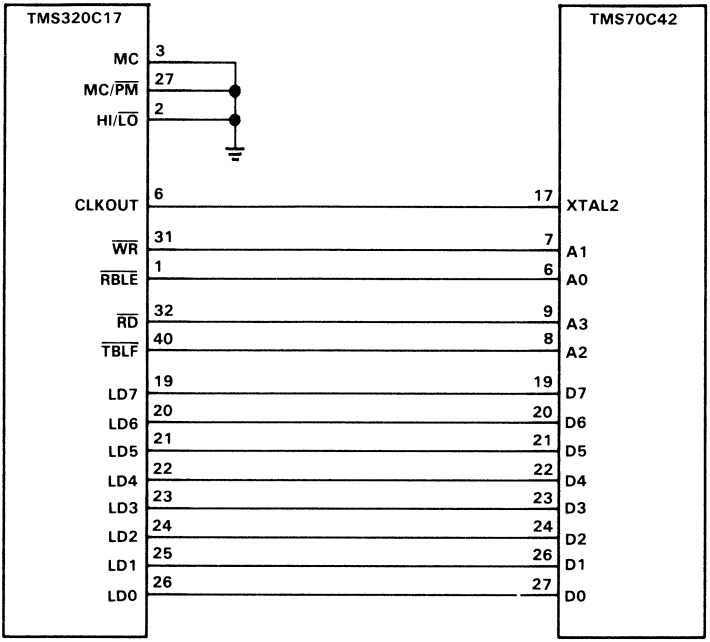
The coprocessor port, accessed through I/O port 5 using IN and OUT instructions, provides a direct connection to most 4/8-bit microcomputers and 16/32-bit microprocessors. The coprocessor interface allows the TMS320C17 to act as a peripheral (slave) microcomputer to a microprocessor, or a master to a peripheral microcomputer such as TMS7042. The coprocessor port is enabled by setting MC/$\overline{\text{PM}}$ and MC low. The microcomputer mode is enabled by setting these two pins high. (Note that MC/$\overline{\text{PM}}$ $\neq$ MC is undefined.) The 16 data lines are used for the 6 parallel 16-bit I/O ports.

In the coprocessor mode, the 16-bit coprocessor port is reconfigured to operate as a 16-bit latched bus interface. Control bit 30 (CR30) in control register 1 is used to configure the coprocessor port to either an 8-bit or a 16-bit length. When CR30 is high, the coprocessor port is 16 bits wide, thereby making all 16 bits of the data port available for 16-bit transfers to 16 and 32-bit microprocessors. When CR30 is low, the port is 8 bits wide and mapped to the low byte of the data port for interfacing to 8-bit microcomputers. When operating in the 8-bit mode, both halves of the 16-bit latch can be addressed using the HI/$\overline{\text{LO}}$ pin, thus allowing 16-bit transfers over 8 data lines. When not in the coprocessor mode, port 5 can be used as a generic I/O port.

The external processor recognizes the coprocessor interface, in which both processors run asynchronously, as a memory-mapped I/O operation. The external processor lowers the $\overline{\text{WR}}$ line and places data on the bus. It next raises the $\overline{\text{WR}}$ line to clock the data into the on-chip latch. The rising edge of $\overline{\text{WR}}$ automatically creates an interrupt to the TMS320C17 and the falling edge of $\overline{\text{WR}}$ clears the $\overline{\text{RBLE}}$ (receive buffer latch empty) flag. Likewise, the external processor reads from the latch by driving the $\overline{\text{RD}}$ line active low, thus enabling the output latch to drive the latched data. When the data has been read, the external device will again bring the $\overline{\text{RD}}$ line high. This activates the $\overline{\text{BIO}}$ line to signal that the transfer is complete and the latch is available for the next transfer. The falling edge of $\overline{\text{RD}}$ resets the $\overline{\text{TBLF}}$ (transmit buffer latch full) flag. Note that the $\overline{\text{EXINT}}$ and $\overline{\text{BIO}}$ lines are reserved for coprocessor interface and cannot be driven externally when in the coprocessor mode.

coprocessor port (continued)

An example of the use of a coprocessor interface is shown below, in which the TMS320C17 is interfaced to the TMS70C42, an 8-bit microcontroller.



NMOS DEVICE ELECTRICAL SPECIFICATIONS

This section contains all the electrical specifications for the TMS320 NMOS first-generation devices. Refer to the top corner for the specific device.

absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{CC}‡ . . . . . −0.3 V to 7 V
Input voltage range . . . . . −0.3 V to 15 V
Output voltage range . . . . . −0.3 V to 15 V
Continuous power dissipation . . . . . 1.5 W
Air temperature range above operating device . . . . . 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		4.75	5	5.25	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	All inputs except CLKIN	2			V
		CLKIN	2.8			
V_{IL}	Low-level input voltage (all inputs)				0.8	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		0		70	°C

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
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		V_{CC} = MAX	T_A = 0°C		180	275	mA
				T_A = 70°C			235§	
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 except for I_{CC} are at V_{CC} = 5 V, T_A = 25°C.
‡I_{CC} characteristics are inversely proportional to temperature; i.e., I_{CC} decreases approximately linearly with temperature.
§Value derived from characterization data and not tested.

PARAMETER MEASUREMENT INFORMATION

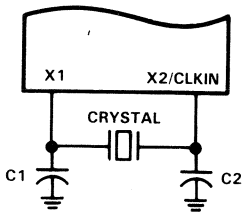


FIGURE 1. INTERNAL CLOCK OPTION

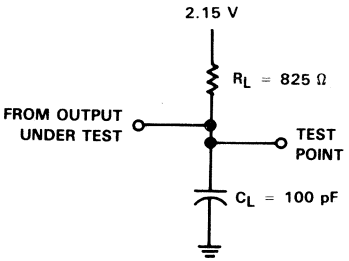


FIGURE 2. TEST LOAD CIRCUIT

input synchronization requirements

For systems using asynchronous inputs to the $\overline{INT}$ and $\overline{BIO}$ pins on the TMS32010, the external hardware shown in the diagrams below is recommended to ensure proper execution of interrupts and the BIOZ instruction. This hardware synchronizes the $\overline{INT}$ and $\overline{BIO}$ input signals with the rising edge of CLKOUT on the TMS32010. The pulse width required for these input signals is $t_{c(C)}$, which is one TMS32010 clock cycle, plus sufficient setup time for the flip-flop (dependent upon the flip-flop used). Note that these input synchronization requirements apply only to NMOS versions of the TMS32010 and not to other members of the TMS320 family.

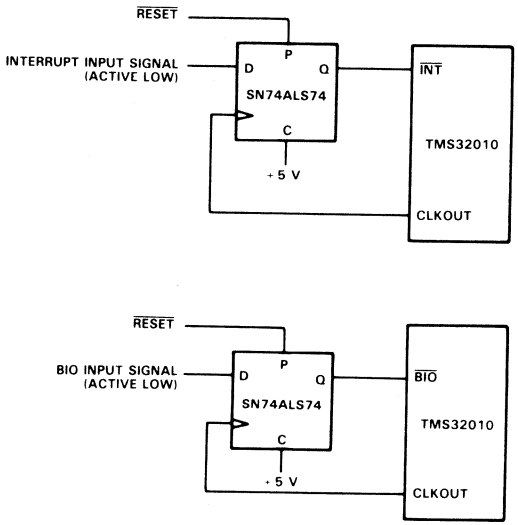


FIGURE 3. ASYNCHRONOUS INPUT SYNCHRONIZATION CIRCUITS

CLOCK CHARACTERISTICS AND TIMING

The TMS32010 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and be specified at a load capacitance of 20 pF.

PARAMETER	TEST CONDITIONS	TMS32010			TMS32010-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Crystal frequency f_x	0°C to 70°C	6.7		20.5	6.7		25.0	MHz
C1, C2	0°C to 70°C		10			10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS32010			TMS32010-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(C)}$	CLKOUT cycle time†	R_L = 825 Ω, C_L = 100 pF, See Figure 2	195.12	200		160			ns
$t_{r(C)}$	CLKOUT rise time			10			10		ns
$t_{f(C)}$	CLKOUT fall time			8			8		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92			74		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90			72		ns
$t_{d(MCC)}$	Delay time CLKIN↑ to CLKOUT↓		25‡		60‡	25‡		60‡	ns

†$t_{c(C)}$ is the cycle time of CLKOUT, i.e., $4 * t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
‡Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TMS32010			TMS32010-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(MC)}$	Master clock cycle time	48.78	50	150	40		150	ns
$t_{r(MC)}$	Rise time master clock input		5†	10†		5†	10†	ns
$t_{f(MC)}$	Fall time master clock input		5†	10†		5†	10†	ns
$t_{w(MCP)}$	Pulse duration master clock	$0.475t_{c(MC)}$†		$0.525t_{c(MC)}$†	$0.475t_{c(MC)}$†		$0.525t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration master clock low, $t_{c(MC)}$ = 50 ns		20†			18†		ns
$t_{w(MCH)}$	Pulse duration master clock high, $t_{c(MC)}$ = 50 ns		20†			18†		ns

†Values derived from characterization data and not tested.

.MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS32010			TMS32010-25			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{d1}	Delay time CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF, See Figure 2	10†		50	10†		40	ns
t_{d2}	Delay time CLKOUT↓ to $\overline{MEN}$↓		¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 15	¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 12	ns
t_{d3}	Delay time CLKOUT↓ to $\overline{MEN}$↑		− 10†		15	− 10†		12	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{DEN}$↓		¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 15	¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 12	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{DEN}$↑		− 10†		15	− 10†		12	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{WE}$↓		½$t_{c(C)}$ − 5†		½$t_{c(C)}$ + 15	½$t_{c(C)}$ − 5†		½$t_{c(C)}$ + 12	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{WE}$↑		− 10†		15	− 10†		12	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid				¼$t_{c(C)}$ + 65			¼$t_{c(C)}$ + 52	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		¼$t_{c(C)}$ − 5†			¼$t_{c(C)}$ − 5†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				¼$t_{c(C)}$ + 30†			¼$t_{c(C)}$ + 30†	ns
t_v	Data bus OUT valid after CLKOUT↓		¼$t_{c(C)}$ − 10			¼$t_{c(C)}$ − 10			ns
$t_{h(A\text{-}WMD)}$	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑ or $\overline{DEN}$↑ (see Note 1)		0			0			ns
$t_{su(A\text{-}MD)}$	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		¼$t_{c(C)}$ − 45			¼$t_{c(C)}$ − 35			ns

†Values derived from characterization data and not tested.
NOTE 1: Address bus will be valid upon $\overline{WE}$↑, $\overline{DEN}$↑, or $\overline{MEN}$↑.

timing requirements over recommended operating conditions

		TEST CONDITIONS	TMS32010			TMS32010-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(D)}$	Setup time data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF, See Figure 2	50			40			ns
$t_{h(D)}$	Hold time data bus held valid after CLKOUT↓ (see Note 2)		0			0			ns

NOTE 2: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time $\overline{DEN}\uparrow$, $\overline{WE}\uparrow$, and $\overline{MEN}\uparrow$ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF, See Figure 2			$\frac{1}{2}t_{c(C)} + 50^\dagger$	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$\frac{1}{4}t_{c(C)} + 50^\dagger$	ns

† Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TMS32010			TMS32010-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(R)}$	Reset $\overline{RS}$ setup time prior to CLKOUT (see Note 3)	50			40			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			$5t_{c(C)}$			ns

NOTE 3: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

		TMS32010			TMS32010-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(INT)}$	Fall time ($\overline{INT}$)			15			15	ns
$t_{w(INT)}$	Pulse duration $\overline{INT}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time $\overline{INT}\downarrow$ before $\overline{CLKOUT}\downarrow$	50			40			ns

I/O ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

		TMS32010			TMS32010-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(IO)}$	Fall time $\overline{BIO}$			15			15	ns
$t_{w(IO)}$	Pulse duration $\overline{BIO}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time $\overline{BIO}\downarrow$ before CLKOUT↓	50			40			ns

CLOCK CHARACTERISTICS AND TIMING

The TMS32010-16 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and be specified at a load capacitance of 20 pF.

PARAMETER	TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency f_X	0°C to 70°C	6.7		16	MHz
C1, C2	0°C to 70°C		10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

	PARAMETER	TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLKOUT cycle time†	R_L = 870 Ω, C_L = 100 pF, See Figure 2	250			ns
$t_{r(C)}$	CLKOUT rise time			12		ns
$t_{f(C)}$	CLKOUT fall time			10		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			115		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			113		ns
$t_{d(MCC)}$	Delay time CLKIN↑ to CLKOUT↓		25‡		60‡	ns

†$t_{c(C)}$ is the cycle time of CLKOUT, i.e., 4*$t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
‡Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(MC)}$	Master clock cycle time	62.5		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	0.475$t_{c(MC)}$†		0.525$t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration master clock low, $t_{c(MC)}$ = 50 ns	22	30†		ns
$t_{w(MCH)}$	Pulse duration master clock high, $t_{c(MC)}$ = 50 ns	22	30†		ns

†Values derived from characterization data and not tested.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

	PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d1}	Delay time CLKOUT↓ to address bus valid (see Note 1)	R_L = 870 Ω, C_L = 100 pF, See Figure 2	$10^\dagger$		50	ns
t_{d2}	Delay time CLKOUT↓ to $\overline{MEN}$↓		$\frac{1}{4}t_{c(C)} - 5^\dagger$		$\frac{1}{4}t_{c(C)} + 15$	ns
t_{d3}	Delay time CLKOUT↓ to $\overline{MEN}$↑		$-10^\dagger$		15	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{DEN}$↓		$\frac{1}{4}t_{c(C)} - 5^\dagger$		$\frac{1}{4}t_{c(C)} + 15$	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{DEN}$↑		$-10^\dagger$		15	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{WE}$↓		$\frac{1}{2}t_{c(C)} - 5^\dagger$		$\frac{1}{2}t_{c(C)} + 15$	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{WE}$↑		$-10^\dagger$		15	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid				$\frac{1}{4}t_{c(C)} + 65$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$\frac{1}{4}t_{c(C)} - 5^\dagger$			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				$\frac{1}{4}t_{c(C)} + 30^\dagger$	ns
t_v	Data bus OUT valid after CLKOUT↓		$\frac{1}{4}t_{c(C)} - 10$			ns
$t_{h(A\text{-}WMD)}$	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑ (see Note 1)		0			ns
$t_{su(A\text{-}MD)}$	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		$\frac{1}{4}t_{c(C)} - 45$			ns

†Values derived from characterization data and not tested.
NOTE 1: Address bus will be valid upon $\overline{WE}$↑, $\overline{DEN}$↑, or $\overline{MEN}$↑.

timing requirements over recommended operating conditions

		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{su(D)}$	Setup time data bus valid prior to CLKOUT↓	R_L = 870 Ω, C_L = 100 pF, See Figure 2	50			ns
$t_{h(D)}$	Hold time data bus held valid after CLKOUT↓ (see Note 2)		0			ns

NOTE 2: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 870 Ω, C_L = 100 pF, See Figure 2			½$t_{c(C)}$ + 50†	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				¼$t_{c(C)}$ + 50†	ns

†Values 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 3)	50			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	5$t_{c(C)}$			ns

NOTE 3: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

		MIN	TYP	MAX	UNIT
$t_{f(INT)}$	Fall time $\overline{INT}$			15	ns
$t_{w(INT)}$	Pulse duration $\overline{INT}$	$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time $\overline{INT}$↓ before CLKOUT↓	50			ns

I/O ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

		MIN	TYP	MAX	UNIT
$t_{f(IO)}$	Fall time $\overline{BIO}$			15	ns
$t_{w(IO)}$	Pulse duration $\overline{BIO}$	$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time $\overline{BIO}$↓ before CLKOUT↓	50			ns

CMOS DEVICE ELECTRICAL SPECIFICATIONS

This section contains all the electrical specifications for the TMS320 CMOS first-generation devices. Refer to the top corner for the specific device.

Absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{CC}‡ . . . . . −0.3 V to 7 V
Input voltage range . . . . . −0.3 V to 15 V
Output voltage range . . . . . −0.3 V to 15 V
Continuous power dissipation: 20-MHz version . . . . . 0.3 W
25-MHz version . . . . . 0.35 W
Air temperature range above operating device: L version . . . . . 0°C to 70°C
A version . . . . . −40°C to 85°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
		All other devices	4.5	5	5.5	
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	All inputs except CLKIN	2			V
		CLKIN	3			
V_{IL}	Low-level input voltage	All inputs except MC/$\overline{MP}$			0.8	V
		MC/$\overline{MP}$			0.6	
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
		A version	−40		85	°C

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 6)		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	
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.
‡Values derived from characterization data and not tested.
NOTE 6: 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.

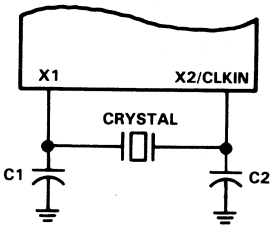


FIGURE 4. INTERNAL CLOCK OPTION

PARAMETER MEASUREMENT INFORMATION

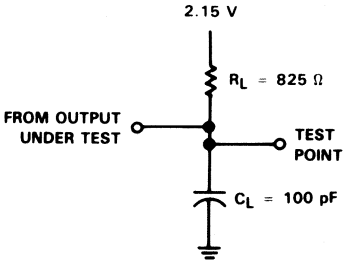


FIGURE 5. TEST LOAD CIRCUIT

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_{CC}‡ Supply current	TMS320C10	f = 20.5 MHz, V_{CC} = 5.5 V		33	55	mA
	TMS320C10-25	f = 25.6 MHz, V_{CC} = 5.5 V		40	65	mA

†All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡I_{CC} characteristics are inversely proportional to temperature. For I_{CC} dependance on temperature, frequency, and loading, see Figure 9.

CLOCK CHARACTERISTICS AND TIMING

The TMS320C10 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 4). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and be specified at a load capacitance of 20 pF.

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency f_x	TMS320C10	T_A = −40°C to 85°C	6.7		20.5	MHz
	TMS320C10-25	T_A = −40°C to 85°C	6.7		25.6	MHz
C1, C2		T_A = −40°C to 85°C		10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS320C10			TMS320C10-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(C)}$	CLKOUT cycle time†	R_L = 825 Ω, C_L = 100 pF, See Figure 5	195.12	200		156.25	160		ns
$t_{r(C)}$	CLKOUT rise time			10‡			10‡		ns
$t_{f(C)}$	CLKOUT fall time			8‡			8‡		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92‡			72‡		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90‡			70‡		ns
$t_{d(MCC)}$	Delay time CLKIN↑ to CLKOUT↓		25‡		60‡	25‡		50‡	ns

†$t_{c(C)}$ is the cycle time of CLKOUT, i.e., 4 * $t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
‡Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TMS320C10			TMS320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(MC)}$	Master clock cycle time	48.78	50	150	39.06	40	150	ns
$t_{r(MC)}$	Rise time master clock input		5†	10†		5†	10†	ns
$t_{f(MC)}$	Fall time master clock input		5†	10†		5†	10†	ns
$t_{w(MCP)}$	Pulse duration master clock	$0.4t_{c(MC)}$†		$0.6t_{c(MC)}$†	$0.45t_{c(MC)}$†		$0.55t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration master clock low		20†			15†		ns
$t_{w(MCH)}$	Pulse duration master clock high		20†			15†		ns

†Values derived from characterization data and not tested.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS320C10 MIN	TMS320C10 TYP	TMS320C10 MAX	TMS320C10-25 MIN	TMS320C10-25 TYP	TMS320C10-25 MAX	UNIT
t_{d1}	Delay time CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF, See Figure 5	10†		50	10†		40	ns
t_{d2}	Delay time CLKOUT↓ to $\overline{MEN}$↓		¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 15	¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 12	ns
t_{d3}	Delay time CLKOUT↓ to $\overline{MEN}$↑		− 10†		15	− 10†		12	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{DEN}$↓		¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 15	¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 12	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{DEN}$↑		− 10†		15	− 10†		12	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{WE}$↓		½$t_{c(C)}$ − 5†		½$t_{c(C)}$ + 15	½$t_{c(C)}$ − 5†		½$t_{c(C)}$ + 12	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{WE}$↑		− 10†		15	− 10†		12	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid				¼$t_{c(C)}$ + 65			¼$t_{c(C)}$ + 52	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		¼$t_{c(C)}$ − 5†			¼$t_{c(C)}$ − 5†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				¼$t_{c(C)}$ + 30†			¼$t_{c(C)}$ + 30†	ns
t_v	Data bus OUT valid after CLKOUT↓		¼$t_{c(C)}$ − 10			¼$t_{c(C)}$ − 10			ns
$t_{h(A\text{-}WMD)}$	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑ (see Note 7)		− 10			− 10			ns
$t_{su(A\text{-}MD)}$	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		¼$t_{c(C)}$ − 45			¼$t_{c(C)}$ − 35			ns

†Values derived from characterization data and not tested.
NOTE 7: For interfacing I/O devices, see Figure 6.

timing requirements over recommended operating conditions

		TEST CONDITIONS	TMS320C10 MIN	TMS320C10 NOM	TMS320C10 MAX	TMS320C10-25 MIN	TMS320C10-25 NOM	TMS320C10-25 MAX	UNIT
$t_{su(D)}$	Setup time data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF, See Figure 5	50			40			ns
$t_{h(D)}$	Hold time data bus held valid after CLKOUT↓ (see Note 2)		0			0			ns

NOTE 2: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

SUGGESTED I/O DECODE CIRCUIT

The circuit shown in Figure 6 is a design example for interfacing I/O devices to the TMS320C10. This circuit decodes the address for output operations using the OUT instruction. The same circuit can be used to decode input and output operations if the inverter ('ALS04) is replaced with a NAND gate and both $\overline{DEN}$ and $\overline{WE}$ are connected. Inputs and outputs can be decoded at the same port provided the output of the decoder ('AS137) is gated with the appropriate signal ($\overline{DEN}$ or $\overline{WE}$) to select read or write (using an 'ALS32). Access times can be increased when the circuit shown in Figure 6 is repeated to support IN instructions with $\overline{DEN}$ connected rather than $\overline{WE}$.

The table write (TBLW) function requires a different circuit. A detailed discussion of an example circuit for this function is described on page 315 of the application report, "Interfacing External Memory to the TMS32010," published in the book, *Digital Signal Processing Applications with the TMS320 Family*. A schematic of this circuit as shown on page 318 of that book.

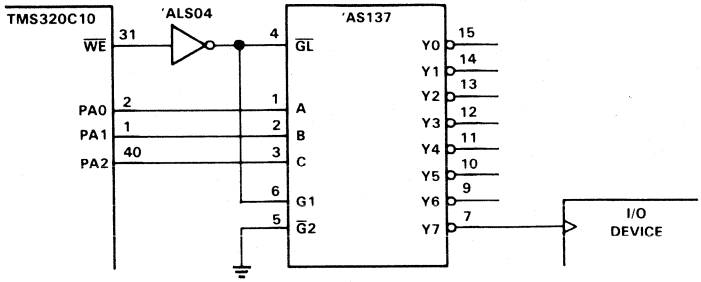


FIGURE 6. I/O DECODE CIRCUIT

RESET ($\overline{RS}$) TIMING

witching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF, See Figure 5			$\frac{1}{2}t_{c(C)} + 50$↑	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$\frac{1}{4}t_{c(C)} + 50$↑	ns

These values were derived from characterization data and not tested.

iming requirements over recommended operating conditions

		TMS320C10			TMS320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 3)	50			40			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			$5t_{c(C)}$			ns

OTE 3: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{INT}$) TIMING

iming requirements over recommended operating conditions

		TMS320C10			TMS320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(INT)}$	Fall time $\overline{INT}$			15			15	ns
$t_{w(INT)}$	Pulse duration $\overline{INT}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time $\overline{INT}$↓ before CLKOUT↓	50			40			ns

I/O ($\overline{BIO}$) TIMING

iming requirements over recommended operating conditions

		TMS320C10			TMS320C10-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(IO)}$	Fall time $\overline{BIO}$			15			15	ns
$t_{w(IO)}$	Pulse duration $\overline{BIO}$	$t_{c(C)}$			$t_{c(C)}$		ns	
$t_{su(IO)}$	Setup time $\overline{BIO}$↓ before CLKOUT↓	50			40			ns

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_{CC}‡ Supply current	TMS320C15	f = 20.5 MHz, V_{CC} = 5.5 V		45		mA
	TMS320C15-25	f = 25.6 MHz, V_{CC} = 5.5 V		50		
	TMS320E15	f = 20.5 MHz, V_{CC} = 5.25 V		60		

†All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡I_{CC} characteristics are inversely proportional to temperature. For I_{CC} dependance on temperature, frequency, and loading, see Figure 9.

CLOCK CHARACTERISTICS AND TIMING

The TMS320C15/E15 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 4). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and be specified at a load capacitance of 20 pF.

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency f_x	TMS320C15/E15	T_A = 0°C to 70°C	6.7		20.5	MHz
	TMS320C15-25	T_A = 0°C to 70°C	6.7		25.6	MHz
C1, C2		T_A = 0°C to 70°C		10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS320C15/E15			TMS320C15-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(C)}$	CLKOUT cycle time†	R_L = 825 Ω, C_L = 100 pF, See Figure 5	195.12	200		156.25	160		ns
$t_{r(C)}$	CLKOUT rise time			10‡			10‡		ns
$t_{f(C)}$	CLKOUT fall time			8‡			8‡		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92‡			72‡		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90‡			70‡		ns
$t_{d(MCC)}$	Delay time CLKIN↑ to CLKOUT↓		25‡		60‡	25‡		50‡	ns

†$t_{c(C)}$ is the cycle time of CLKOUT, i.e., 4*$t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
‡Values derived from characterization data and not tested.

iming requirements over recommended operating conditions

		TMS320C15/E15			TMS320C15-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(MC)}$	Master clock cycle time	48.78	50	150	39.06	40	150	ns
$t_{r(MC)}$	Rise time master clock input		5†	10†		5†	10†	ns
$t_{f(MC)}$	Fall time master clock input		5†	10†		5†	10†	ns
$t_{w(MCP)}$	Pulse duration master clock	$0.4t_{c(MC)}$†		$0.6t_{c(MC)}$†	$0.45t_{c(MC)}$†		$0.55t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration master clock low		20†			15†		ns
$t_{w(MCH)}$	Pulse duration master clock high		20†			15†		ns

†Values derived from characterization data and not tested.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS320C15/E15			TMS320C15-25			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{d1}	Delay time CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF, See Figure 5		10†	50		10†	40	ns
t_{d2}	Delay time CLKOUT↓ to $\overline{MEN}$↓		$¼t_{c(C)} - 5$†		$¼t_{c(C)} + 15$	$¼t_{c(C)} - 5$†		$¼t_{c(C)} + 12$	ns
t_{d3}	Delay time CLKOUT↓ to $\overline{MEN}$↑			−10†	15		−10†	12	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{DEN}$↓		$¼t_{c(C)} - 5$†		$¼t_{c(C)} + 15$	$¼t_{c(C)} - 5$†		$¼t_{c(C)} + 12$	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{DEN}$↑			−10†	15		−10†	12	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{WE}$↓		$½t_{c(C)} - 5$†		$½t_{c(C)} + 15$	$½t_{c(C)} - 5$†		$½t_{c(C)} + 12$	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{WE}$↑			−10†	15		−10†	12	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid				$¼t_{c(C)} + 65$			$¼t_{c(C)} + 52$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$¼t_{c(C)} - 5$†			$¼t_{c(C)} - 5$†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				$¼t_{c(C)} + 30$†			$¼t_{c(C)} + 30$†	ns
t_v	Data bus OUT valid after CLKOUT↓		$¼t_{c(C)} - 10$			$¼t_{c(C)} - 10$			ns
$t_{h(A-WMD)}$	Address hold time after $\overline{WE}$↑, $\overline{MEN}$↑, or $\overline{DEN}$↑ (see Note 7)		−10			−10			ns
$t_{su(A-MD)}$	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		$¼t_{c(C)} - 45$			$¼t_{c(C)} - 35$			ns

†Values derived from characterization data and not tested.
NOTE 7: For interfacing I/O devices, see Figure 6.

timing requirements over recommended operating conditions

		TEST CONDITIONS	TMS320C15/E15			TMS320C15-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(D)}$	Setup time data bus valid prior to CLKOUT↓	R_L = 825 Ω, C_L = 100 pF, See Figure 5	50			40			ns
$t_{h(D)}$	Hold time data bus held valid after CLKOUT↓ (see Note 2)		0			0			ns

NOTE 2: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT.

ADVANCE INFORMATION

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF, See Figure 5			$\frac{1}{2}t_{c(C)} + 50^\dagger$	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$\frac{1}{4}t_{c(C)} + 50^\dagger$	ns

†These values were derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TMS320C15/E15			TMS320C15-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 3)	50			40			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			$5t_{c(C)}$			ns

NOTE 3: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

		TMS320C15/E15			TMS320C15-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(INT)}$	Fall time $\overline{INT}$			15			15	ns
$t_{w(INT)}$	Pulse duration $\overline{INT}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time $\overline{INT}$↓ before CLKOUT↓	50			40			ns

I/O ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

		TMS320C15/E15			TMS320C15-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(IO)}$	Fall time $\overline{BIO}$			15			15	ns
$t_{w(IO)}$	Pulse duration $\overline{BIO}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time $\overline{BIO}$↓ before CLKOUT↓	50			40			ns

ADVANCE INFORMATION

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_{CC}‡ Supply current	TMS320C17	f = 20.5 MHz, V_{CC} = 5.5 V		50		mA
	TMS320C17-25	f = 25.6 MHz, V_{CC} = 5.5 V		55		
	TMS320E17	f = 20.5 MHz, V_{CC} = 5.25 V		65		

†All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡I_{CC} characteristics are inversely proportional to temperature. For I_{CC} dependance on temperature, frequency, and loading, see Figure 9.

CLOCK CHARACTERISTICS AND TIMING

The TMS320C17/E17 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 4). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and be specified at a load capacitance of 20 pF.

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency f_x	TMS320C17/E17	T_A = 0°C to 70°C	6.7		20.5	MHz
	TMS320C17-25	T_A = 0°C to 70°C	6.7		25.6	MHz
C1, C2		T_A = 0°C to 70°C		10		pF

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS320C17/E17			TMS320C17-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(C)}$	CLKOUT cycle time†	R_L = 825 Ω, C_L = 100 pF, See Figure 5	195.12	200		156.25	160		ns
$t_{r(C)}$	CLKOUT rise time			10‡			10‡		ns
$t_{f(C)}$	CLKOUT fall time			8‡			8‡		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92‡			72‡		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90‡			70‡		ns
$t_{d(MCC)}$	Delay time CLKIN↑ to CLKOUT↓		25‡		60‡	25‡		50‡	ns

†$t_{c(C)}$ is the cycle time of CLKOUT, i.e., 4 * $t_{c(MC)}$ (4 times CLKIN cycle time if an external oscillator is used).
‡Values derived from characterization data and not tested.

ADVANCE INFORMATION

timing requirements over recommended operating conditions

		TMS320C17/E17			TMS320C17-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{c(MC)}$	Master clock cycle time	48.78	50	150	39.06	40	150	ns
$t_{r(MC)}$	Rise time master clock input		5†	10†		5†	10†	ns
$t_{f(MC)}$	Fall time master clock input		5†	10†		5†	10†	ns
$t_{w(MCP)}$	Pulse duration master clock	$0.4t_{c(MC)}$†		$0.6t_{c(MC)}$†	$0.45t_{c(MC)}$†		$0.55t_{c(MC)}$†	ns
$t_{w(MCL)}$	Pulse duration master clock low		20†			15†		ns
$t_{w(MCH)}$	Pulse duration master clock high		20†			15†		ns

†Values derived from characterization data and not tested.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	TMS320C17/E17			TMS320C17-25			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{d1}	Delay time CLKOUT↓ to address bus valid	R_L = 825 Ω, C_L = 100 pF, See Figure 5	10†		50	10†		40	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{DEN}$↓		$¼t_{c(C)} - 5$†		$¼t_{c(C)} + 15$	$¼t_{c(C)} - 5$†		$¼t_{c(C)} + 12$	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{DEN}$↑		−10†		15	−10†		12	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{WE}$↓		$½t_{c(C)} - 5$†		$½t_{c(C)} + 15$	$½t_{c(C)} - 5$†		$½t_{c(C)} + 12$	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{WE}$↑		−10†		15	−10†		12	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid				$¼t_{c(C)} + 65$			$¼t_{c(C)} + 52$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$¼t_{c(C)} - 5$†			$¼t_{c(C)} - 5$†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				$¼t_{c(C)} + 30$†			$¼t_{c(C)} + 30$†	ns
t_v	Data bus OUT valid after CLKOUT↓		$¼t_{c(C)} - 10$			$¼t_{c(C)} - 10$			ns
$t_{h(A\text{-}WMD)}$	Address hold time after $\overline{WE}$↑ or $\overline{DEN}$↑ (see Note 7)		−10			−10			ns
$t_{su(A\text{-}MD)}$	Address bus setup time prior to $\overline{DEN}$↓		$¼t_{c(C)} - 45$			$¼t_{c(C)} - 35$			ns

†Values derived from characterization data and not tested.
NOTE 7: For interfacing I/O devices, see Figure 6.

timing requirements over recommended operating conditions

		TEST CONDITIONS	TMS320C17/E17			TMS320C17-25			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(D)}$	Setup time data bus valid prior to CLKOUT↓	R_L 825 Ω, C_L 100 pF, See Figure 5	50			40			ns
$t_{h(D)}$	Hold time data bus held valid after CLKOUT↓ (see Note 2)		0			0			ns

NOTE 2: Data may be removed from the data bus upon $\overline{DEN}$↑ preceding CLKOUT↓.

ADVANCE INFORMATION

RESET ($\overline{RS}$) TIMING

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF, See Figure 5			½$t_{c(C)}$ + 50†	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				¼$t_{c(C)}$ + 50†	ns
t_{d12}	Delay time from $\overline{RS}$↓ to high-impedance SCLK		100†		200†	ns
t_{d13}	Delay time from $\overline{RS}$↓ to high-impedance DX1, DX0		100†		200†	ns

†Values derived from characterization data and not tested.

timing requirements over recommended operating conditions

		TMS320C17/E17			TMS320C17-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 3)	50			40			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	5$t_{c(C)}$			5$t_{c(C)}$			ns

NOTE 3: RS can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.

INTERRUPT ($\overline{EXINT}$) TIMING

timing requirements over recommended operating conditions

		TMS320C17/E17			TMS320C17-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(INT)}$	Fall time $\overline{EXINT}$			15			15	ns
$t_{w(INT)}$	Pulse duration $\overline{EXINT}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time $\overline{EXINT}$↓ before CLKOUT↓	50			40			ns

I/O ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

		TMS320C17/E17			TMS320C17-25			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
$t_{f(IO)}$	Fall time $\overline{BIO}$			15			15	ns
$t_{w(IO)}$	Pulse duration $\overline{BIO}$	$t_{c(C)}$			$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time $\overline{BIO}$↓ before CLKOUT↓	50			40			ns

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{d(XF)}$	Delay time CLKOUT↓ to valid XF	R_L = 825 Ω, C_L = 100 pF, See Figure 5	5†		115	ns

†Values derived from characterization data and not tested.

ADVANCE INFORMATION

SERIAL PORT TIMING

switching characteristics over recommended operating conditions

PARAMETER		MIN	TYP	MAX	UNIT
$t_{d(CH-FR)}$	Internal framing (FR) delay from SCLK rising edge			70	ns
$t_{d(DX1-CL)}$	DX bit 1 valid before SCLK falling edge	20			ns
$t_{d(DX2-CL)}$	DX bit 2 valid before SCLK falling edge	20			ns

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(SCLK)}$	Serial port clock (SCLK) cycle time (see Note 8)	390		4770	ns
$t_{f(SCLK)}$	Serial port clock (SCLK) fall time			30	ns
$t_{r(SCLK)}$	Serial port clock (SCLK) rise time			30	ns
$t_{w(SCLKL)}$	Serial port clock (SCLK) low-pulse duration (see Note 9)	185		2500	ns
$t_{w(SCLKH)}$	Serial port clock (SCLK) high-pulse duration (see Note 9)	185		2500	
$t_{su(FS)}$	$\overline{FSX}$/$\overline{FSR}$ setup time before SCLK falling edge	100			ns
$t_{su(DR)}$	DR setup time before SCLK falling edge	20			ns
$t_{h(DR)}$	DR hold time after SCLK falling edge	20			ns
$t_{h(DX)}$	DX hold time after SCLK falling edge	$t_{c(SCLK)}/2$			ns

NOTES: 8. Minimum cycle time is $2t_{c(C)}$ where $t_{c(C)}$ is CLKOUT cycle time.
9. The duty cycle of the serial port clock must be within 45 to 55 percent.

COPROCESSOR INTERFACE TIMING

switching characteristics over recommended operating conditions

PARAMETER		MIN	NOM	MAX	UNIT
$t_{d(R-A)}$	$\overline{RD}$ low to $\overline{TBLF}$ high			75	ns
$t_{d(W-A)}$	$\overline{WR}$ low to $\overline{RBLE}$ high			75	ns
$t_{a(LPR)}$	$\overline{RD}$ low to data valid			60	ns
$t_{h(LPR)}$	Data hold time after $\overline{RD}$ high	50			ns

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{h(BA)}$	Byte address hold time after $\overline{WR}$ or $\overline{RD}$ high	25			ns
$t_{su(BA)}$	Byte address setup time to $\overline{WR}$ or $\overline{RD}$ low	40			ns
$t_{su(LP)}$	Data setup time to $\overline{WR}$ high	30			ns
$t_{h(LP)}$	Data hold time after $\overline{WR}$ high	25			ns
$t_{w(LP)}$	$\overline{WR}$ low pulse duration	60			ns

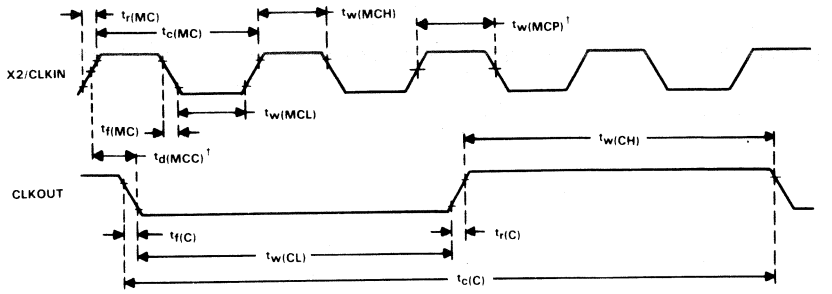
ADVANCE INFORMATION

TIMING DIAGRAMS

This section contains all the timing diagrams for the TMS320 first-generation devices. Refer to the top corner for the specific device.

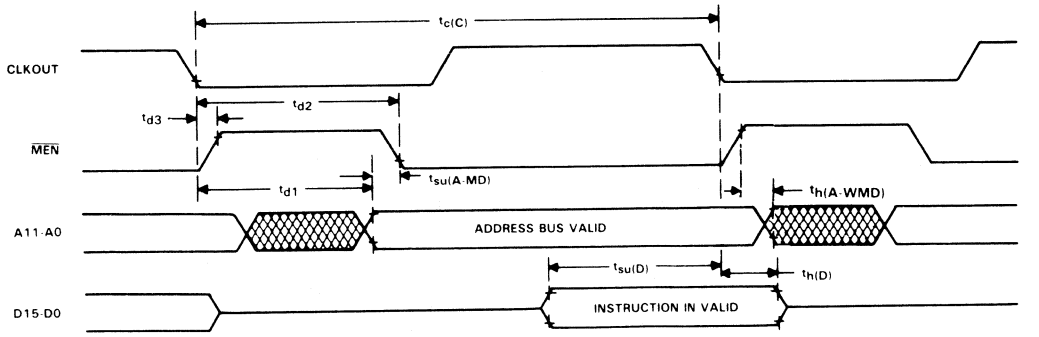
Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

clock timing

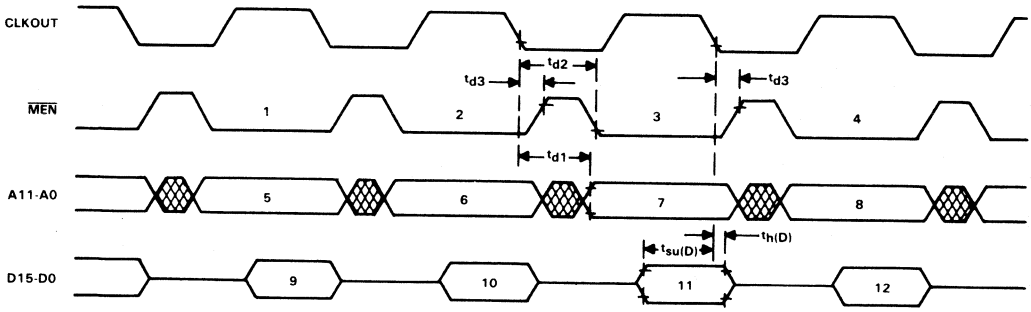


†$t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5 volts on the CLKIN waveform.

memory read timing



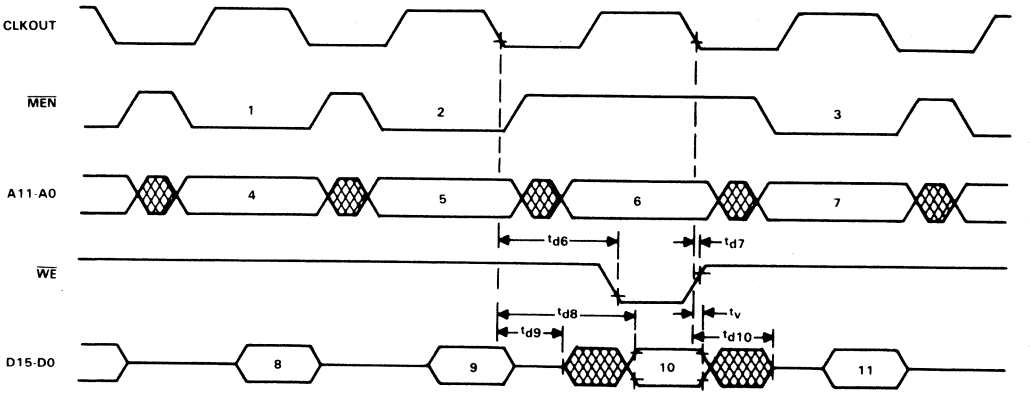
TBLR instruction timing



LEGEND:

1. TBLR INSTRUCTION PREFETCH
2. DUMMY PREFETCH
3. DATA FETCH
4. NEXT INSTRUCTION PREFETCH
5. ADDRESS BUS VALID
6. ADDRESS BUS VALID
7. ADDRESS BUS VALID
8. ADDRESS BUS VALID
9. INSTRUCTION IN VALID
10. INSTRUCTION IN VALID
11. DATA IN VALID
12. INSTRUCTION IN VALID

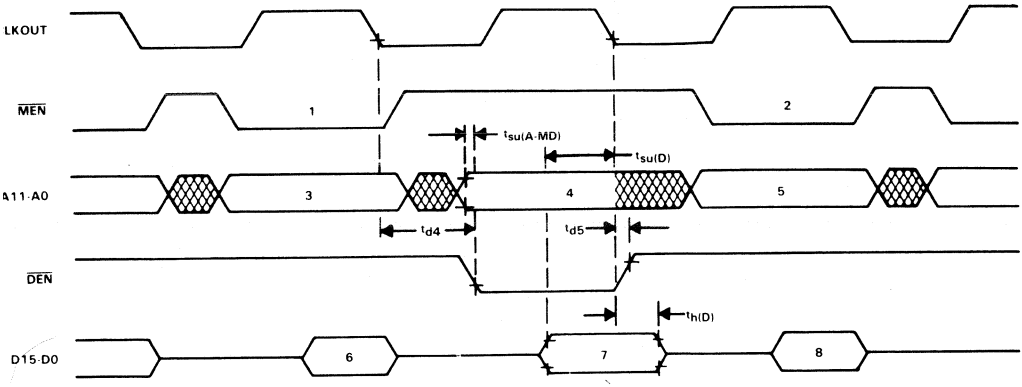
TBLW instruction timing



LEGEND:

1. TBLW INSTRUCTION PREFETCH
2. DUMMY PREFETCH
3. NEXT INSTRUCTION PREFETCH
4. ADDRESS BUS VALID
5. ADDRESS BUS VALID
6. ADDRESS BUS VALID
7. ADDRESS BUS VALID
8. INSTRUCTION IN VALID
9. INSTRUCTION IN VALID
10. DATA OUT VALID
11. INSTRUCTION IN VALID

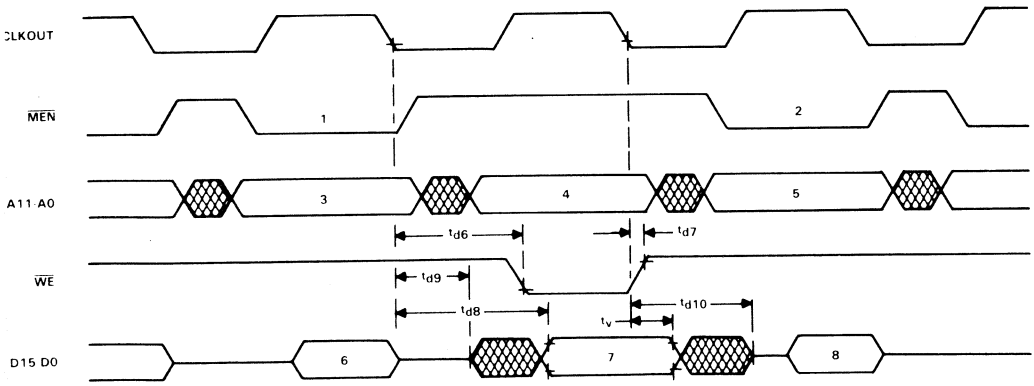
IN instruction timing



EGEND:

1. IN INSTRUCTION PREFETCH
2. NEXT INSTRUCTION PREFETCH
3. ADDRESS BUS VALID
4. PERIPHERAL ADDRESS VALID
5. ADDRESS BUS VALID
6. INSTRUCTION IN VALID
7. DATA IN VALID
8. INSTRUCTION IN VALID

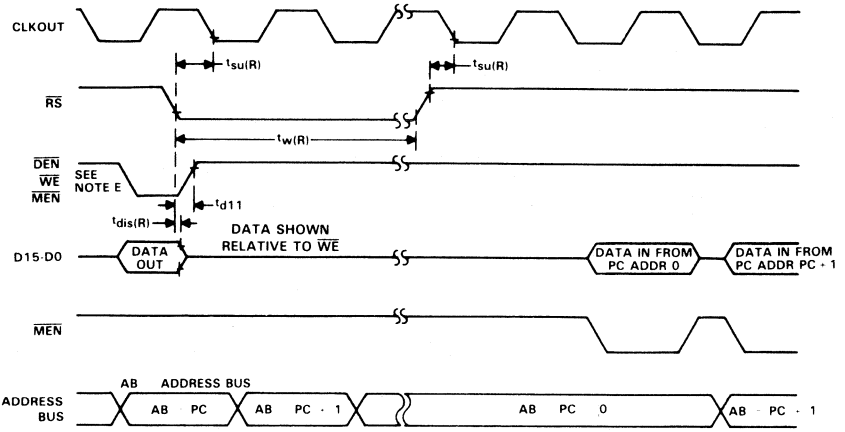
OUT instruction timing



EGEND:

1. OUT INSTRUCTION PREFETCH
2. NEXT INSTRUCTION PREFETCH
3. ADDRESS BUS VALID
4. PERIPHERAL ADDRESS VALID
5. ADDRESS BUS VALID
6. INSTRUCTION IN VALID
7. DATA IN VALID
8. INSTRUCTION IN VALID

reset timing



NOTES: A. $\overline{RS}$ forces $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ high and places data bus D0 through D15 in a high-impedance state. AB outputs (and program counter) are synchronously cleared to zero after the next complete CLK cycle from $\overline{RS}\downarrow$.

B. $\overline{RS}$ must be maintained for a minimum of five clock cycles.

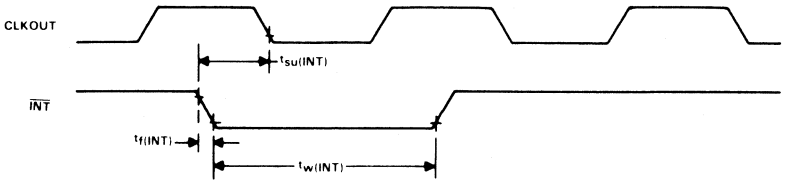
C. Resumption of normal program will commence after one complete CLK cycle from $\overline{RS}\uparrow$.

D. Due to the synchronizing action on $\overline{RS}$, time to execute the function can vary dependent upon when $\overline{RS}\uparrow$ or $\overline{RS}\downarrow$ occur in the CLK cycle.

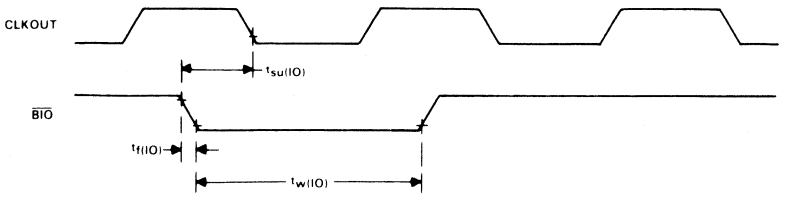
E. Diagram shown is for definition purpose only. $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ are mutually exclusive.

F. During a write cycle, $\overline{RS}$ may produce an invalid write address.

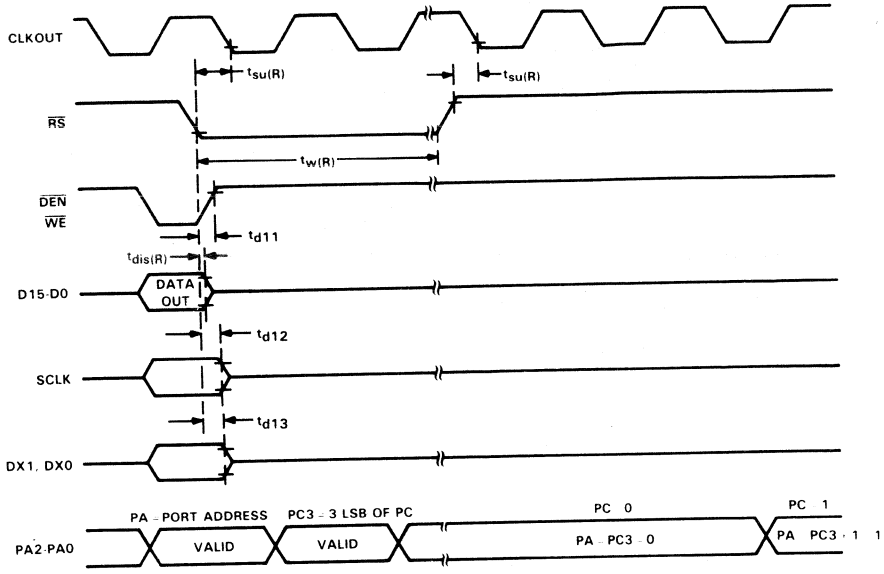
interrupt timing



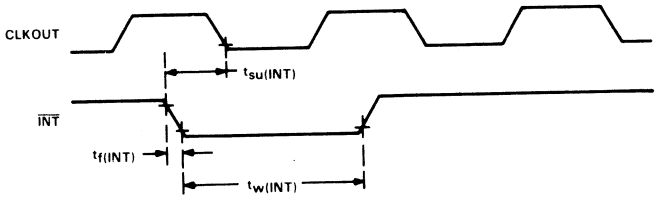
$\overline{BIO}$ timing



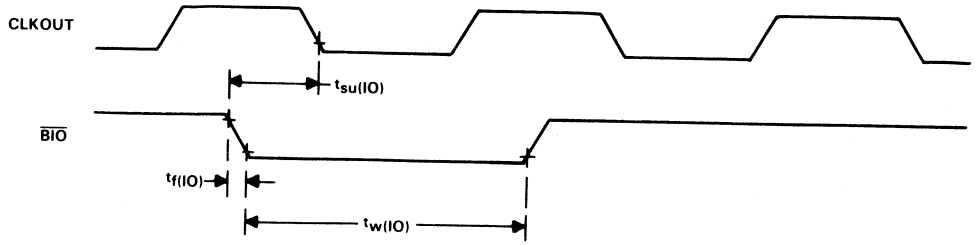
eset timing



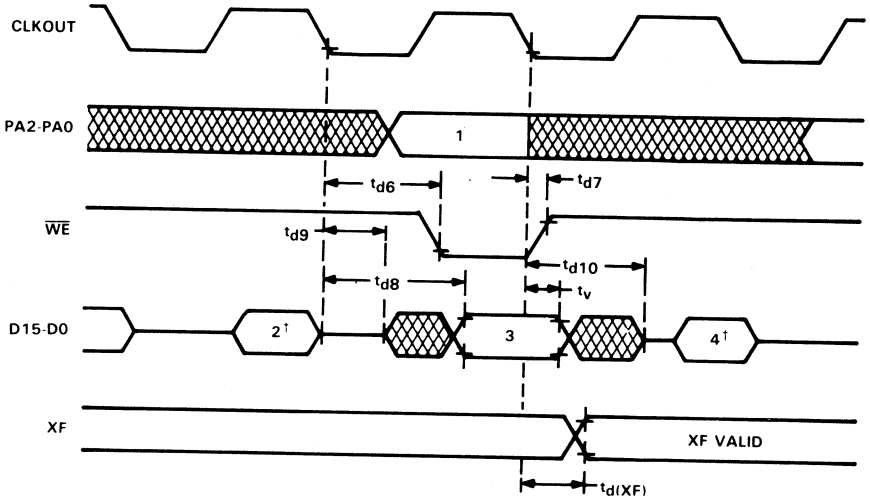
interrupt timing



$\overline{\text{BIO}}$ timing



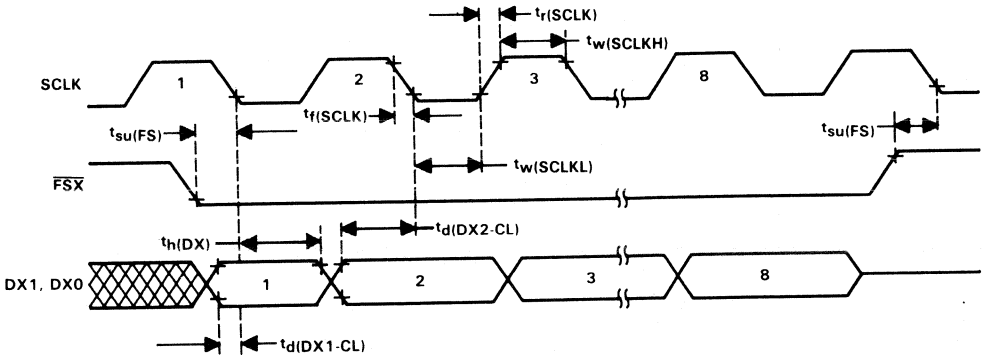
XF timing



LEGEND:

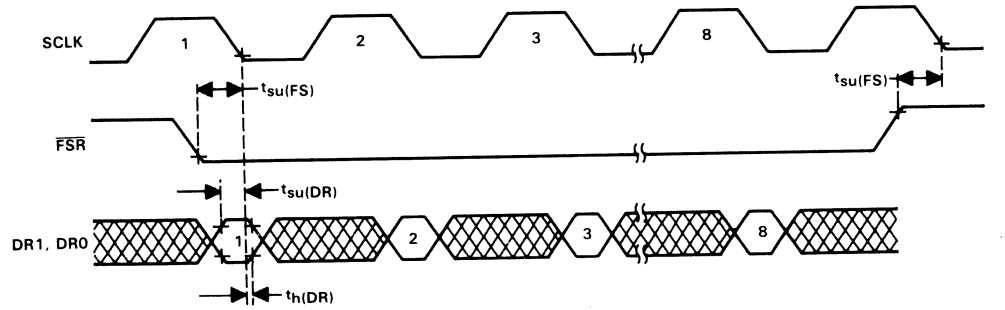
1. PORT ADDRESS VALID
2. OUT OPCODE VALID
3. PORT DATA VALID
4. NEXT INSTRUCTION OPCODE VALID

external framing: transmit timing



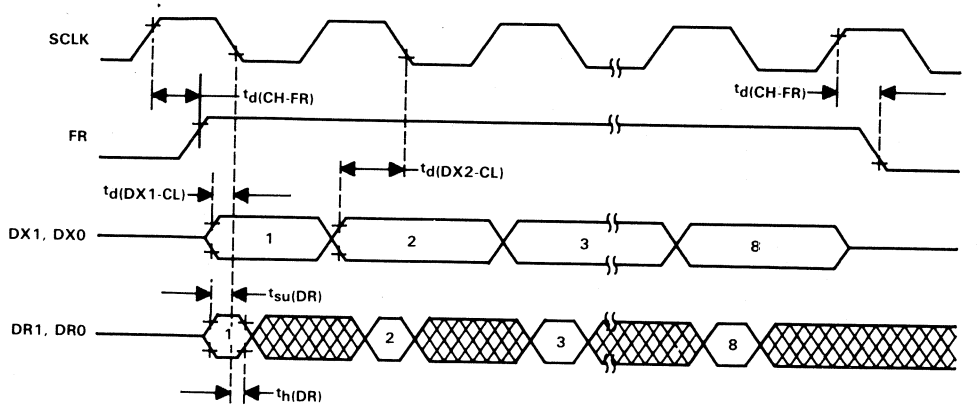
NOTES: G. Data valid on transmit outputs until SCLK rises.
H. The most significant bit is shifted first.

external framing: receive timing



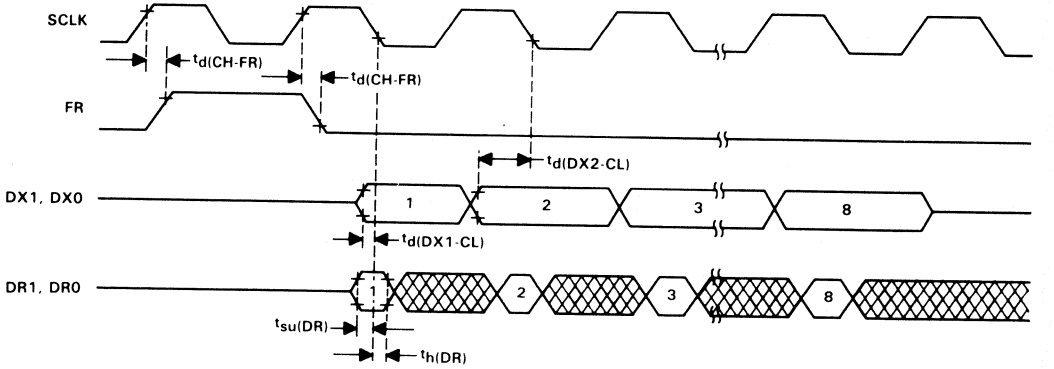
NOTE H: The most significant bit is shifted first.

internal framing: variable-data rate



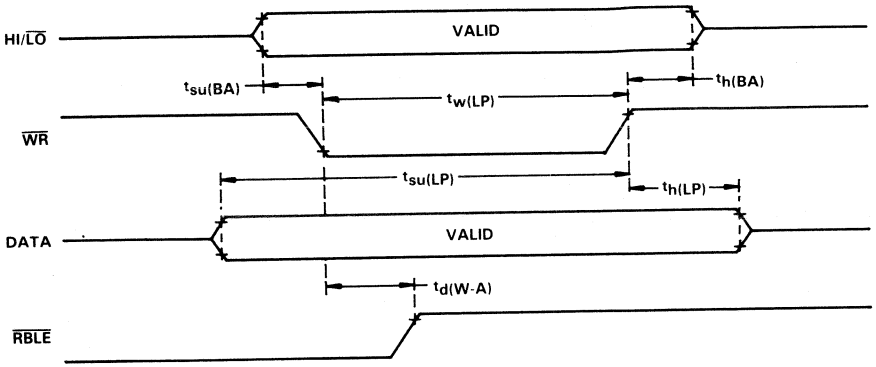
NOTE H: The most significant bit is shifted first.

internal framing: fixed-data rate

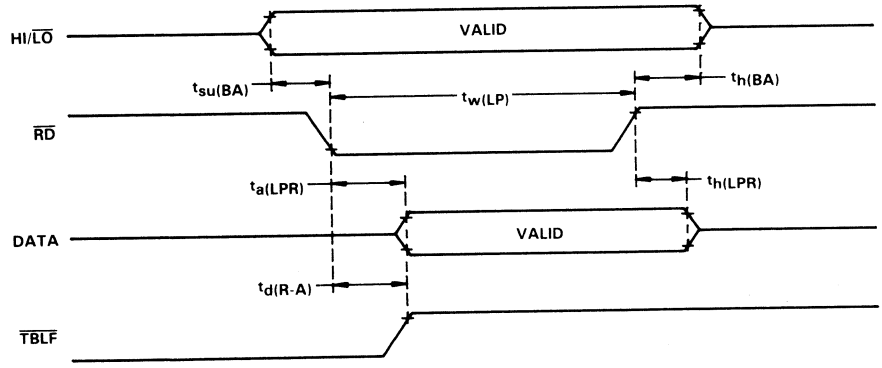


NOTE H: The most significant bit is shifted first.

:oprocessor timing: external write to coprocessor port



coprocessor timing: external read from coprocessor port



EPROM PROGRAMMING

absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{PP} (see Note 1) . . . . . −0.6 V to 14 V

†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.

NOTE 1. All voltage values are with respect to GND.

recommended operating conditions

	MIN	NOM	MAX	UNIT
V_{PP} Supply voltage (see Note 2)	0	12	12.5	V

NOTE 2: V_{PP} can be connected to V_{CC} directly (except in the program mode). V_{CC} supply current in this case would be $I_{CC} + I_{PP}$. During programming, V_{PP} must be maintained at 12.5 V (±0.5 V).

electrical characteristics over specified temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_{PP1} V_{PP} supply current	$V_{PP} = V_{CC} = 5.25$ V			100	μA
I_{PP2} V_{PP} supply current (during program pulse)	$V_{PP} = 13$ V		30	50	mA

†All typical values except for I_{CC} are at $V_{CC} = 5$ V, $T_A = 25°C$.

recommended timing requirements for programming, $T_A = 25°C$, $V_{CC} = 6$ V, $V_{PP} = 12.5$ V (see Note 3)

		MIN	NOM	MAX	UNIT
$t_{w(IPGM)}$	Initial program pulse duration	0.95	1	1.05	ms
$t_{w(FPGM)}$	Final pulse duration	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
$t_{dis(G)}$	Output disable time from $\overline{G}$	0		130	ns
$t_{en(G)}$	Output enable time from $\overline{G}$			150	ns
$t_{su(D)}$	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

NOTES 3. 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.

4. Common test conditions apply for $t_{dis(G)}$ except during programming.

PROGRAMMING THE TMS320E15/E17 EPROM CELL

The TMS320E15/E17 includes a 4K x 16-bit industry-standard EPROM cell for prototyping, early field testing, and low-volume production. The TMS320C15/C17 with a 4K-word masked ROM then provides a migration path for cost-effective production. An EPROM programmer adaptor socket (part #RTC/PGM320A-06), shown in Figure 7, is available to provide 40-pin to 28-pin conversion for programming the TMS320E15/E17.

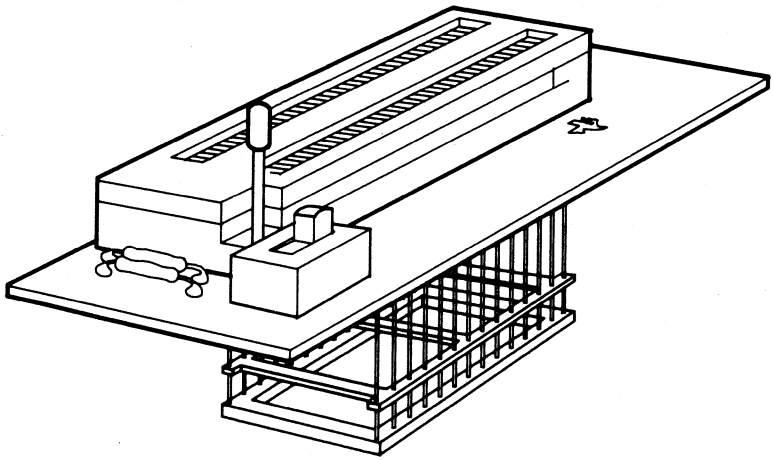
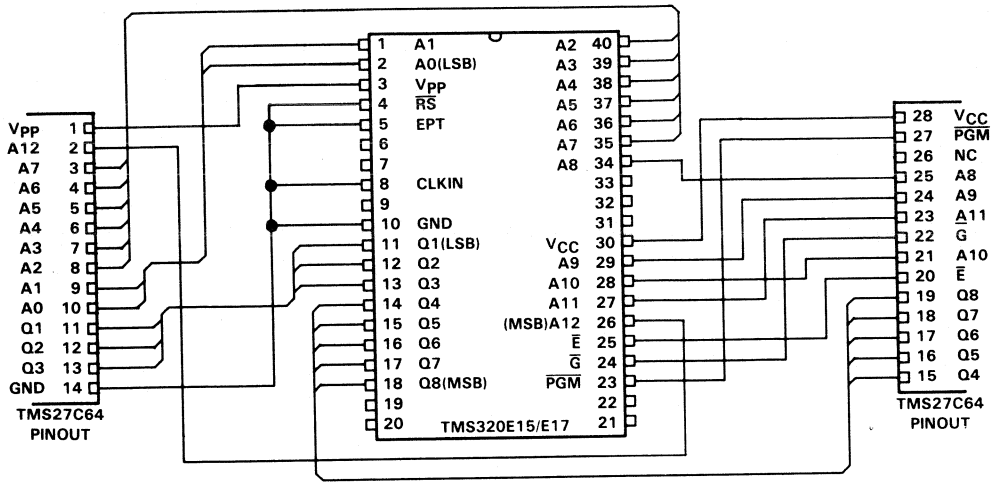


FIGURE 7. EPROM PROGRAMMER ADAPTOR SOCKET

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 TMS320E15/E17 EPROM cell is programmed using the same family and device pinout 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. 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, 12.5-V V_{PP} and 6-V V_{CC} supplies are 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 8 shows the wiring conversion to program the TMS320E15/E17 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. The TMS320E15/E17 uses 13 address lines to address the 4K-word memory in byte format.



PIN NOMENCLATURE (TMS320E15/TMS320E17)

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 select
EPT	I	EPROM test mode select
$\overline{G}$	I	EPROM read/verify select
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 power supply
V_{PP}	I	12.5-V power supply

FIGURE 8. TMS320E15/E17 EPROM PROGRAMMING CONVERSION TO TMS27C64 EPROM PINOUT

Table 8 shows the programming levels required for programming, verifying, reading, and protecting the EPROM cell.

TABLE 8. TMS320E15/E17 PROGRAMMING MODE LEVELS

SIGNAL NAME	TMS320E15/E17 PIN	TMS27C64 PIN	PROGRAM	VERIFY	READ	PROTECT VERIFY	ROM PROTECT
$\overline{E}$	25	20	V_{IL}	V_{IL}	V_{IL}	V_{IL}	V_{IH}
$\overline{G}$	24	22	V_{IH}	$\overline{PULSE}$	$\overline{PULSE}$	V_{IL}	V_{IH}
$\overline{PGM}$	23	27	$\overline{PULSE}$	V_{IH}	V_{IH}	V_{IH}	V_{IH}
V_{PP}	3	1	V_{PP}	V_{PP}	V_{CC}	V_{CC}	V_{PP}
V_{CC}	30	28	$V_{CC}+1$	$V_{CC}+1$	V_{CC}	V_{CC}	$V_{CC}+1$
V_{SS}	10	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
CLKIN	8	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
$\overline{RS}$	4	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
EPT	5	26	V_{SS}	V_{SS}	V_{SS}	V_{PP}	V_{PP}
Q8-Q1	18-11	19-15, 13-11	D_{IN}	Q_{OUT}	Q_{OUT}	Q8 = RBIT	X
A12-A10	26-28	2,23,21	ADDR	ADDR	ADDR	X	X
A9-A7	29,34,35	24,25,3	ADDR	ADDR	ADDR	X	X
A6	36	4	ADDR	ADDR	ADDR	V_{IL}	X
A5	37	5	ADDR	ADDR	ADDR	X	X
A4	38	6	ADDR	ADDR	ADDR	X	V_{IH}
A3-A0	39,40,1,2	7-10	ADDR	ADDR	ADDR	X	X

LEGEND:
V_{IH} = TTL high level; V_{IL} = TTL low level; ADDR = byte address bit
V_{PP} = 12.5 V ±0.5 V; V_{CC} = 5 V ±5% for read only, otherwise, V_{CC} = 6 V; X = don't care
$\overline{PULSE}$ = low-going TTL level pulse; D_{IN} = byte to be programmed at ADDR
Q_{OUT} = byte stored at ADDR; RBIT = ROM protect bit.

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 V_{PP} 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 8, assuming the inhibit bit has not been programmed.

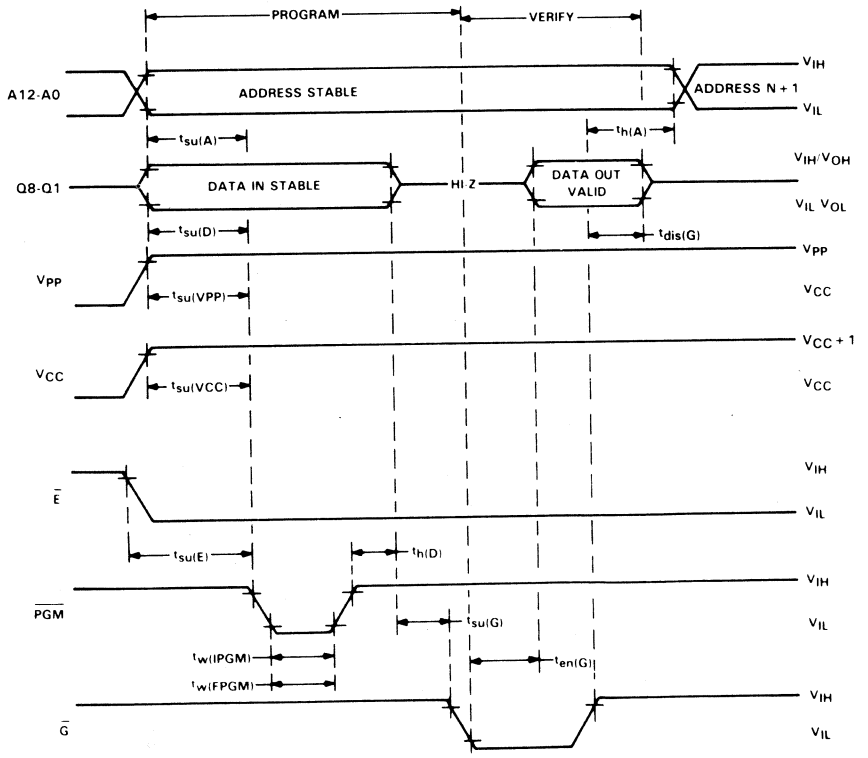
program inhibit

To protect the proprietary algorithms existing in the code programmed on-chip, the ability to read or verify code from external accesses can be completely disabled. Programming the RBIT disables external access of the EPROM cell and disables the microprocessor mode, making it impossible to access the code resident in the EPROM cell. The only way to remove this protection is to erase the entire EPROM cell, thus removing the proprietary information. The signal requirements for programming this bit are shown in Table 8. The cell can be determined as protected by verifying the programming of the RBIT shown in the table.

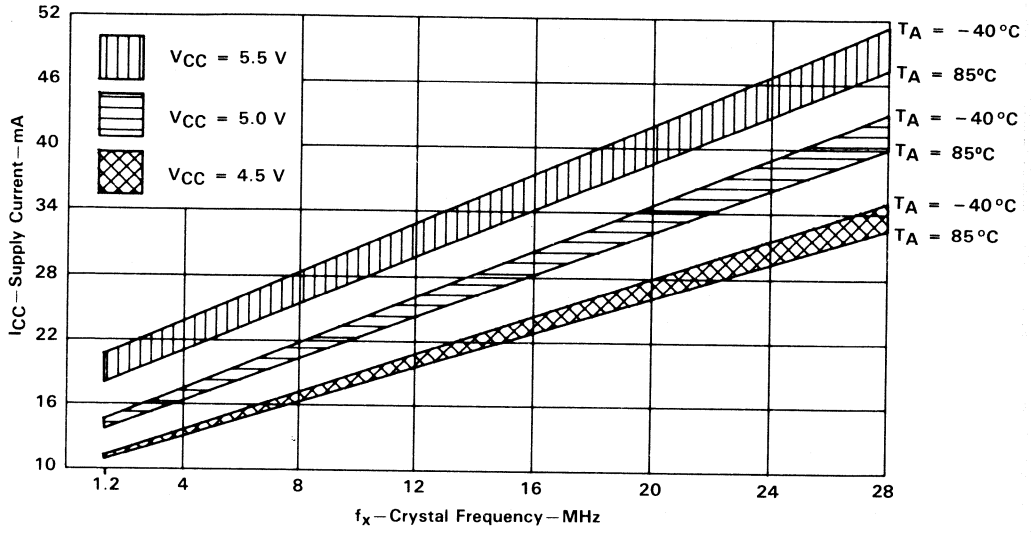
standard programming procedure

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.

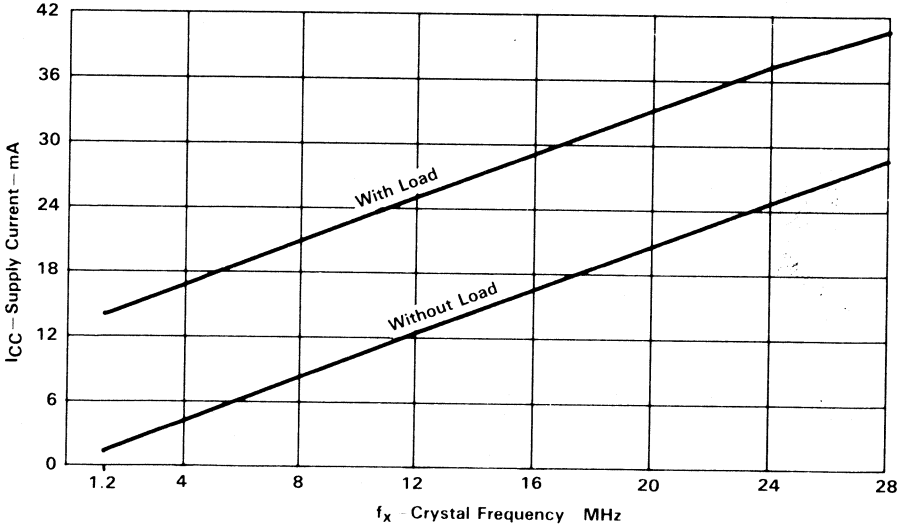
program cycle timing



TYPICAL POWER VS. FREQUENCY GRAPHS



(a) – 40°C TO 85°C TEMPERATURE RANGE



(b) VOLTAGE = 5 V; TEMPERATURE = 25°C

FIGURE 9. TYPICAL CMOS I_{CC} VS. FREQUENCY

PACKAGE TYPES

PACKAGE TYPE	SUFFIX	FAMILY MEMBERS
40-pin plastic DIP (100-mil pin spacing)	N	NMOS: TMS32010, TMS32010-25, TMS32010-14, TMS32011 CMOS: TMS320C10, TMS320C10-25, TMS320C15†, TMS320C15-25†, TMS320C17†, TMS320C17-25†
44-lead PLCC (50-mil pin spacing)	FN	CMOS: TMS320C10, TMS320C10-25†, TMS320C15†, TMS320C15-25†, TMS320C17†, TMS320C17-25†
40-pin windowed ceramic DIP	JD	CMOS: TMS320E15, TMS320E17

†Planned versions; contact TI representative for availability.

THERMAL DATA

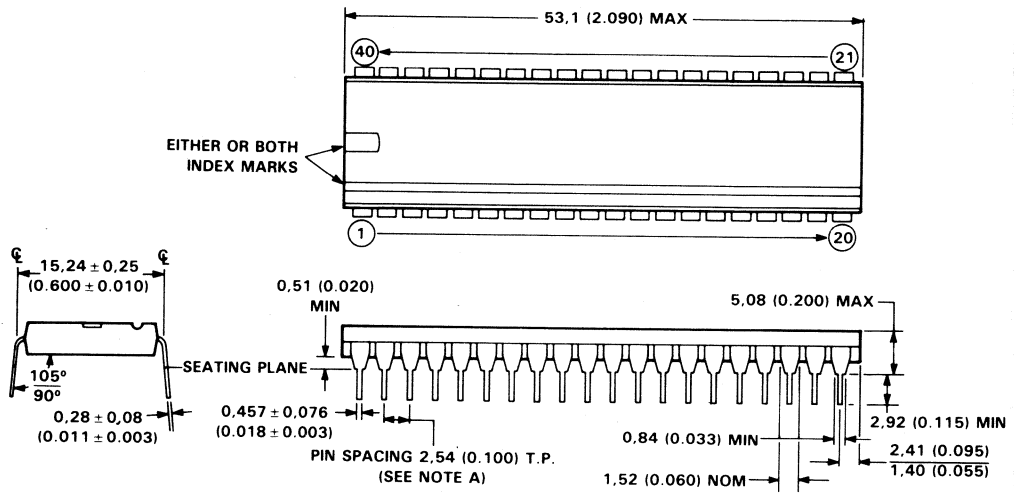
thermal resistance characteristics

PACKAGE	$R_{\theta JA}$ (°C/W)	$R_{\theta JC}$ (°C/W)
40-pin plastic dual-in-line package (NMOS†)	51.6	16.6
40-pin plastic dual-in-line package (CMOS)	84	26
40-pin windowed ceramic dual-in-line package (CMOS)	41	8
44-lead plastic chip carrier package (CMOS)	60	17

†A more thermally efficient package has been used to compensate for higher power dissipation of the NMOS part.

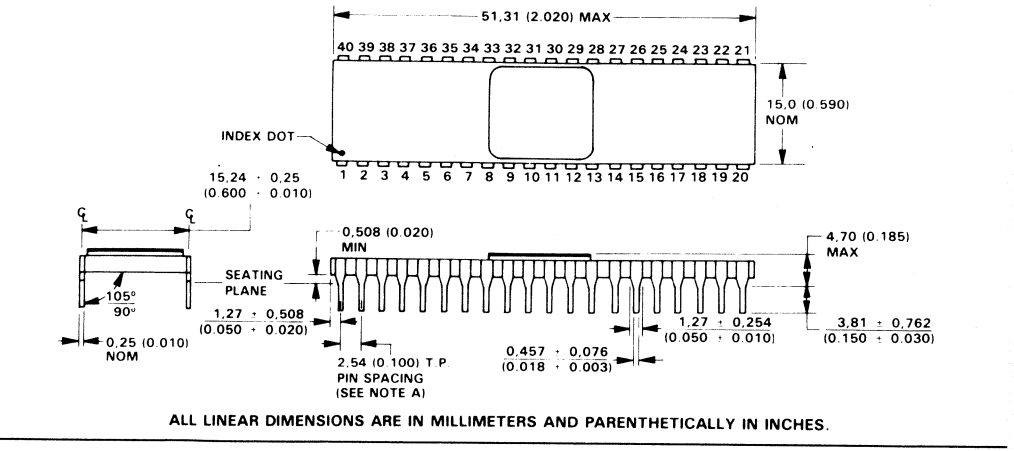
MECHANICAL DATA

40-pin plastic dual-in-line package



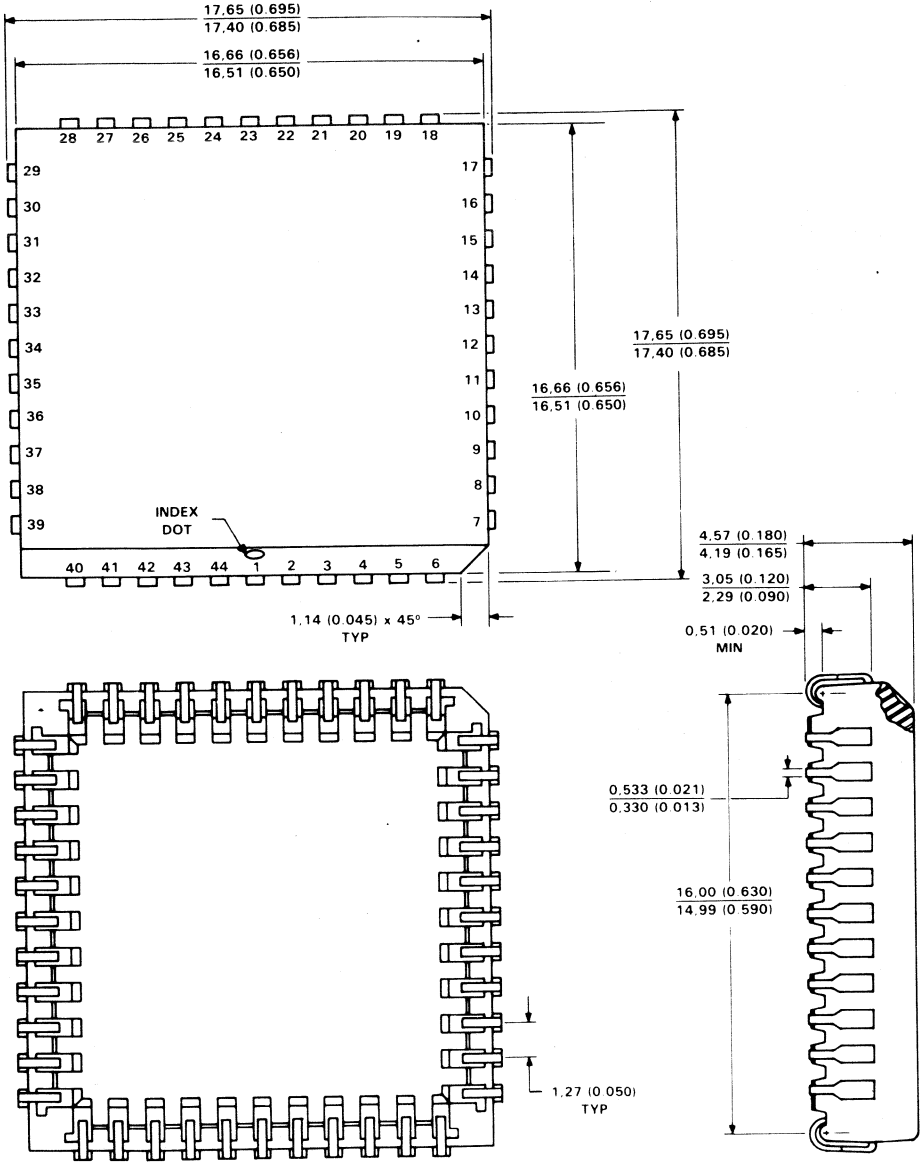
NOTE A: Each pin centerline is located within 0,254 (0.010) of its true longitudinal position.

40-pin windowed ceramic dual-in-line package (TMS320E15/E17)



NOTE A: Each pin centerline is located within 0,254 (0.010) of its true longitudinal position.

44-lead plastic chip carrier package



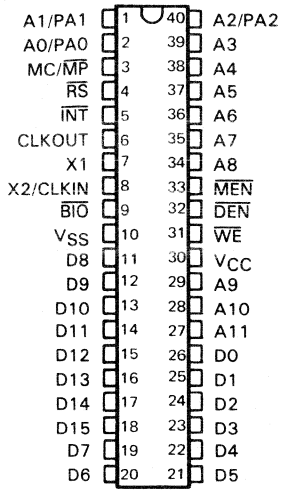
ALL DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

INDEX

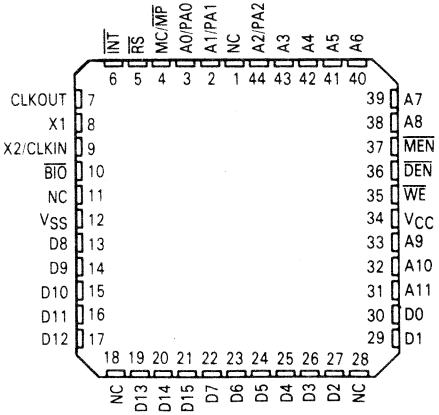
MAY 1983 – REVISED MAY 1987

- High Reliability Class B Processing
- 200-ns Instruction Cycle
- 144-Word On-Chip Data RAM
- Currently Microprocessor Mode Only (All Program Memory is Extended)
- External Memory Expansion to Total of 4K Words at Full Speed
- 16-Bit Instruction/Data Word
- 32-Bit ALU/Accumulator
- 16 × 16-Bit Multiply in One Instruction Cycle
- 0 to 16-Bit Barrel Shifter
- Eight Input and Eight Output Channels
- 16-Bit Bidirectional Data Bus with 40-Megabits-per-Second Transfer Rate
- Interrupt with Full Context Save
- Signed Two's-Complement Fixed-Point Arithmetic
- 2.4-Micron NMOS Technology
- Single 5-V Supply [±10% for (−55°C to 100°C) Temperature Range (S Suffix)]

SMJ32010 . . . JD PACKAGE
(TOP VIEW)



SMJ32010 . . . FD PACKAGE
(TOP VIEW)



description

The SMJ32010 is a member of the TMS320 digital signal processing family, designed to support a wide range of high-speed or numeric-intensive applications. This 16/32-bit single-chip microcomputer combines the flexibility of a high-speed controller with the numerical capability of an array processor, thereby offering an inexpensive alternative to multichip bit-slice processors. The TMS320 family contains the first MOS microcomputers capable of executing five million instructions per second. This high throughput is the result of the comprehensive, efficient, and easily programmed instruction set and of the highly pipelined architecture. Special instructions have been incorporated to speed the execution of digital signal processing (DSP) algorithms.

The TMS320 family's unique versatility and power give the design engineer a new approach to a variety of complicated applications. In addition, these microcomputers are capable of providing the multiple functions often required

for a single application. For example, the TMS320 family can enable an industrial robot to synthesize and recognize speech, sense objects with radar or optical intelligence, and perform mechanical operations through digital servo loop computations.

PIN NOMENCLATURE

SIGNATURE	I/O/Z†	DEFINITION
A11-A0/ PA2-PA0	O	External address bus. I/O port address multiplexed over PA2-PA0.
$\overline{BIO}$	I	External polling input.
CLKOUT	O	System clock output, ¼ crystal/CLKIN frequency.
D15-D0	I/O/Z	16-bit data bus.
$\overline{DEN}$	O	Data enable indicates the processor accepting input data on D15-D0.
$\overline{INT}$	I	Interrupt.
MC/$\overline{MP}$	I	Memory mode select pin. High selects microcomputer mode. Low selects microprocessor mode.
$\overline{MEN}$	O	Memory enable indicates that D15-D0 will accept external memory instruction.
$\overline{RS}$	I	Reset used to initialize the device.
V_{CC}	I	Power.
V_{SS}	I	Ground.
$\overline{WE}$	O	Write-enable indicates valid data on D15-D0.
X1	O	Crystal output.
X2/CLKIN	I	Crystal input or external clock input.

†Input/Output/High-impedance state

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 the 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 separated coefficient ROM. It also makes available immediate instructions and subroutines based on computed values.

The SMJ32010 utilizes hardware to implement functions that other processors typically perform in software. For example, this device contains a hardware multiplier to perform a multiplication in a single 200-ns cycle. There is also a hardware barrel shifter for shifting data on its way into the ALU. Extra hardware has been included so that auxiliary registers, which provide indirect data RAM addresses, can be configured in an autoincrement/decrement mode for single-cycle manipulation of data tables. This hardware-intensive approach gives the design engineer the type of power previously unavailable on a single chip.

32-bit ALU/accumulator

The SMJ32010 contains a 32-bit ALU and accumulator that support double-precision arithmetic. The ALU operates on 16-bit words taken from the data RAM or derived from immediate instructions. Besides the usual arithmetic instructions, the ALU can perform Boolean operations, providing the bit manipulation ability required of a high-speed controller.

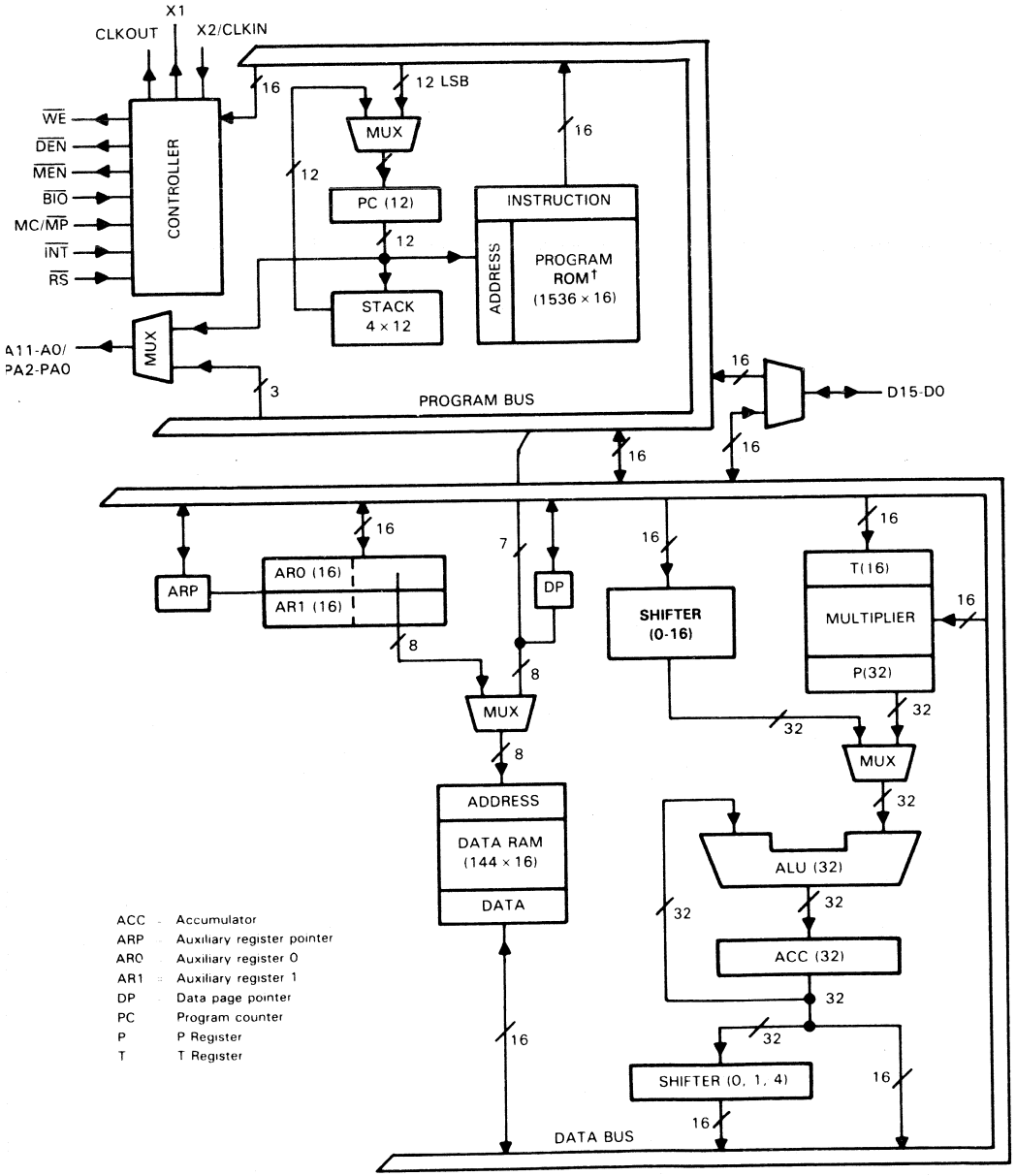
shifters

A barrel shifter is available for left-shifting data 0 to 16 places before it is loaded into, subtracted from, or added to the accumulator. This shifter extends the high-order bit of the data word and zero-fills the low-order bits for two's complement arithmetic. A second shifter left-shifts the upper half of the accumulator 0, 1, or 4 places while it is being stored in the data RAM. Both shifters are useful for scaling and bit extraction.

16 × 16-bit parallel multiplier

The SMJ32010's multiplier performs a 16 × 16-bit, two's complement multiplication in one 200-ns instruction cycle. The 16-bit T Register temporarily stores the multiplicand; the P Register stores the 32-bit result. 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 SMJ32010 to perform such fundamental operations as convolution, correlation, and filtering at the rate of 2.5 million samples per second.

functional block diagram



† The Program ROM is available in SMJ320M10 only.

input/output

The SMJ32010's 16-bit parallel data bus can be utilized to perform I/O functions at burst rates of 40 million bits per second. Available for interfacing to peripheral devices are 128 input and 128 output bits consisting of eight 16-bit multiplexed input ports and eight 16-bit multiplexed output ports. In addition, a polling input for bit test and jump operations (BIO) and an interrupt pin (INT) have been incorporated for multi-tasking.

interrupts and subroutines

The SMJ32010 contains a four-level hardware stack for saving the contents of the program counter during interrupts and subroutine calls. Instructions are available for saving the SMJ32010's complete context. The instructions, PUSH stack from accumulator, and POP stack to accumulator, permit a level of nesting restricted only by the amount of available RAM. The interrupts used in the SMJ32010 are maskable.

instruction set

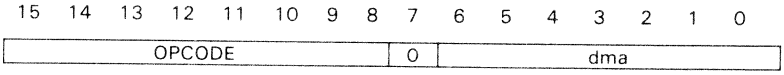
The SMJ32010's comprehensive instruction set supports both numeric-intensive operations, such as signal processing, and general purpose operations, such as high-speed control. The instruction set, explained in Tables 1 and 2, consists primarily of single-cycle single-word instructions, permitting execution rates of up to five million instructions per second. Only frequently used branch and I/O instructions are multicycle.

The SMJ32010 also contains a number of instructions that shift data a part of an arithmetic operation. These all execute in a single cycle and are very useful for scaling data in parallel with other operations.

Three main addressing modes are available with the SMJ32010 instruction set: direct, indirect, and immediate addressing.

direct addressing

In direct addressing, seven bits of the instruction word concatenated with the data page pointer form the data memory address. This implements a paging scheme in which the first page contains 128 words and the second page contains 16 words. In a typical application, infrequently accessed variables, such as those used for servicing an interrupt, are stored on the second page. The instruction format for direct addressing is shown below.



Bit 7 = 0 defines direct addressing mode. The opcode is contained in bits 15 through 8. Bits 6 through 0 contain data memory address.

The seven bits of the data memory address (dma) field 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 144 words of data memory.

Direct addressing can be used with all instructions requiring data operands except for the immediate operand instructions.

indirect addressing

Indirect addressing forms the data memory address from the least significant eight bits of one of 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 in parallel with the execution of any indirect instruction to permit single-cycle manipulation of data tables. The instruction format for indirect addressing 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

Bit 7 = defines indirect addressing mode. The opcode is contained in bits 15 through 8. Bits 6 through 0 contain indirect addressing contol 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 indicates 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.

Indirect addressing can be used with all instructions requiring data operands, except for the immediate operand instructions.

immediate addressing

The SMJ32010 instruction set contains special "immediate" instructions. These instructions derive data from part of the instruction word rather than from the data RAM. Some very 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 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 0 I ← D →
ADDS	Add to accumulator with no sign extension	1	1	0 1 1 0 0 0 0 0 1 I ← D →
AND	AND with accumulator	1	1	0 1 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 1 0 ← K →
OR	OR with accumulator	1	1	0 1 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 0 I ← D →
SUB	Subtract from accumulator with shift	1	1	0 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 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 —►
BIOZ	Branch on $\overline{BIO}$ = 0	2	2	1 1 1 1 0 1 1 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 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 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 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 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	3	1	0 1 1 1 1 1 0 1 I ◄—— D ——►

development support

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 development support products for the first-generation TMS320 devices.

System development begins with the use of the Evaluation Module (EVM) or Emulator (XDS). These hardware tools allow 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 simulator, XDS, or EVM. 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* for further information about TMS320 development support products and DSP workshops. When technical questions arise regarding the TMS320, contact the Texas Instruments Regional Technology Centers (see last pages).

TABLE 3. TMS320 FIRST-GENERATION SOFTWARE AND HARDWARE SUPPORT

SOFTWARE TOOLS	PART NUMBER
Macro Assembler/Linker	
VAX VMS	TMDS3240210-08
TI/IBM MS/PC-DOS	TMDS3240810-02
Simulator	
VAX VMS	TMDS3240211-08
TI/IBM MS/PC-DOS	TMDS3240811-02
Digital Filter Design Package (DFDP)	
TI PC MS-DOS	DFDP-TI001
IBM PC PC-DOS	DFDP-IBM001
DSP Software Library	
VAX VMS	TMDC3240212-18
TI/IBM MS/PC-DOS	TMDC3240812-12
HARDWARE TOOLS	**PART NUMBER**
Evaluation Module (EVM)	RTC/EVM320A-03
Analog Interface Board (AIB)	RTC/EVM320C-06
XDS/22 Emulator	TMDS3262211
XDS/22 Upgrade	
Factory Upgrade	TMDS3282215
Customer Upgrade	TMDS3282216
EPROM Programmer Adaptor Socket	RTC/PGM320A-06
TMS320 Design Kit	TMS320DDK

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.*

A series of DSP textbooks is being published by Prentice-Hall and John Wiley & Sons to support digital signal processing research and education. 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 Regional Technology Centers (see last pages).

absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{CC}‡ . . . . . −0.3 V to 7 V
Input voltage range . . . . . −0.3 V to 15 V
Output voltage range . . . . . −0.3 V to 15 V
Continuous power dissipation . . . . . 1.5 W
Maximum operating case temperature . . . . . 100°C
Minimum operating free-air temperature . . . . . −55°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		4.5	5	5.5	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	All inputs except CLKIN	2			V
		CLKIN	2.8			
V_{IL}	Low-level input voltage	X2/CLKIN and data			0.8	V
		$\overline{BIO}$, $\overline{INT}$, MC/$\overline{MP}$, $\overline{RS}$			0.7	
I_{OH}	High-level output current (all outputs)				300	μA
I_{OL}	Low-level output current (all outputs)				2	mA
T_C	Maximum operating case temperature				100	°C
T_A	Minimum free-air temperature		−55			°C

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
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}				±50	μA
I_{CC}	Supply current		V_{CC} = MAX,	f_x = MAX,		180	275	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		
		All others				10		

†All typical values are at V_{CC} = 5 V, T_A = 25°C.

CLOCK CHARACTERISTICS AND TIMING

The SMJ32010 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency.

PARAMETER	TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency f_x	−55°C to 100°C	6.7		20	MHz
C1, C2			10		pF

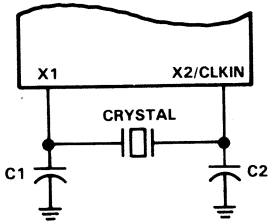


FIGURE 1. INTERNAL CLOCK OPTION

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the following table.

timing requirements over recommended operating conditions

PARAMETER		MIN	NOM	MAX	UNIT
$t_{c(MC)}$	Master clock cycle time	50		150	ns
$t_{r(MC)}$	Rise time master clock input		5	10	ns
$t_{f(MC)}$	Fall time master clock input		5	10	ns
$t_{w(MCL)}$	Pulse duration master clock low, $t_{c(MC)}$ = 50 ns		20		ns
$t_{w(MCH)}$	Pulse duration master clock high, $t_{c(MC)}$ = 50 ns		20		ns

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLKOUT cycle time	R_L = 825 Ω, C_L = 100 pF, See Figure 2	200			ns
$t_{r(C)}$	CLKOUT rise time			10		ns
$t_{f(C)}$	CLKOUT fall time			8		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90		ns

PARAMETER MEASUREMENT INFORMATION

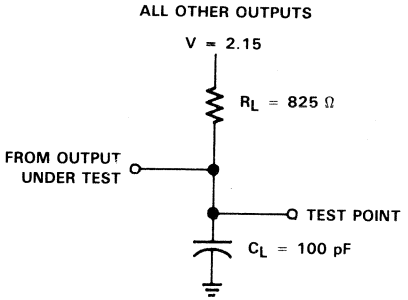
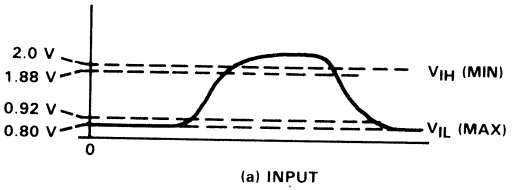
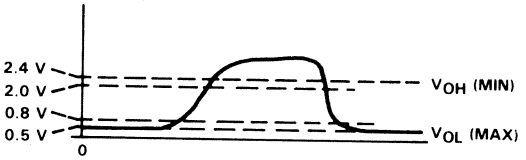


FIGURE 2. TEST LOAD CIRCUIT



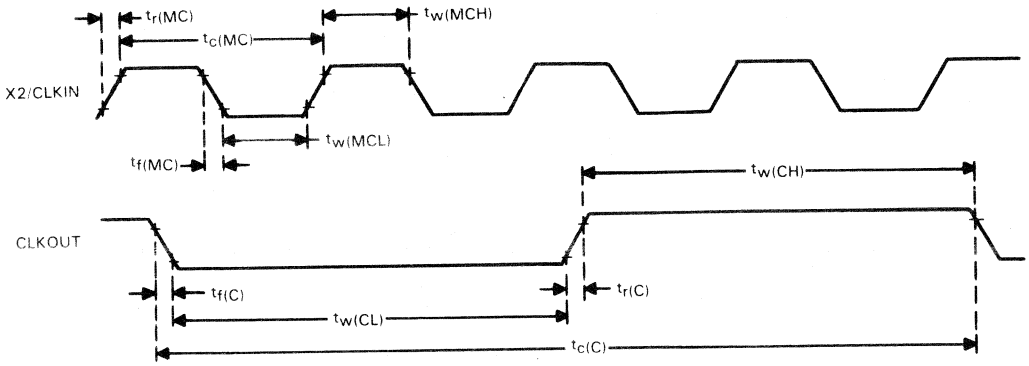
(a) INPUT



(b) OUTPUTS

FIGURE 3. VOLTAGE REFERENCE LEVELS

clock timing



NOTE 1: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

	PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d1}	Delay time CLKOUT↓ to address bus valid (see Note 2)	R_L = 825 Ω, C_L = 100 pF, See Figure 2	$10^{\dagger}$		60	ns
t_{d2}	Delay time CLKOUT↓ to $\overline{MEN}$↓		$¼t_{c(C)} - 10^{\dagger}$		$¼t_{c(C)} + 15$	ns
t_{d3}	Delay time CLKOUT↓ to $\overline{MEN}$↑		$-15^{\dagger}$		15	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{DEN}$↓		$¼t_{c(C)} - 10^{\dagger}$		$¼t_{c(C)} + 15$	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{DEN}$↑		$-15^{\dagger}$		15	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{WE}$↓		$½t_{c(C)} - 10^{\dagger}$		$½t_{c(C)} + 15$	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{WE}$↑		$-10^{\dagger}$		15	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid				$¼t_{c(C)} + 65$	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		$¼t_{c(C)} - 10^{\dagger}$			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				$¼t_{c(C)} + 30^{\dagger}$	ns
t_v	Data bus OUT valid after CLKOUT↓		$¼t_{c(C)} - 10$			ns

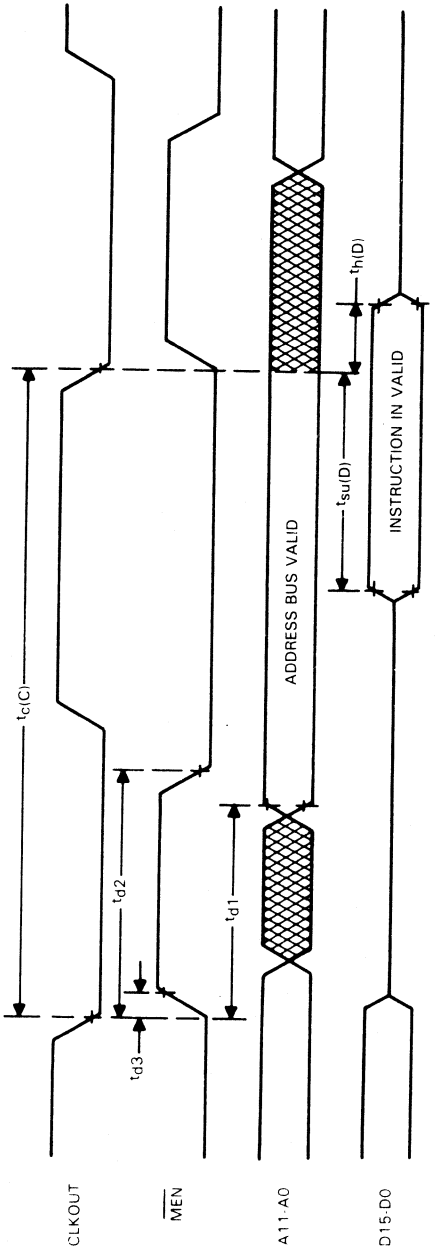
NOTE 2: Address bus will be valid upon $\overline{WE}$↑, $\overline{DEN}$↑, or $\overline{MEN}$↑, and address bus will be valid upon $\overline{MEN}$↓ or $\overline{DEN}$↓.
†These values were derived from characterization data and are not tested.

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 Ω, C_L = 100 pF, See Figure 2	50			ns
$t_{h(D)}$	Hold time data bus held valid after CLKOUT↓ (see Note 3)		0			ns

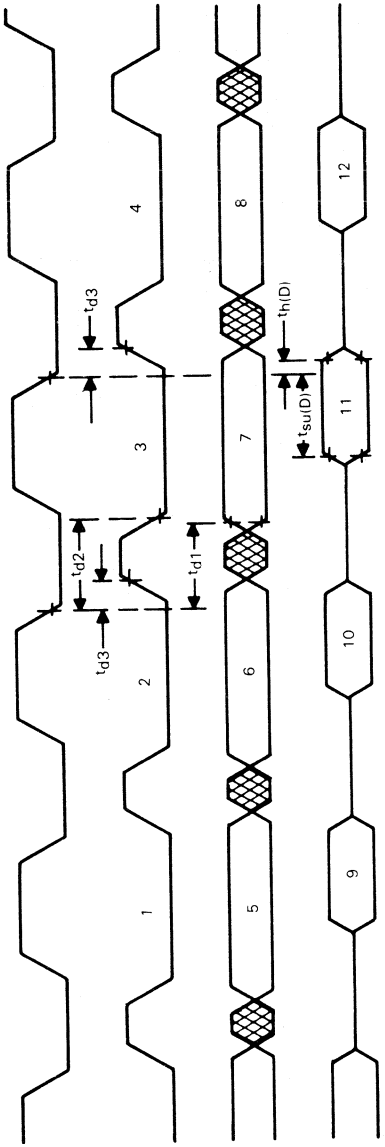
NOTE 3: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

memory read



NOTE 1: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

TBLR instruction timing

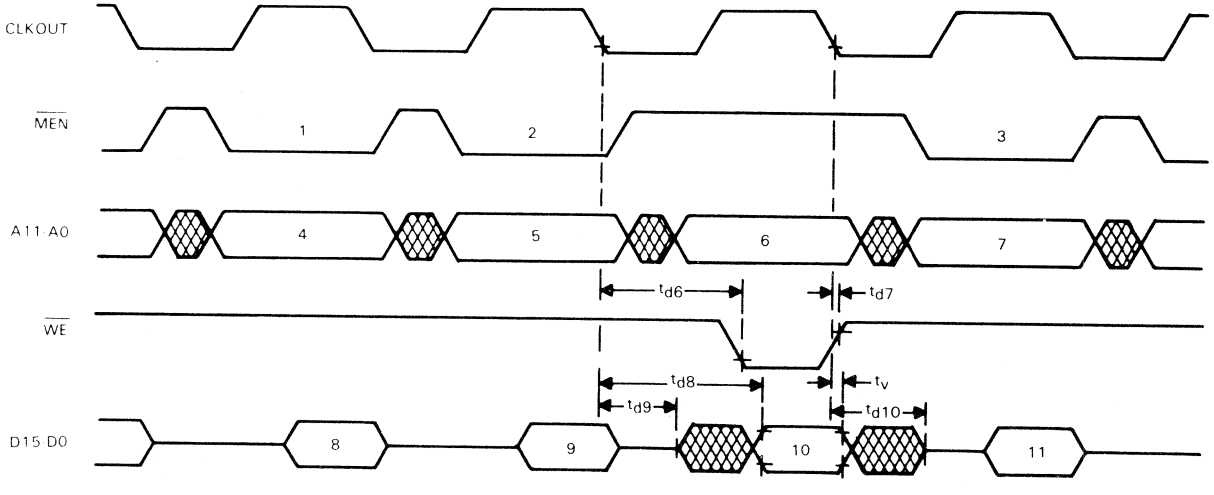


LEGEND:

1. TBLR INSTRUCTION PREFETCH
2. DUMMY PREFETCH
3. DATA FETCH
4. NEXT INSTRUCTION PREFETCH
5. ADDRESS BUS VALID
6. ADDRESS BUS VALID
7. ADDRESS BUS VALID
8. ADDRESS BUS VALID
9. INSTRUCTION IN VALID
10. INSTRUCTION IN VALID
11. DATA IN VALID
12. INSTRUCTION IN VALID

NOTE 1: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

TBLW instruction timing



LEGEND:

1. TBLW INSTRUCTION PREFETCH
2. DUMMY PREFETCH
3. NEXT INSTRUCTION PREFETCH
4. ADDRESS BUS VALID
5. ADDRESS BUS VALID
6. ADDRESS BUS VALID
7. ADDRESS BUS VALID
8. INSTRUCTION IN VALID
9. INSTRUCTION IN VALID
10. DATA OUT VALID
11. INSTRUCTION IN VALID

NOTE 1: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

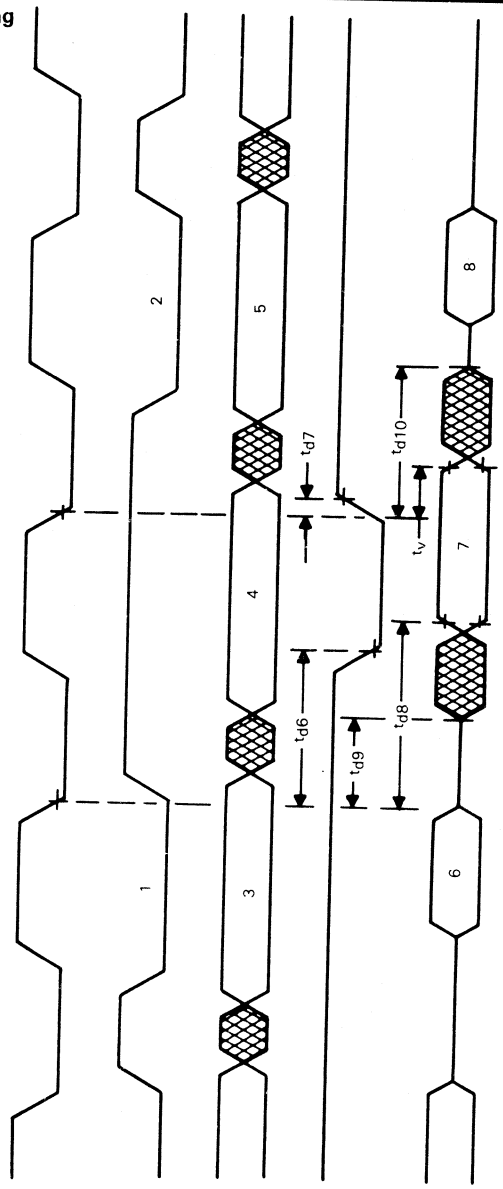
IN instruction timing

LEGEND:

1. IN INSTRUCTION PREFETCH
2. NEXT INSTRUCTION PREFETCH
3. ADDRESS BUS VALID
4. PERIPHERAL BUS VALID
5. ADDRESS BUS VALID
6. INSTRUCTION IN VALID
7. DATA IN VALID
8. INSTRUCTION IN VALID

NOTE 1: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

OUT instruction timing



LEGEND:

1. OUT INSTRUCTION PREFETCH
2. NEXT INSTRUCTION PREFETCH
3. ADDRESS BUS VALID
4. PERIPHERAL ADDRESS VALID
5. ADDRESS BUS VALID
6. INSTRUCTION IN VALID
7. DATA IN VALID
8. INSTRUCTION IN VALID

NOTE 1: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

RESET ($\overline{RS}$) TIMING

timing requirements over recommended operating conditions

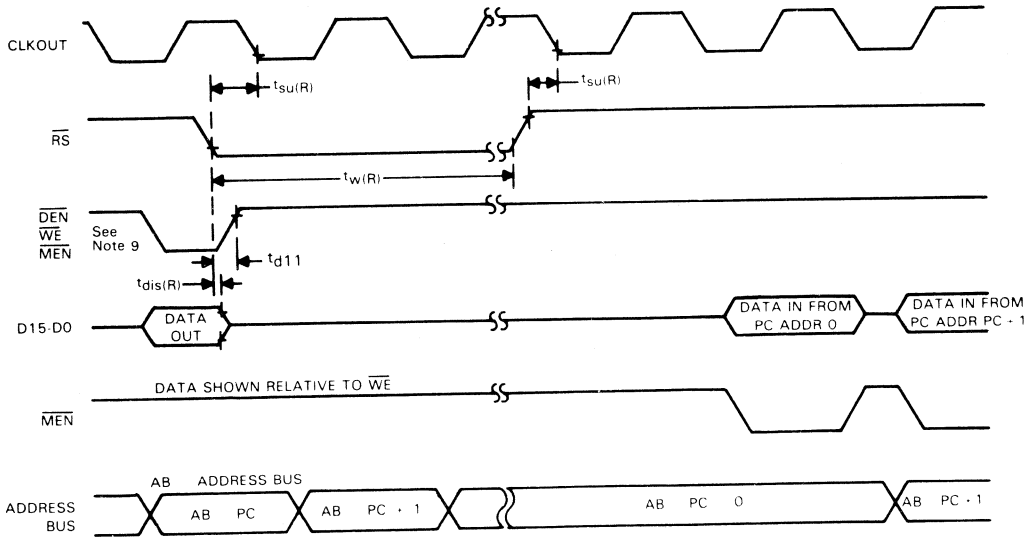
		MIN	NOM	MAX	UNIT
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 4)	50			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			ns

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF, See Figure 2			$\frac{1}{2}t_{c(C)} + 50$†	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$\frac{1}{4}t_{c(C)} + 50$†	ns

NOTE 4: RS can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.
†These values were derived from characterization data and are not tested.

reset timing



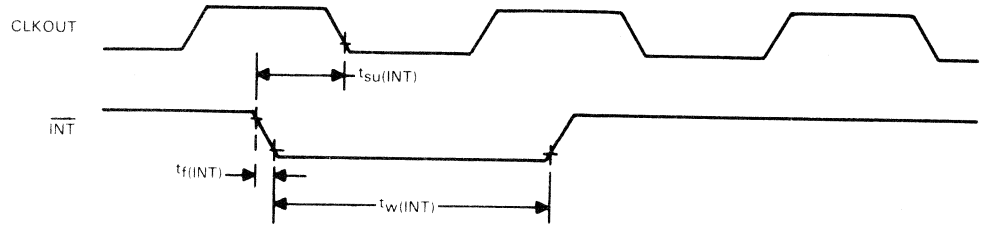
NOTES:
1. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.
5. $\overline{RS}$ forces $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ high and three-states data bus D0 through D15. AB outputs (and program counter) are synchronously cleared to zero after the next complete CLK cycle from ↓$\overline{RS}$.
6. $\overline{RS}$ must be maintained for a minimum of five clock cycles.
7. Resumption of normal program will commence after one complete CLK cycle from ↑$\overline{RS}$.
8. 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.
9. Diagram shown is for definition purpose only. $\overline{DEN}$, $\overline{WE}$, $\overline{MEN}$ are mutually exclusive.
10. During a write cycle, $\overline{RS}$ may produce an invalid write address.

INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

PARAMETER		MIN	TYP	MAX	UNIT
$t_{f(INT)}$	Fall time $\overline{INT}$ (see Note 11)			10	ns
$t_{w(INT)}$	Pulse duration $\overline{INT}$	$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time $\overline{INT}$↓ before CLKOUT↓	50			ns

interrupt timing



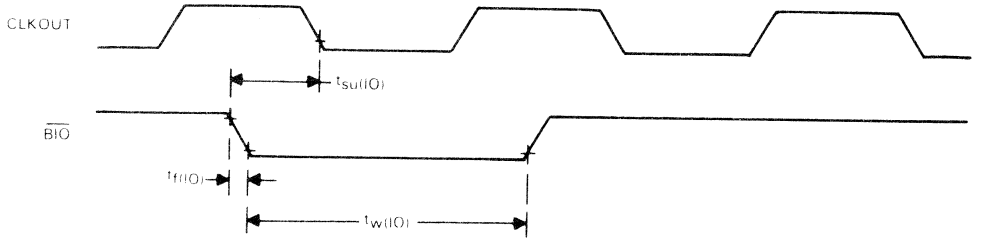
NOTES: 1. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.
11. $\overline{INT}$ fall time must be less than 15 ns.

I/O ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

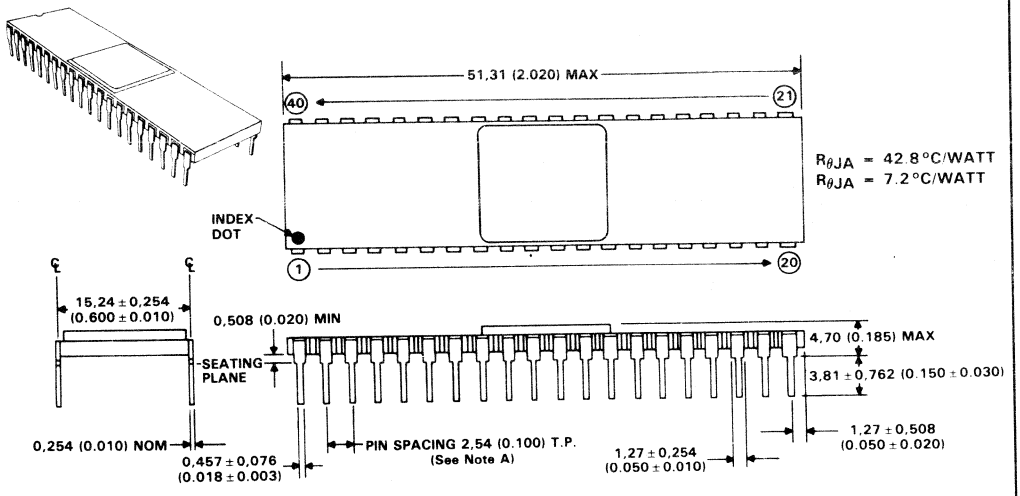
PARAMETER		MIN	TYP	MAX	UNIT
$t_{f(IO)}$	Fall time $\overline{BIO}$ (see Note 12)			10	ns
$t_{w(IO)}$	Pulse duration $\overline{BIO}$	$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time $\overline{BIO}$↓ before CLKOUT↓	50			ns

$\overline{BIO}$ timing



NOTES: 1. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.
12. $\overline{BIO}$ fall time must be less than 15 ns.

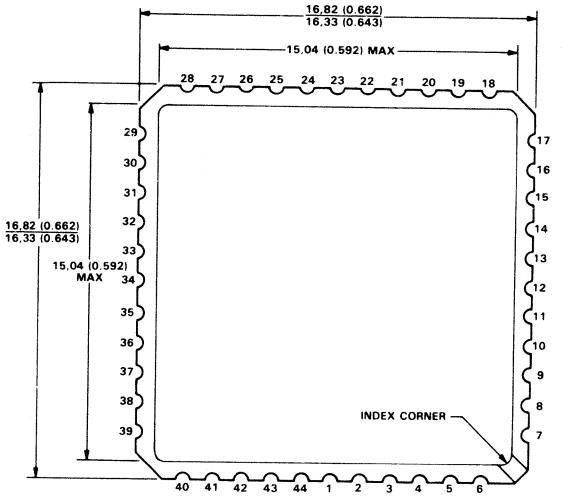
40-pin JD ceramic dual-in-line package



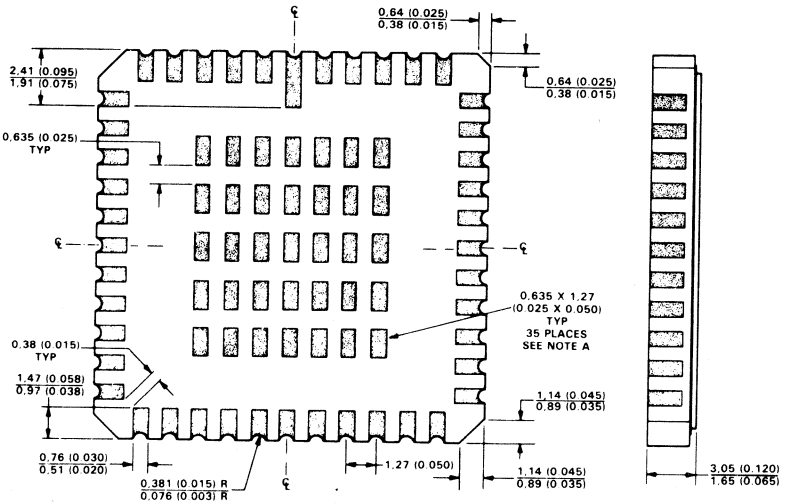
NOTE A: Each pin centerline is located within 0,254 (0.010) of its true longitudinal position.

ALL DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

44-pad ceramic chip carrier package



$R_{\theta JA}$ = 44.7°C/WATT
$R_{\theta JA}$ = 13.3°C/WATT



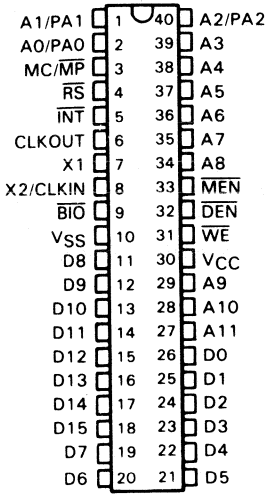
NOTE A: The checkerboard pattern is aligned vertically and is symmetrical horizontally as shown.

ALL DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

MAY 1987

- 200-ns Instruction Cycle
- 144-Word On-Chip Data RAM
- ROMless Version — SMJ320C10
- 1.5K-Word On-Chip Program ROM — SMJ320CM10
- External Memory Expansion to a Total of 4K Words at Full Speed
- 16-Bit Instruction/Data Word
- 32-Bit ALU/Accumulator
- 16 × 16-Bit Multiply in One Instruction Cycle
- 0 to 16-Bit Barrel Shifter
- Eight Input and Eight Output Channels
- 16-Bit Bidirectional Data Bus with 40-Megabits-per-Second Transfer Rate
- Interrupt with Full Context Save
- Signed Two's-Complement Fixed-Point Arithmetic
- CMOS Technology
- Single 5-V Supply

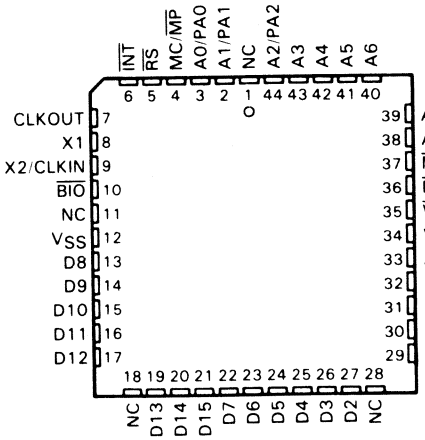
SMJ320C10 . . . JD PACKAGE
(TOP VIEW)



description

The SMJ320C10 is the first low-power CMOS member of the Texas Instruments TMS320 family of Digital Signal Processors. This device is a CMOS pin-for-pin compatible version of the industry-standard TMS32010 Digital Signal Processor. The 165-mW typical power dissipation of the SMJ320C10 enables power-sensitive applications to take advantage of the SMJ32010's high performance. The 16/32-bit microcomputer was designed to support a wide range of high-speed and numeric-intensive applications. The SMJ320C10 combines the flexibility of a high-speed controller with the numerical capability of an array processor, thereby offering an inexpensive alternative to multichip bit-slice processors. The highly pipelined architecture and efficient instruction set of the SMJ320C10 provides the capability of executing more than five million instructions per second. The instruction set is easily programmed and contains general-purpose as well as digital signal processing instructions.

SMJ320C10 . . . FD PACKAGE
(TOP VIEW)



absolute maximum ratings over specified temperature range (unless otherwise noted)†

Supply voltage range, V_{CC}‡ . . . . . −0.3 V to 7 V
Input voltage range . . . . . −0.3 V to 15 V
Output voltage range . . . . . −0.3 V to 15 V
Continuous power dissipation . . . . . 0.4 W
Maximum operating case temperature . . . . . 125°C
Minimum operating free-air temperature . . . . . −55°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		4.5	5	5.5	V
V_{SS}	Supply voltage			0		V
V_{IH}	High-level input voltage	All inputs except CLKIN	2			V
		CLKIN	3			
V_{IL}	Low-level input voltage (all inputs)				0.8	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		−55			°C
T_C	Operating case temperature				125	°C

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 1)		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}				±50	μA
I_{CC}‡ Supply current		T_A = –55°C, f_x = 20.5 MHz				75	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§		
	All others				10§		

†All typical values except for I_{CC} are at V_{CC} = 5 V, T_A = 25°C.
‡I_{CC} characteristics are inversely proportional to temperature; i.e., I_{CC} decreases approximately linearly with temperature.
§Value derived from characterization data and not tested.
NOTE 1: This voltage specification is included for interface to HC logic. However, note that all other timing parameters defined in this data sheet are specified for TTL logic levels and will differ for HC logic levels.

CLOCK CHARACTERISTICS AND TIMING

The SMJ320C10 can use either its internal oscillator or an external frequency source for a clock.

internal clock option

The internal oscillator is enabled by connecting a crystal across X1 and X2/CLKIN (see Figure 1). The frequency of CLKOUT is one-fourth the crystal fundamental frequency. The crystal should be fundamental mode, and parallel resonant, with an effective series resistance of 30 ohms, a power dissipation of 1 mW, and be specified at a load capacitance of 20 pF.

PARAMETER	TEST CONDITIONS	MIN	NOM	MAX	UNIT
Crystal frequency f_x	–55°C to 125°C	6.7		20.5	MHz
C1, C2			10		pF

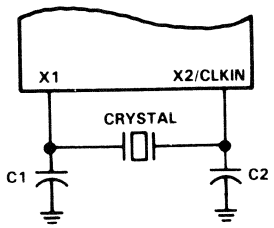


FIGURE 1. INTERNAL CLOCK OPTION

external clock option

An external frequency source can be used by injecting the frequency directly into X2/CLKIN with X1 left unconnected. The external frequency injected must conform to the specifications listed in the table below.

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{c(MC)}$	Master clock cycle time	48.78		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†	$0.475t_{c(C)}$		$0.525t_{c(C)}$	ns
$t_{w(MCL)}$	Pulse duration master clock low, $t_{c(MC)}$ = 50 ns		20		ns
$t_{w(MCH)}$	Pulse duration master clock high, $t_{c(MC)}$ = 50 ns		20		ns

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLKOUT cycle time‡	R_L = 825 Ω, C_L = 100 pF, See Figure 2	195.12		600	ns
$t_{r(C)}$	CLKOUT rise time			10		ns
$t_{f(C)}$	CLKOUT fall time			8		ns
$t_{w(CL)}$	Pulse duration, CLKOUT low			92		ns
$t_{w(CH)}$	Pulse duration, CLKOUT high			90		ns
$t_{d(MCC)}$	Delay time CLKIN↑ to CLKOUT↓		25		60	ns

†Values given were derived from characterization data and are 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).

PARAMETER MEASUREMENT INFORMATION

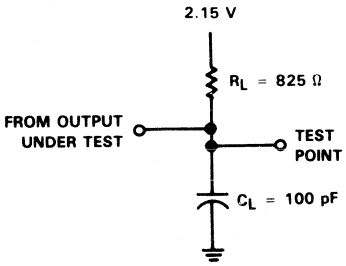
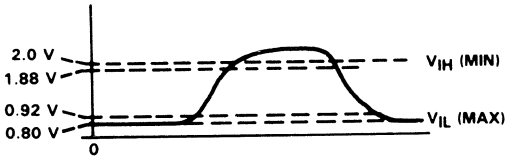
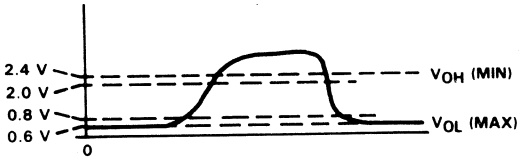


FIGURE 2. TEST LOAD CIRCUIT



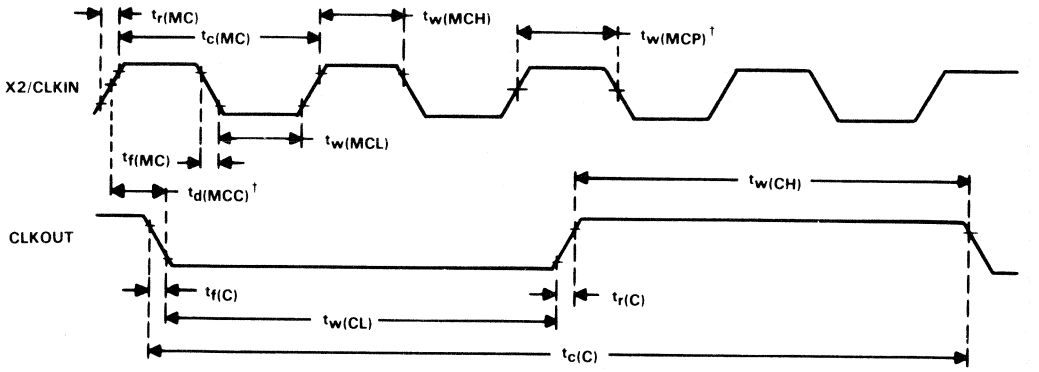
(a) INPUT



(b) OUTPUTS

FIGURE 3. VOLTAGE REFERENCE LEVELS

clock timing



†$t_{d(MCC)}$ and $t_{w(MCP)}$ are referenced to an intermediate level of 1.5 volts on the CLKIN waveform.
NOTE 2: 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.

MEMORY AND PERIPHERAL INTERFACE TIMING

switching characteristics over recommended operating conditions

	PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d1}	Delay time CLKOUT↓ to address bus valid (see Note 3)	R_L = 825 Ω, C_L = 100 pF, See Figure 2	10†		50	ns
t_{d2}	Delay time CLKOUT↓ to $\overline{MEN}$↓		¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 15	ns
t_{d3}	Delay time CLKOUT↓ to $\overline{MEN}$↑		−10†		15	ns
t_{d4}	Delay time CLKOUT↓ to $\overline{DEN}$↓		¼$t_{c(C)}$ − 5†		¼$t_{c(C)}$ + 15	ns
t_{d5}	Delay time CLKOUT↓ to $\overline{DEN}$↑		−10†		15	ns
t_{d6}	Delay time CLKOUT↓ to $\overline{WE}$↓		½$t_{c(C)}$ − 5†		½$t_{c(C)}$ + 15	ns
t_{d7}	Delay time CLKOUT↓ to $\overline{WE}$↑		−10†		15	ns
t_{d8}	Delay time CLKOUT↓ to data bus OUT valid				¼$t_{c(C)}$ + 65	ns
t_{d9}	Time after CLKOUT↓ that data bus starts to be driven		¼$t_{c(C)}$ − 5†			ns
t_{d10}	Time after CLKOUT↓ that data bus stops being driven				¼$t_{c(C)}$ + 30†	ns
t_v	Data bus OUT valid after CLKOUT↓		¼$t_{c(C)}$ − 10			ns
$t_{su(A-MD)}$	Address bus setup time prior to $\overline{MEN}$↓ or $\overline{DEN}$↓		5			ns

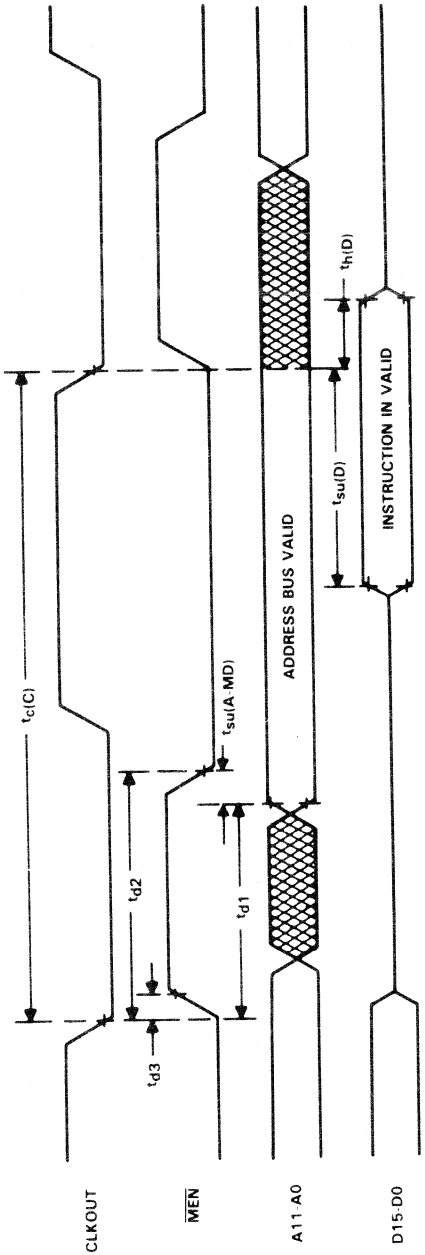
NOTE 3: Address bus will be valid upon $\overline{WE}$↑, $\overline{DEN}$↑, or $\overline{MEN}$↑.
†These values were derived from characterization data and are not tested.

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 Ω, C_L = 100 pF, See Figure 2	50			ns
$t_{h(D)}$	Hold time data bus held valid after CLKOUT↓ (see Note 4)		0			ns

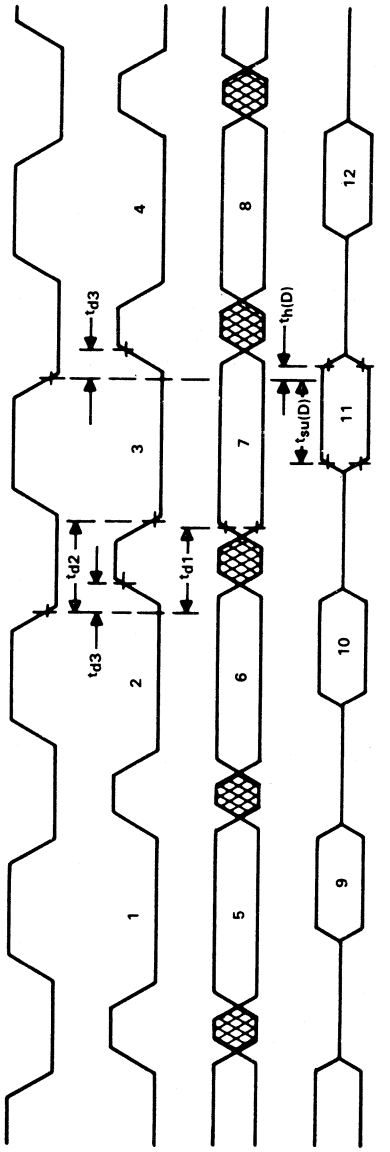
NOTE 4: Data may be removed from the data bus upon $\overline{MEN}$↑ or $\overline{DEN}$↑ preceding CLKOUT↓.

nemory read



NOTE 2: 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.

TBLR instruction timing

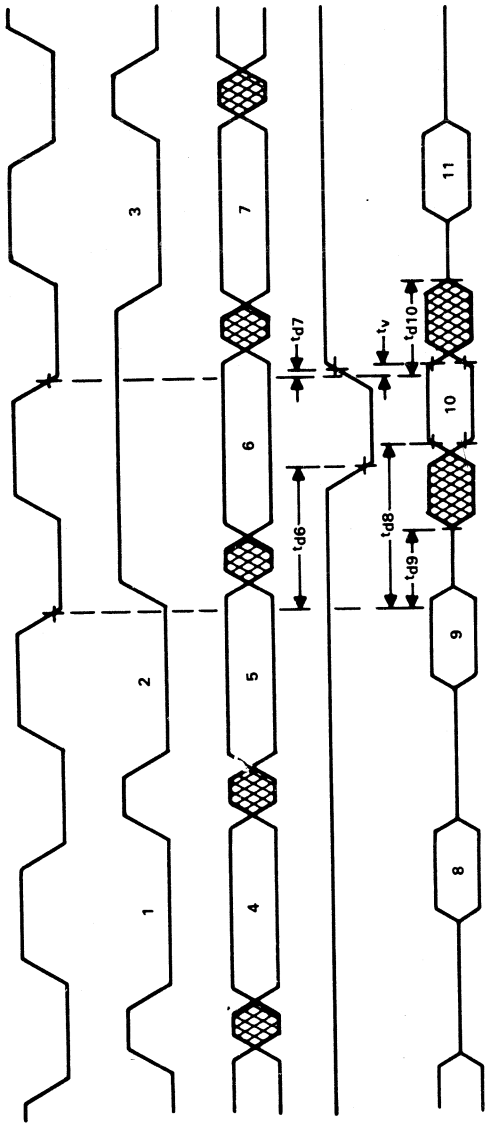


LEGEND:

1. TBLR INSTRUCTION PREFETCH
2. DUMMY PREFETCH
3. DATA FETCH
4. NEXT INSTRUCTION PREFETCH
5. ADDRESS BUS VALID
6. ADDRESS BUS VALID
7. ADDRESS BUS VALID
8. ADDRESS BUS VALID
9. INSTRUCTION IN VALID
10. INSTRUCTION IN VALID
11. DATA IN VALID
12. INSTRUCTION IN VALID

NOTE 2: 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.

TBLW instruction timing

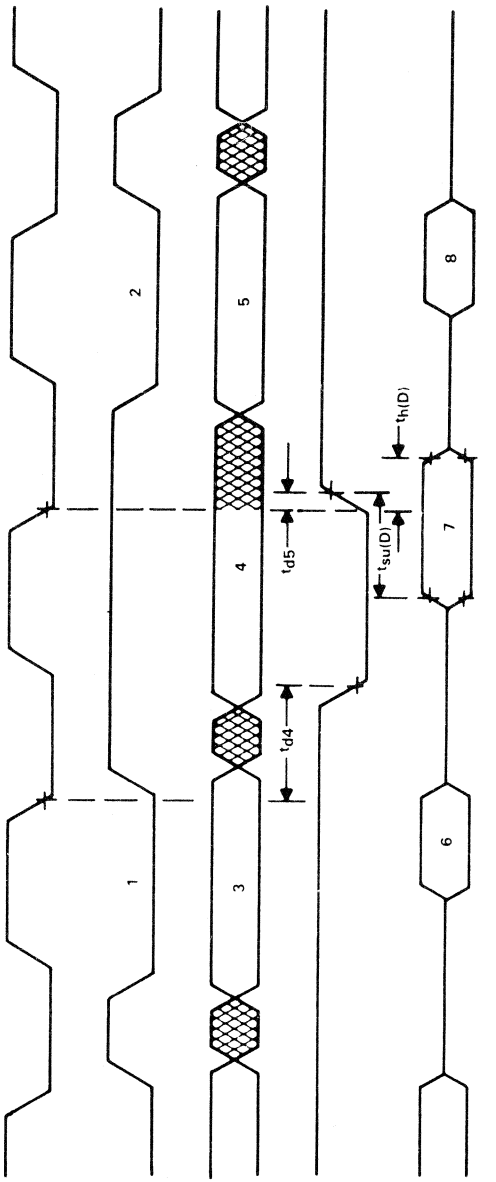


LEGEND:

1. TBLW INSTRUCTION PREFETCH
2. DUMMY PREFETCH
3. NEXT INSTRUCTION PREFETCH
4. ADDRESS BUS VALID
5. ADDRESS BUS VALID
6. ADDRESS BUS VALID
7. ADDRESS BUS VALID
8. INSTRUCTION IN VALID
9. INSTRUCTION IN VALID
10. DATA OUT VALID
11. INSTRUCTION IN VALID

NOTE 2: 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.

IN instruction timing

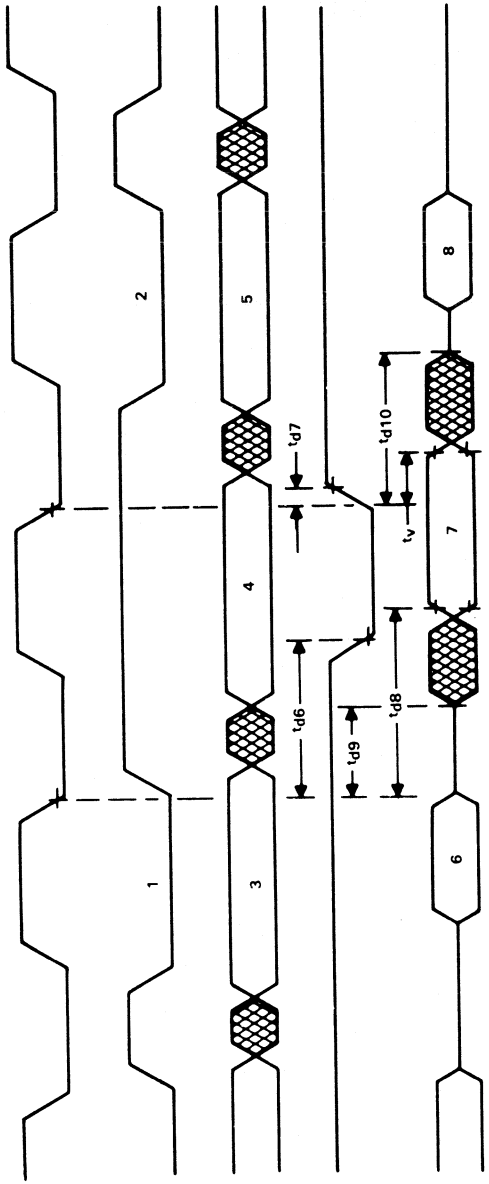


LEGEND:

1. IN INSTRUCTION PREFETCH
2. NEXT INSTRUCTION PREFETCH
3. ADDRESS BUS VALID
4. PERIPHERAL ADDRESS VALID
5. ADDRESS BUS VALID
6. INSTRUCTION IN VALID
7. DATA IN VALID
12. INSTRUCTION IN VALID

NOTE 2: 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.

)UT instruction timing



LEGEND:

1. OUT INSTRUCTION PREFETCH
2. NEXT INSTRUCTION PREFETCH
3. ADDRESS BUS VALID
4. PERIPHERAL ADDRESS VALID
5. ADDRESS BUS VALID
6. INSTRUCTION IN VALID
7. DATA IN VALID
12. INSTRUCTION IN VALID

NOTE 2: 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.

RESET ($\overline{RS}$) TIMING

timing requirements over recommended operating conditions

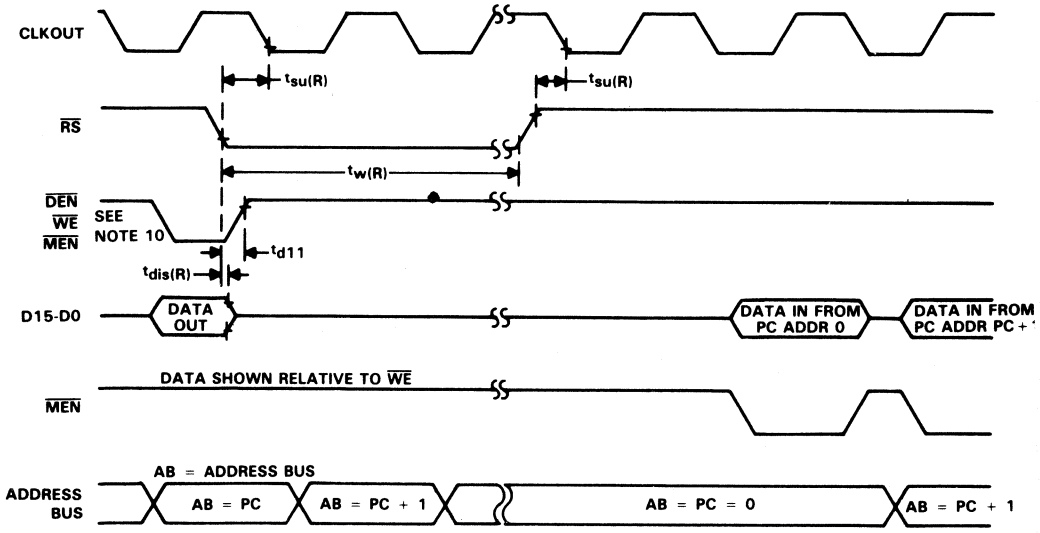
		MIN	NOM	MAX	UNIT
$t_{su(R)}$	Reset ($\overline{RS}$) setup time prior to CLKOUT (see Note 5)	50			ns
$t_{w(R)}$	$\overline{RS}$ pulse duration	$5t_{c(C)}$			ns

switching characteristics over recommended operating conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{d11}	Delay time $\overline{DEN}$↑, $\overline{WE}$↑, and $\overline{MEN}$↑ from $\overline{RS}$	R_L = 825 Ω, C_L = 100 pF, See Figure 2			$\frac{1}{2}t_{c(C)} + 50^{\dagger}$	ns
$t_{dis(R)}$	Data bus disable time after $\overline{RS}$				$\frac{1}{4}t_{c(C)} + 50^{\dagger}$	ns

NOTE 5: $\overline{RS}$ can occur anytime during a clock cycle. Time given is minimum to ensure synchronous operation.
†These values were derived from characterization data and are not tested.

reset timing



NOTES:
2. 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.
6. $\overline{RS}$ forces $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ high and tristates data bus D0 through D15. 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 ↓$\overline{RS}$ occurs in the CLK cycle.
10. Diagram shown is for definition purposes only. $\overline{DEN}$, $\overline{WE}$, and $\overline{MEN}$ are mutually exclusive.
11. During a write cycle, $\overline{RS}$ may produce an invalid write address.

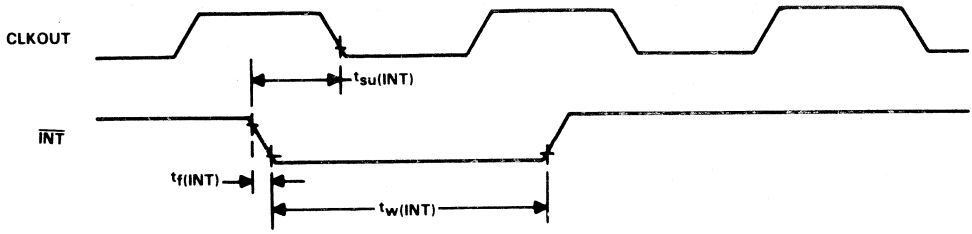
INTERRUPT ($\overline{INT}$) TIMING

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{f(INT)}$	Fall time $\overline{INT}$ (see Note 12)			10	ns
$t_{w(INT)}$	Pulse duration $\overline{INT}$	$t_{c(C)}$			ns
$t_{su(INT)}$	Setup time $\overline{INT}$↓ before CLKOUT↓	50			ns

NOTE 12. $\overline{INT}$ fall time must be less than 15 ns.

interrupt timing



NOTE 2. 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.

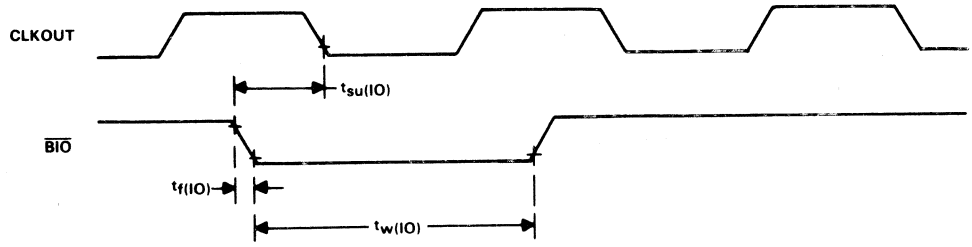
I/O ($\overline{BIO}$) TIMING

timing requirements over recommended operating conditions

		MIN	NOM	MAX	UNIT
$t_{f(IO)}$	Fall time $\overline{BIO}$ (see Note 13)			10	ns
$t_{w(IO)}$	Pulse duration $\overline{BIO}$	$t_{c(C)}$			ns
$t_{su(IO)}$	Setup time $\overline{BIO}$↓ before CLKOUT↓	50			ns

NOTE 13. $\overline{BIO}$ fall time must be less than 15 ns.

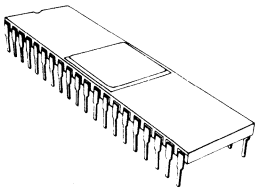
$\overline{BIO}$ timing



NOTE 2. 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.

MECHANICAL DATA

40-pin JD ceramic dual-in-line package

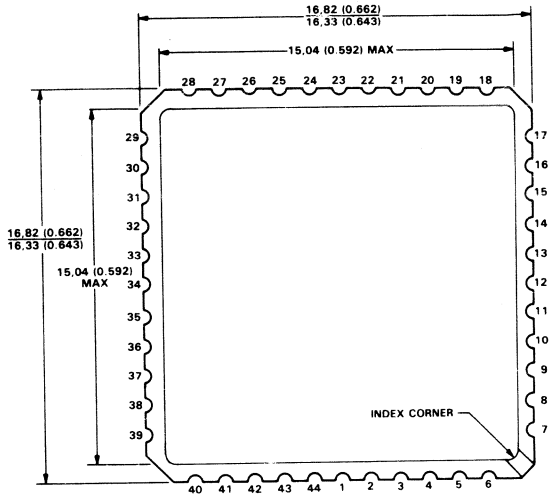


NOTE A: Each pin centerline is located within 0,254 (0.010) of its true longitudinal position.

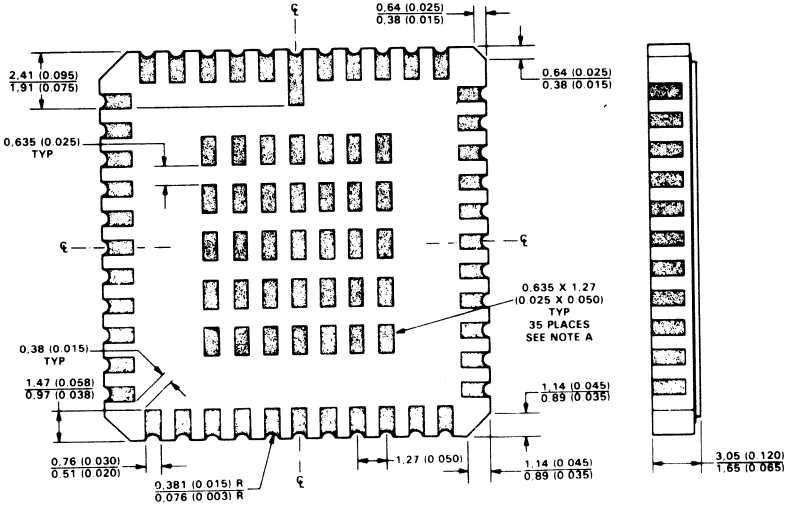
ALL DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

MECHANICAL DATA

44-pad ceramic chip carrier package



$R_{\theta JA}$ = 44.7°C/WATT
$R_{\theta JA}$ = 13.3°C/WATT



NOTE A: The checkerboard pattern is aligned vertically and is symmetrical horizontally as shown.

ALL DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

C. 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, Texas Instruments offers microcomputer versions for TMS320C1x (first-generation TMS320) devices. The on-chip ROM of these processors can be masked with the customer's own code. This allows the user 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, all prototype work is performed using a standard TMS320C1x microprocessor. TMS320C1x 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 MC/$\overline{\text{MP}}$ (microcomputer/microprocessor) mode, offered on maskable TMS320C1x devices (excluding the TMS320C17/E17, and TMS320C17-25), often shortens design and field upgrade cycle times, thereby reducing expense. This mode permits the customer to use the TMS320C1x as a standard device operating out of external program memory.
When TMS320C1x 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 masked into the on-chip ROM space of a TMS320C1x device. TMS320C1x programs can also be expanded using external memory. With a reduced chip count and this program memory flexibilty, multiple functions can be easily implemented in a single hardware device, thus enhancing a product's capabilities.

The TMS320C1x devices with mask option include the TMS32010/C10 with 1.5K words of on-chip ROM, and the TMS320C15/C17 with 4K words of on-chip ROM. The customer's code must fit within the specified ROM size of the chosen processor.

Figure C-1 illustrates the procedure flow for implementing TMS320C1x masked parts.

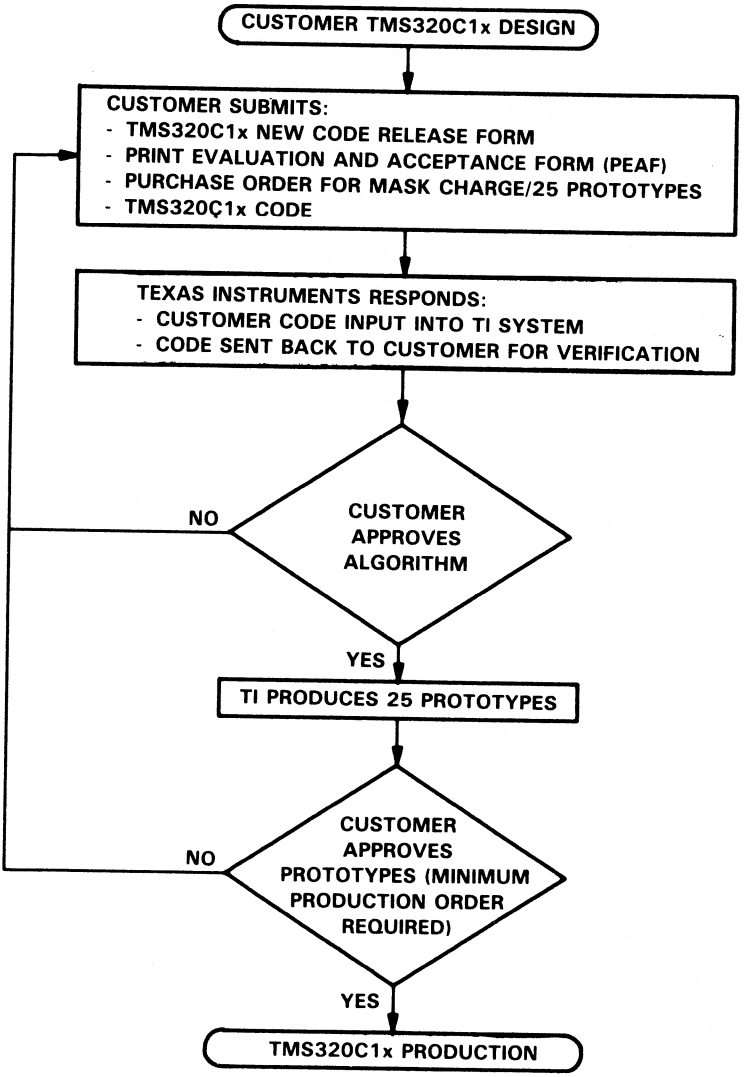


Figure C-1. TMS320C1x ROM Code Flowchart

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 TMS320C1x prototypes is 8 weeks and for masked TMS320C1x 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 TMS320C1x ROM code may be submitted in one of the following formats (the preferred media is 5 1/4" floppies):

PROM:	TBP28S166, TBP28S86
EPROM:	TMS2764, TMS2508, TMS2516, TMS2532, TMS2564
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.

D. 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.

D.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 D-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 D-1. Microprocessor and Microcontroller Tests

TEST	DURATION	SAMPLE SIZE	
		PLASTIC	CERAMIC
Operating life, 125°C, 5.0 V	1000 hrs	195	195
Operating life, 150°C, 5.0 V	1000 hrs	77*	77
Storage life, 150°C	1000 hrs	129	129
Biased 85°C/85 percent RH, 5.0 V	1000 hrs	129	-
Autoclave, 121°C, 1 ATM	240 hrs	105	-
Temperature cycle, -65 to 150°C	1000 cyc	129	129
Thermal shock, -65 to 150°C	500 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		-	15
Internal water vapor		-	5
Solderability		22	22
Solder heat		22	22
Resistance to solvents		12	12
Lead integrity		15	15
Lead pull		15	-
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.

Table D-2 provides a list of the TMS320C1x devices, the approximate number of transistors, and the equivalent gates. The numbers have been determined from design verification runs.

Table D-2. TMS320C1x Transistors

DEVICE	# TRANSISTORS	# GATES
NMOS:		
TMS32010 (all speeds)	50K	17K
CMOS:		
TMS320C10 (all speeds)	58K	15K
TMS320C15 (all speeds)	110K	44K
TMS320E15	113K	45K
TMS320C17 (all speeds)	115K	46K
TMS320E17	118K	47K

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.

E. Development Support/Part Order Information

This section provides development support information, device part numbers, and support tool ordering information for all TMS320C1x (first-generation TMS320) products. Figure E-1 shows the software and hardware development tools available for the TMS320C1x.

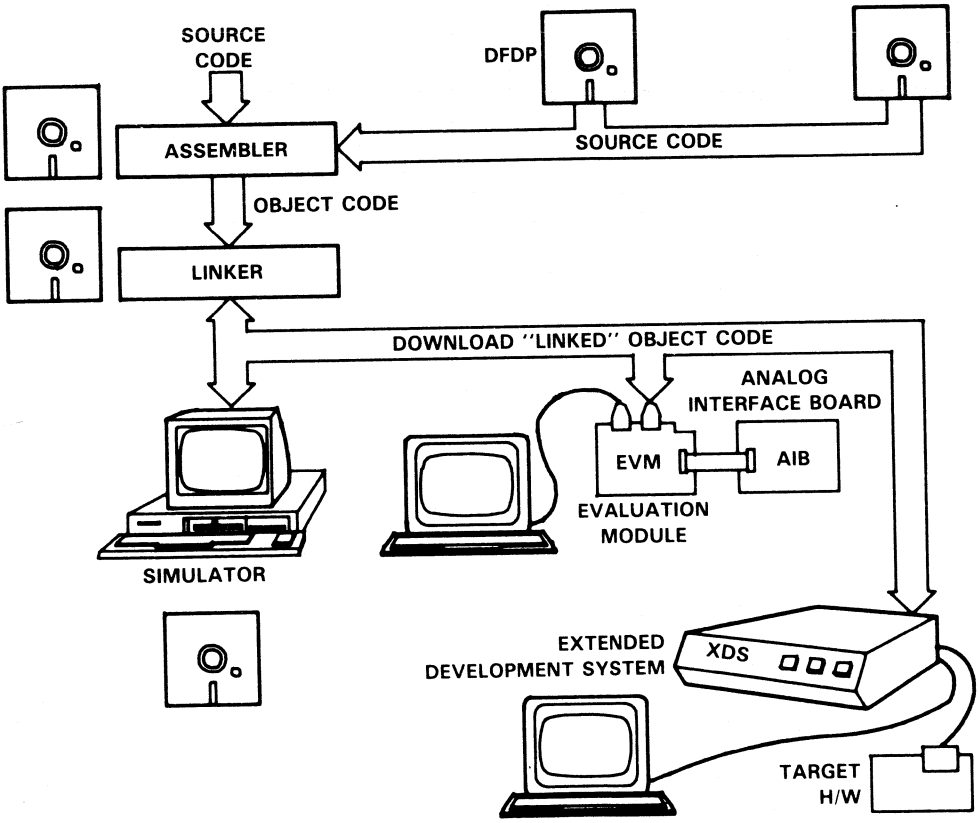


Figure E-1. TMS320C1x Development Tools

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 Regional Technology Centers. For more information about support products and documentation, refer to the *TMS320 Family Development Support Reference Guide*.

The nearest TI field sales office can be contacted for support tool availability or further details (see list of sales offices at end of book). For technical support, contact the TI Regional Technology Centers.

The major topics discussed in this section are listed below.

E.1 First-Generation TMS320 Development Support

Texas Instruments offers extensive development support and complete documentation with the first-generation TMS320 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 Macro Assembler/Linker, Simulator, Evaluation Module (EVM), Emulator (XDS), and other support products.

A description and key features for each TMS320C1x development support tool is provided in the following subsections. For more information about support products, refer to the *TMS320 Family Development Support Reference Guide*. For ordering information, see Section E.2.

E.1.1 TMS320C1x Macro Assembler/Linker

The TMS320C1x Macro Assembler translates TMS320C1x assembly language source code into executable object code. The assembler allows the programmer to work with mnemonics rather than hexadecimal machine instructions and to reference memory locations with symbolic addresses. The macro assembler supports macro calls and definitions along with conditional assembly.

The TMS320C1x Linker permits a program to be designed and implemented in separate modules that will later be linked together to form the complete program. The linker resolves external definitions and references for relocatable code, creating an object file that can be executed by the TMS320C1x Simulator, Emulator, or DSP device.

The following key features distinguish the TMS320C1x Macro Assembler/Linker:

- Macro capabilities and library functions
- Conditional assembly
- Relocatable modules
- Complete error diagnostics
- Symbol table and cross reference.

The macro assembler/linker is currently available for the VAX/VMS, TI PC/MS-DOS, and IBM PC/PC-DOS operating systems.

E.1.2 TMS320C1x Simulator

The TMS320C1x Simulator is a software program that simulates operation of the TMS320C1x 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 Macro Assembler/Linker. 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.

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
- I/O device simulation.

The simulator is currently available for the VAX/VMS, TI PC/MS-DOS, and IBM PC/PC-DOS operating systems.

E.1.3 TMS320C1x Evaluation Module (EVM)

The TMS320C1x Evaluation Module (EVM) is a low-cost development board for TMS32010/C10/C15/E15 devices, used for full-speed in-circuit emulation and hardware debugging. (Note that the EVM does not support the TMS320C17/E17 devices.) It consists of a single board that enables a designer to evaluate certain characteristics of the processor to determine if it meets the requirements of an application.

The powerful firmware package of the TMS320C1x EVM contains a debug monitor, assembler/reverse assembler, and software communication via three EIA ports. The EVM can communicate to a host computer and several peripherals. The three EIA ports allow the EVM to communicate with a designer's terminal, a host computer, a printing device, or audio cassette. In addition, the EVM also supports an onboard PROM utility for programming TMS2764 EPROMs, used for mass program storage.

The EVM assembles source code created on a host computer or on the EVM's text editor, a line-numbered editor with character-editing capabilities. The EVM has a one-pass assembler, which resolves both forward and reverse labels and converts the incoming text into executable code. Object code produced by the EVM assembler is stored in memory. The reverse assembler converts object code back to assembly language mnemonics, and the patch assembler allows modification of the code.

Some key features of the TMS320C1x EVM are:

- On-board TMS32010
- 20-MHz operation
- Event counter for one breakpoint
- Text editor
- On-board EPROM programmer
- Audio cassette interface
- 4K words of on-board program RAM
- Target connector for full-speed in-circuit emulation from EVM memory
- Debug monitor including commands with full prompting
- Line-by-line assembler/reverse assembler
- Transparency mode for host CPU upload/download
- Eight instruction breakpoints available
- Single-step execution with software trace
- Standalone or host CPU configurable.

The TMS320C1x EVM functions in two modes: host computer mode or PC mode (single-user system). In the host computer mode, object and source code can be uploaded/downloaded between the host computer and EVM. In the PC mode, the EVM can support host uploads/downloads over a single port to allow a single-user system, such as a TI or IBM PC, to function as both a terminal and a host (see Figure E-2). Commercially available terminal emulation software for the single-user system is required in this configuration.

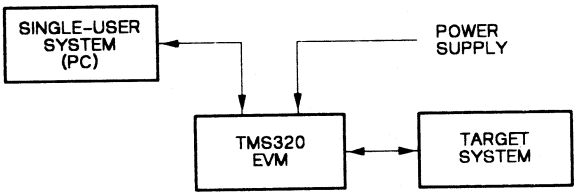


Figure E-2. TMS320C1x EVM/Single-User System

E.1.4 TMS320C1x Emulator (XDS)

The TMS320C1x Emulator (XDS/22) is a user-friendly system that has all the features necessary for realtime in-circuit emulation. This allows integration of hardware and software modules in the debug mode. 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 Macro Assembler/Linker 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 TMS320C1x 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 TMS320C1x 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 E-3 shows a block diagram of a typical system configuration using the TMS320C1x XDS/22 Emulator.

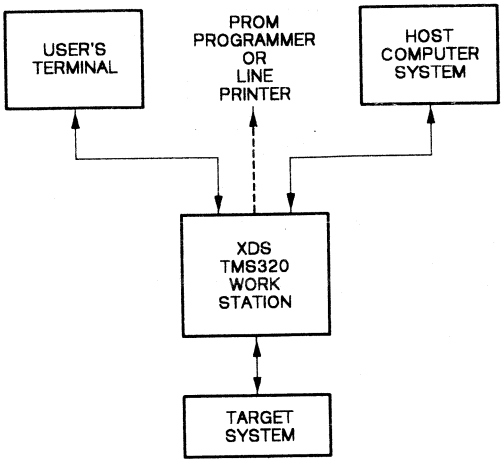


Figure E-3. TMS320C1x XDS/22 System Configuration

E.1.5 TMS320C1x XDS/22 Upgrade

Texas Instruments offers a TMS320C1x XDS upgrade kit, which extends the functionality of existing development systems at a minimum of cost through an enhancement of current customer equipment. The upgrade kit can enable a TMS32010 XDS/22 to emulate operation of all first-generation devices. Note that early systems support TMS32010, TMS32010-14, and TMS320C10 performance. Upgrade kits allow upgrade only within a generation, not from a first- to a second-generation XDS.

E.1.6 TMS320 Analog Interface Board

The TMS320 Analog Interface Board (AIB) is an analog-to-digital, digital-to-analog conversion board used as a preliminary target system with the TMS320C1x EVM, XDS, or another emulator (see Figure E-4). The AIB is an educational tool that provides a simple, inexpensive way to become familiar with digital signal processing (DSP) techniques.

The AIB allows testing of application programs with analog I/O by providing an interface to the TMS320C1x. The AIB provides 12-bit A/D and D/A converters with expansion ports for additional A/D and D/A converters. Key features of the AIB are as follows:

- 12-bit analog-to-digital converter with sample and hold
- 12-bit digital-to-analog converter
- One 16-bit output port for additional D/A or user-defined application
- One 16-bit input port for additional A/D or user-defined application
- Two lowpass filters; an audio amplifier
- TBLW (table write) decode
- Extended I/O data memory
- Prototyping area for user applications.

The sample rate clock on the AIB is derived from the CLKOUT signal on the TMS320 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 band-limits 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. An audio amplifier that will drive an 8-ohm speaker is provided for applications with audio output. 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.

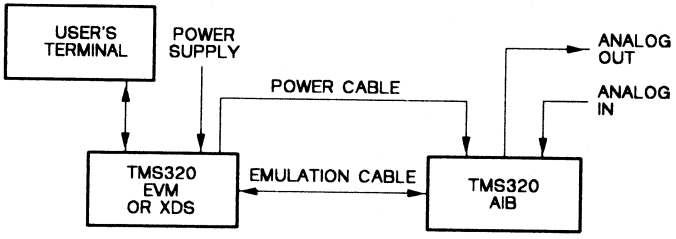


Figure E-4. TMS320 AIB System Configuration

E.1.7 TMS320 Design Kit

The TMS320 DSP Design Kit has been created by Texas Instruments to aid the user in becoming familiar with the TMS320 family of digital signal processors, thus accelerating the evaluation of these devices. The kit contains the following:

- Samples: one TMS32020GBL, one TMS32010NL, one Codec (TCM2916), and four preprogrammed PROMs (TBP38L165-45).
- ADPCM Design Example using the TMS32010.
- FFT Design Example using the TMS32020.
- *Digital Signal Processing Applications with the TMS320 Family*, a comprehensive 750-page book filled with TMS320 applications.
- Digital Signal Processing Software Library, containing source code for most of the DSP applications discussed in the Applications Book as well as other valuable routines.
- TMS320C1x and TMS320C2x User's Guides.
- Latest copy of the TMS320 quarterly newsletter, *Details on Signal Processing*.

The Design Kit is available through local TI authorized distributors or directly from Texas Instruments. Contact the nearest TI Sales Office for more information.

E.1.8 TMS320E15 EPROM DSP Starter Kit

To assist with developing, debugging, and testing programs, Texas Instruments offers the TMS320E15 EPROM DSP Starter Kit. The kit includes the following:

- TMS320C1x Evaluation Module (EVM) to provide a standalone development system for the TMS32010/C10/C15/E15.
- Two TMS320E15JDL devices (TMS320EPROM/15 - EPROM DSP Twin-Pack), each of which provides an on-chip 256-word RAM and 4K-word program EPROM for realtime code development and modification. The device is object-code and pin compatible with the TMS32010/C10 and features EPROM code protection for copyright security.
- 40-pin to 28-pin EPROM programmer adaptor socket (RTC/PGM320A-06) to facilitate TMS320E15 programming using the EVM or any standard PROM programmer capable of programming 28-pin 64K CMOS EPROMs.
- Documentation.

Contact the nearest TI sales office or distributor for availability or further information regarding the TMS320E15 EPROM DSP Starter Kit (Part # RTC/EVM320E-15).

E.1.9 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 TI PC, IBM PC/XT/AT, and compatible systems. Operating systems must have 192 kbytes of memory available. For more information, contact the nearest TI field sales office.

E.1.10 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*. 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 TI/IBM MS/PC-DOS (version 1.1 or later) or a 1600 BPI magnetic tape for the VAX/VMS version. All the directories on the MS/PC-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*, is the major reference for the theory and algorithms, and also provides printed code in the appendices of each application report.

The software library and applications book are included in the purchase of a TMS320 Design Kit (see Section B.1.7). The library can also be ordered separately through TI (see Table E-2 for ordering information).

All the software in the library is copyrighted by Texas Instruments. The library is continually being updated; therefore, check the nearest TI sales office.

E.1.11 TMS320 Bell 212A Modem Software

Texas Instruments is offering a software package containing source code and documentation for the design and implementation of a 1200-bps Bell 212A modem with the TMS320 digital signal processor and the TMS7000 microcontroller.

The documentation included in the package consists of two reports. One report discusses in detail the theory behind the design of the modem, as well as the functions implemented. The second report describes the hardware, algorithms, and coding techniques used in the implementation of a Bell 212A modem demonstration unit. This implementation has been built and tested to verify its operation. After reading this report, the user should be able to design and build a similar unit as well as understand some tradeoffs involved in making custom modifications.

The source code for the TMS320 Bell 212A Modem Software package is provided on a 5 1/4" floppy for MS/PC-DOS or compatible operating systems. Contact the nearest TI field sales office for further information.

E.2 Part Order Information

This section provides the device and support tool part numbers. Table E-1 lists the part numbers for all the first-generation members of the TMS320 family. Table E-2 gives ordering information for TMS320C1x hardware and software support tools. Table E-3 provides a list and description of the development tool connections to a target system. 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 E-1. TMS320C1x Digital Signal Processor Part Numbers

DEVICE NAME	TECHNOLOGY	OPERATING FREQUENCY	PACKAGE TYPE	TYPICAL DISSIPATION
TMS32010NL-25	2.4-µm NMOS	25 MHz	Plastic 40-pin DIP	900 mW
TMS32010NL	2.4-µm NMOS	20 MHz†		900 mW
TMS32010NL-16	2.4-µm NMOS	16 MHz		900 mW
TMS320C10NL-25	2.0-µm CMOS	25 MHz	Plastic 40-pin DIP	200 mW
TMS320C10NL	2.0-µm CMOS	20 MHz‡		165 mW
TMS320C10FNL-25	2.0-µm CMOS	25 MHz	Plastic 44-lead PLCC	200 mW
TMS320C10FNL	2.0-µm CMOS	20 MHz		165 mW
TMS320C15NL-25	2.0-µm CMOS	25 MHz	Plastic 40-pin DIP§	250 mW
TMS320C15NL	2.0-µm CMOS	20 MHz‡		225 mW
TMS320E15JDL	2.0-µm CMOS	20 MHz	Ceramic 40-pin DIP	300 mW
TMS320C17NL-25	2.0-µm CMOS	25 MHz	Plastic 40-pin DIP§	275 mW
TMS320C17NL	2.0-µm CMOS	20 MHz‡		250 mW
TMS320E17JDL	2.0-µm CMOS	20 MHz	Ceramic 40-pin DIP	325 mW

†Military version available
‡Military versions planned; contact nearest sales office for availability.
§PLCC version planned; contact nearest sales office for availability.

Table E-2. TMS320C1x Support Tool Part Numbers

TOOL DESCRIPTION	OPERATING SYSTEM	PART NUMBER
SOFTWARE		
Macro Assembler/Linker	VAX VMS TI/IBM MS/PC-DOS	TMDS3240210-08 TMDS3240810-02
Simulator	VAX VMS TI/IBM MS/PC-DOS	TMDS3240211-08 TMDS3240811-02
Digital Filter Design Package	TI PC MS-DOS IBM PC PC-DOS	DFDP/TI001 DFDP/IBM001
DSP Software Library	VAX VMS TI/IBM MS/PC-DOS	TMDC3240212-18 TMDC3240812-12
TMS320 Bell 212A Modem Software	TI/IBM PC/MS-DOS	TMDX3240813-12
HARDWARE		
Evaluation Module (EVM)†		RTC/EVM320A-03
XDS/22 Emulator†		TMDS3262211
XDS/22 Upgrade	Factory Upgrade† Customer Upgrade†	TMDS3282215 TMDS3282216
Analog Interface Board		RTC/EVM320C-06
TMS320 Design Kit		TMS320DDK
TMS320E15 EPROM DSP Starter Kit		RTC/EVM320E-15
EPROM Programmer Adaptor Socket		RTC/PGM320A-06

†See Table A-3 for a list of connections to a target system.

Table E-3. Development Tool Connections to a Target System

TOOL	TARGET CONN.	INCL.	OPT.	PART NUMBER
TMS320C10 XDS/22	40-pin DIP 44-lead PLCC	X	X	TMDS3288810
TMS320C10 XDS/22 Upgrade	40-pin DIP 44-lead PLCC	X	X	TMDS3288810
TMS32010 EVM	40-pin DIP	X		

E.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.

E.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 E-5 provides a legend for reading the complete device name for any TMS320 family member.

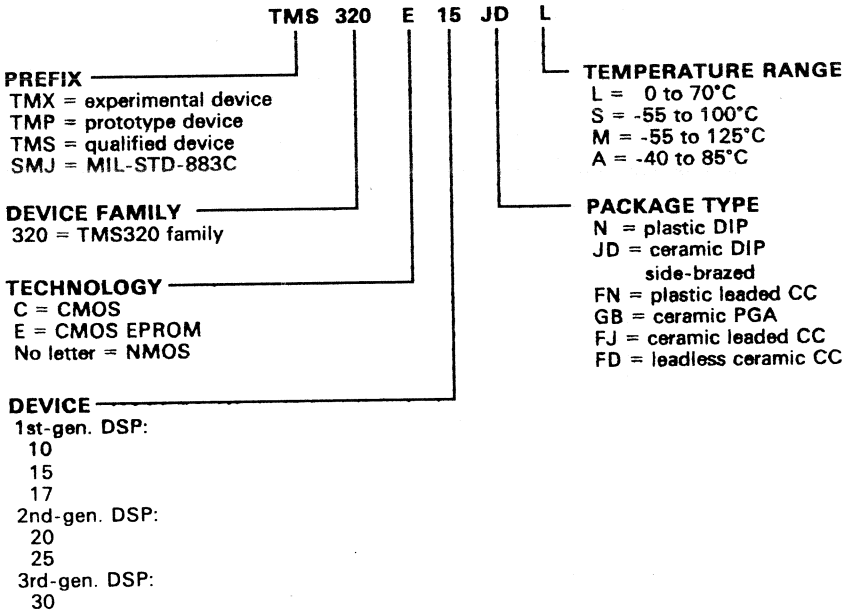
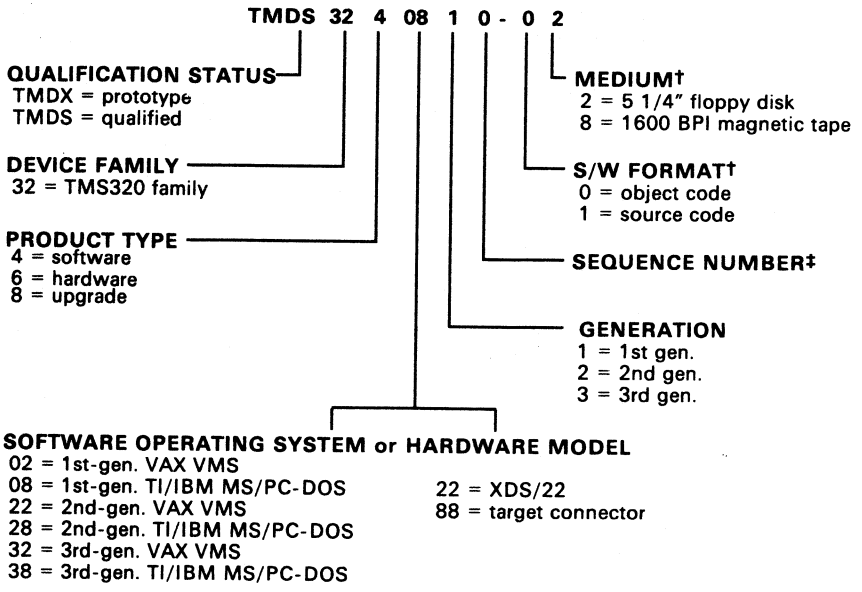


Figure E-5. TMS320 Device Nomenclature

Figure E-6 provides a legend for reading the part number for any TMS320 hardware or software development tool.



† Software only.
‡ Hardware only.

Figure E-6. TMS320 Development Tool Nomenclature

F. Memories, Peripherals, and Sockets

This appendix provides product information regarding memories, peripherals, and sockets, which are manufactured by Texas Instruments and compatible with the TMS320C1x. Information is also given regarding crystal frequencies, specifications, and vendors.

The contents of the major areas in this appendix are listed below.

F.1 TI Memories

This section provides tables of product information for bipolar memories, MOS EPROM memories, and fast CMOS EPROM memories. The table of bipolar memories was taken from the *ALS/AS Logic Data Book*. The MOS EPROM selection guide is copied from the *MOS Memory Data Book*.

All of these memories can be interfaced with TMS320C1x devices. Refer to *Digital Signal Processing Applications with the TMS320 Family* for memory interfaces and to the product data sheets for specification information.

BIPOLAR MEMORIES

USER-PROGRAMMABLE READ-ONLY MEMORIES (PROMs)
STANDARD PROMs

DESCRIPTION	TYPE	ORGANIZATION	TYPE OUTPUT	S	VOLUME
16K Bit Arrays	TBP28S166	2048W · 8B	3-State	●	4
	TBP38S165	2048W · 8B	3-State	●	
	TBP38S166	2048W · 8B	3-State	●	
	TBP38SA165	2048W · 8B	OC	●	
	TBP38SA166	2048W · 8B	OC	●	
	TBP34S162	4096W · 4B	3-State	●	
	TBP34SA162	4096W · 4B	OC	▲	
8K Bit Arrays	TBP24S81	2048W · 4B	3-State	●	
	TBP24SA81	2048W · 4B	OC	●	
	TBP28S85A	1024W · 8B	3-State	▲	
	TBP28S86A	1024W · 8B	3-State	●	
	TBP28SA86A	1024W · 8B	OC	●	
	TBP38S85	1024W × 8B	3-State	▲	
	TBP38S86	1024W × 8B	3-State	▲	
	TBP38SA85	1024W × 8B	OC	▲	
	TBP38SA86	1024W × 8B	OC	▲	
4K Bit Arrays	TBP24S41	1024W · 4B	3-State	●	
	TBP24SA41	1024W · 4B	OC	●	
	TBP28S42	512W · 8B	3-State	●	
	TBP28SA42	512W · 8B	OC	●	
	TBP28S46	512W · 8B	3-State	●	
	TBP28SA46	512W · 8B	OC	●	
2K Bit Arrays	TBP38S22	256W × 8B	3-State	●	
	TBP38SA22	256W × 8B	OC	●	
1K Bit Arrays	TBP24S10	256W · 4B	3-State	●	
	TBP24SA10	256W · 4B	OC	●	
	TBP34S10	256W × 4B	3-State	●	
	TBP34SA10	256W × 4B	OC	●	
256 Bit Arrays	TBP18S030	32W · 8B	3-State	●	
	TBP18SA030	32W · 8B	OC	●	
	TBP38S030	32W · 8B	3-State	●	
	TBP38SA030	32W · 8B	OC	●	

LOW-POWER PROMs

DESCRIPTION	TYPE	ORGANIZATION	TYPE OUTPUT	S	VOLUME
16K Bit Arrays	TBP28L166	2048W × 8B	3-State	●	4
	TBP38L165	2048W × 8B	3-State	●	
	TBP38L166	2048W × 8B	3-State	●	
	TBP34L162	4096W × 4B	3-State	▲	
8K Bit Arrays	TBP28L85A	1024W × 8B	3-State	▲	
	TBP28L86A	1024W × 8B	3-State	●	
	TBP38L85	1024W × 8B	3-State	▲	
	TBP38L86	1024W × 8B	3-State	▲	
4K Bit Arrays	TBP28L42	512W × 8B	3-State	●	
	TBP28L46	512W × 8B	3-State	●	
2K Bit Arrays	TBP28L22	256W · 8B	3-State	●	
	TBP28LA22	256W · 8B	OC	●	
	TBP38L22	256W · 8B	3-State	●	
1K Bit Arrays	TBP34L10	256W × 4B	3-State	●	
256 Bit Arrays	TBP38L030	32W × 8B	3-State	●	

REGISTERED PROMs

DESCRIPTION	TYPE	ORGANIZATION	TYPE OUTPUT	S	VOLUME
16K Bit Arrays	TBP34R162	4096W × 4B	3-State	●	4
	TBP34SR165	4096W × 4B	3-State	●	
	TBP38R165	2048W × 8B	3-State	●	

RANDOM-ACCESS READ-WRITE MEMORIES (RAMs)

DESCRIPTION	ORGANIZATION	TYPE OF OUTPUT	TYPE	TECHNOLOGY					VOLUME
				STD TTL	ALS	AS	LS	S	
256 Bit Arrays	256 · 1	3-State	201					●	4
		OC	301					●	
64 Bit Arrays	16 · 4	OC	89	●					
		3-State	189				A	B	
		3-State	219				A		
		OC	289				A	B	
		OC	319				A		
16-Bit Multiple Port Register File	8 · 2	3-State	172	●					2
16 Bit Register File	4 · 4	OC	170	●			●		
		3-State	670				●		
Dual 64-Bit Register Files	16 · 4	3-State	870			●			3
			871			●			

FIRST-IN FIRST-OUT MEMORIES (FIFOs)

DESCRIPTION	TYPE OF OUTPUT	TYPE	TECHNOLOGY				VOLUME
			ALS	AS	LS	S	
16 · 4	3-State	222			●		LSI
	3-State	224			●		
	3-State	227			●		
	3-State	228			●		
	3-State	232	A				3 & LSI
16 · 5	3-State	225			●	●	LSI
	3-State	229	A				3 & LSI
	3-State	233	A				

● Denotes available technology
▲ Denotes planned new products
A Denotes "A" suffix version available in the technology indicated
B Denotes "B" suffix version available in the technology indicated

MOS EPROM SELECTION GUIDE

Density	Device Number	Organization	Process	Access Time Max (ns)	Cycle Time Min (ns)	Power Supply/ Tolerance (V)	Power Dissipation Max (mW)		Pins	Package†	Page
							Active	Standby			
32K	TMS2732A-17	4K X 8	NMOS	170	170	5 ± 5%	657	158	24	J	6-3
	TMS2732A-20			200	200						
	TMS2732A-25			250	250						
	TMS2732A-45			450	450						
64K	TMS2764-17	8K X 8	NMOS	170	170	5 ± 5%	788	184	28	J	6-11
	TMS2764-20			200	200						
	TMS2764-25			250	250						
	TMS2764-45			450	450						
64K	TMS27C64-1‡	8K X 8	CMOS	150	150	5 ± 5%	210	1.4	28	J	6-21
	TMS27C64-15‡			150	150	5 ± 10%	220				
	TMS27C64-2‡			200	200	5 ± 5%	210				
	TMS27C64-20‡			200	200	5 ± 10%	220				
	TMS27C64‡			250	250	5 ± 5%	210				
	TMS27C64-25‡			250	250	5 ± 10%	220				
	TMS27C64-3‡			300	300	5 ± 5%	210				
	TMS27C64-30‡			300	300	5 ± 10%	220				
	TMS27C64-4‡			450	450	5 ± 5%	210				
	TMS27C64-45‡			450	450	5 ± 10%	220				
128K	TMS27C128-1	16K X 8	CMOS	150	150	5 ± 5%	210	1.4	28	J	6-29
	TMS27C128-15			150	150	5 ± 10%	220				
	TMS27C128-2			200	200	5 ± 5%	210				
	TMS27C128-20			200	200	5 ± 10%	220				
	TMS27C128			250	250	5 ± 5%	210				
	TMS27C128-25			250	250	5 ± 10%	220				
	TMS27C128-3			300	300	5 ± 5%	210				
	TMS27C128-30			300	300	5 ± 10%	220				
	TMS27C128-4			450	450	5 ± 5%	210				
	TMS27C128-45			450	450	5 ± 10%	220				
256K	TMS27C256-1	32K X 8	CMOS	170	170	5 ± 5%	210	1.4	28	J	6-37
	TMS27C256-17			170	170	5 ± 10%	220				
	TMS27C256-2			200	200	5 ± 5%	210				
	TMS27C256-20			200	200	5 ± 10%	220				
	TMS27C256			250	250	5 ± 5%	210				
	TMS27C256-25			250	250	5 ± 10%	220				
	TMS27C256-3			300	300	5 ± 5%	210				
	TMS27C256-30			300	300	5 ± 10%	220				
	TMS27C256-4			450	450	5 ± 5%	210				
	TMS27C256-45			450	450	5 ± 10%	220				

†J = Ceramic DIP

‡Advance information for product under development by TI.

FAST CMOS EPROM SELECTION GUIDE

Density	Device Number	Organization	Process	Access Time Max (ns)	Cycle Time Min (ns)	Power Supply/ Tolerance (V)	Power Dissipation Max (mW)		Pins	Package†
							Active	Standby		
16K	TMS27C292-3	2K X 8	CMOS	35	35	5 ± 5%	374		24	J
	TMS27C292-35			35	35	5 ± 10%	374			
	TMS27C292-5			50	50	5 ± 5%	374			
	TMS27C292-50			50	50	5 ± 10%	374			

†J = Ceramic DIP

F.2 TI Peripherals

The tables of product information for peripherals include programmable logic arrays, codecs, filters, combos, and A/D and D/A converters. The table of programmable logic arrays can be found in the *ALS/AS Logic Data Book*. The table on codecs, filters, and combos is taken from the *Telecom Data Book*. The A/D and D/A converter tables are located in the *Master Selection Guide*.

These peripherals can be used with any of the TMS320C1x devices. Refer to the book, *Digital Signal Processing Applications with the TMS320 Family*, for peripheral interfaces and to the product data sheets for specification information.

PROGRAMMABLE LOGIC ARRAYS

DESCRIPTION	INPUTS	OUTPUTS NO.	OUTPUTS TYPE	TYPE NO	ALS	NO. OF PINS	VOLUME
High-Performance PAL®	16	8	Active-Low	'PAL16L8A	•	20	4
		4	Registered	'PAL16R4A	•		
		6		'PAL16R6A	•		
		8		'PAL16R8A	•		
Half-Power PAL®	16	8	Active-Low	'PAL16L8A-2	•	20	
		4	Registered	'PAL16R4A-2	•		
		6		'PAL16R6A-2	•		
		8		'PAL16R8A-2	•		
High-Performance PAL®	20	8	Active-Low	'PAL20L8A	•	24	
		4	Registered	'PAL20R4A	•		
		6		'PAL20R6A	•		
		8		'PAL20R8A	•		
Half-Power PAL®	20	8	Active-Low	'PAL20L8A-2	•	24	
		4	Registered	'PAL20R4A-2	•		
		6		'PAL20R6A-2	•		
		8		'PAL20R8A-2	•		
Impact PAL®	16	8	Active-Low	'TIBPAL16L8-12	•	20	
		4	Registered	'TIBPAL16R4-12	•		
		6		'TIBPAL16R6-12	•		
		8		'TIBPAL16R8-12	•		
Impact PAL®	16	8	Active-Low	'TIBPAL16L8-15	•	20	
		4	Registered	'TIBPAL16R4-15	•		
		6		'TIBPAL16R6-15	•		
		8		'TIBPAL16R8-15	•		
Impact PAL®	20	8	Active-Low	'TIBPAL20L8-15	•	24	
		4	Registered	'TIBPAL20R4-15	•		
		6		'TIBPAL20R6-15	•		
		8		'TIBPAL20R8-15	•		
Exclusive-OR PAL®	20	10	Active-Low	'TIBPAL20L10-20	•	24	
		4	Registered	'TIBPAL20X4-20	•		
		8		'TIBPAL20X8-20	•		
		10		'TIBPAL20X10-20	•		
Exclusive-OR PAL®	20	8	Active-Low	'TIBPAL20L10-35	•	24	
		4	Registered	'TIBPAL20X4-35	•		
		8		'TIBPAL20X8-35	•		
		10		'TIBPAL20X10-35	•		
Registered-Input PAL®	19	8	Active-Low	'TIBPALR19L8-25	•	24	
		4	Registered	'TIBPALR19R4-25	•		
		6		'TIBPALR19R6-25	•		
		8		'TIBPALR19R8-25	•		
Registered-Input PAL®	19	8	Active-Low	'TIBPALR19L8-40	•	24	
		4	Registered	'TIBPALR19R4-40	•		
		6		'TIBPALR19R6-40	•		
		8		'TIBPALR19R8-40	•		
Latched-Input PAL®	19	8	Active Low	'TIBPALT19L8-25	•	24	
		4	Registered	'TIBPALT19R4-25	•		
		6		'TIBPALT19R6-25	•		
		8		'TIBPALT19R8-25	•		
Latched-Input PAL®	19	8	Active Low	'TIBPALT19L8-40	•	24	
		4	Registered	'TIBPALT19R4-40	•		
		6		'TIBPALT19R6-40	•		
		8		'TIBPALT19R8-40	•		
Field-Programmable 14 × 32 × 6 Logic Arrays	14	6	3-State	'TIFPLA839	•	24	
			OC	'TIFPLA840	•		

® PAL is a registered trademark of Monolithic Memories Incorporated.

• Denotes available technology.

CODECS, FILTERS, COMBOS

DESCRIPTION	DEVICE NUMBER	PROCESS PACKAGE	SUPPLY VOLTAGE	PAGE
PCM Codec – μ-Law	TCM2909	NMOS 22-Pin J	12 V, ±5 V	2-55
PCM Codec – μ-Law	TCM2910A	NMOS 24-Pin J	12 V, ±5 V	2-55
PCM Line Filter	TCM2912B	NMOS 16-Pin J	±5 V	2-75
PCM Line Filter	TCM2912C	NMOS 16-Pin J	±5 V	2-75
Synchronous	TCM2913	NMOS 20-Pin J	±5 V	2-89
Synchronous/Asynchronous	TCM2914	NMOS 24-Pin J 28-Pin FN	±5 V	2-89
μ-Law	TCM2916	NMOS 16-Pin J	±5 V	2-89
A-Law	TCM2917	NMOS 16-Pin J	±5 V	2-89
Synchronous	TCM29C13	CMOS 20-Pin J	±5 V	2-111
Synchronous/Asynchronous	TCM29C14	CMOS 24-Pin J 28-Pin FN	±5 V	2-111
μ-Law	TCM29C16	CMOS 16-Pin J	±5 V	2-111
A-Law	TCM29C17	CMOS 16-Pin J	±5 V	2-111
DSP Combined Codec/Filter				
μ-Law	TCM29C18	CMOS 16-Pin N	±5 V	–
	TCM29C19	CMOS 16-Pin N	±5 V	–
Linear Codec/Filter				
(14-Bit A/D, 14-Bit D/A)	TLC32040	CMOS 28-Pin N	±5 V	–

SUCCESSIVE-APPROXIMATION A/D CONVERTERS

TYPE	SIGNAL INPUTS		RE-SOLU-TION	UNADJUSTED ERROR (MAX) +/−LSB	ADDRESS AND DATA I/O FORMAT	CONVERSION SPEED@ (μS)	POWER DISSIPATION (TYP) (mW)	TEMP* RANGE	PACKAGE	DOCUMENT
	DEDICATED ANALOG	ANALOG/ DIGITAL**								
ADC0803	1	0	8-Bit	0.5	Parallel	100	10	I	N	SLAS004
ADC0804				1				C		SLYD001
ADC0805								I		SLAS004
ADC0808	8			0.75				I	FN,N	SLYD001
ADC0809				1.25						
ADC0831A	1			1	Serial	32		C,I	P	SLAS006
ADC0831B				0.5						
ADC0832A	2			1						
ADC0832B				0.5						
ADC0834A	4			1					N	SLAS007
ADC0834B				0.5						
ADC0838A	8			1					FN,N	
ADC0838B				0.5						
TL0808				1	Parallel	100	0.5	I	FN,N	SLAS001
TL0809										
TL0820A	1			0.75		1	35	C,I,M	FN,N	TBA
TL0820B				1						
TL532A	5	6		0.5		15	6	I,M		
TL533A						29				
TLC540	11	0			Serial	13				SLNS004A
TLC541						25				SLNS011
TLC543	5					13.3			D,N	TBA
TLC544						25				
TLC545	19					13			FN,N	SLNS011
TLC546						25				
TLC548	1					22			D,P	SLNS009
TLC549						25				
TLC1540	11					16			FN,N	SLNS010
TLC1541						31				

* M = −55°C to 125°C, I = −40°C to 85°C, C = 0°C to 70°C.

** Analog/digital inputs can be used either as digital inputs for limiting sensing or digital, or can be used as analog inputs. For example: The TLC532/3A can have 11 analog, inputs and 6 digital outputs, 5 analog inputs and 6 digital inputs, or any combination in between.

@ Includes access time.

D/A CONVERTERS

TYPE	FUNCTION	RESOLUTION	SETTLING TIME (ns)	VOLTAGE RANGE (V)	TEMP* RANGE	PACKAGE	DOCUMENT
TLC7524	Single Multiplying DAC	8-Bits	100	5 to 15	C	N	TBA
TLC7528	Dual Multiplying DAC				I		

* M = −55°C to 125°C, I = −40°C to 85°C, C = 0°C to 70°C.

F.3 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.

IC SOCKETS
DUAL-IN-LINE

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

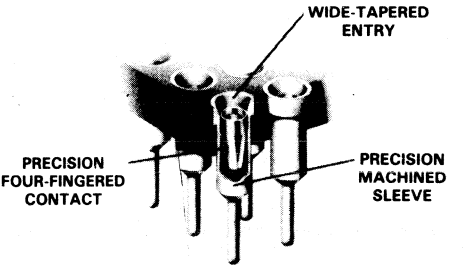
Environmental

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 — clip 30 μin gold over 50 μin nickel or 50 μin tin/lead over 50 μin nickel
Specified by Part Number — 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

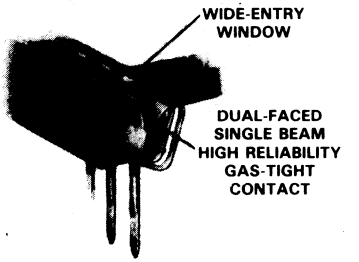


C7X SERIES PART NUMBER SYSTEM

C7X (X) XX – X X

- C7X: Screw Machine Socket
  - 1 — wire wrap
  - 2 — solder tail
- (X): S — Single-in-line package (where applicable)
- XX: Number of Positions
- X: Plating (Sleeve/Clip)
  - 0 — Gold/Gold
  - 5 — Tin/Gold
- X: Variations
  - Solder Tail
    - 9 — Pin length 0.105/0.150
  - Wire Wrap
    - 3 — Pin length 0.510

C86 SERIES — STAMPED AND FORMED



C86 SERIES PART NUMBER SYSTEM

C 86 XX – 01

- C: TI Socket Series
- 86: Tin Dual Face Wipe Single Beam
- XX: Number of positions
- 01: Variation
  - 01 — Standard product

IC SOCKETS
BURN-IN/TEST DIP

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

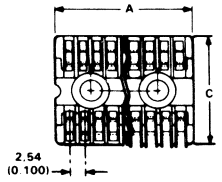
†For additional plating options consult the factory

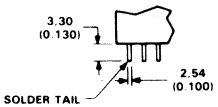
PART NUMBER SYSTEM

C X 37 XX – 22 S X

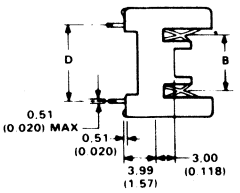
- X: Pin to pin — A—0.100 centers; B—0.070 centers
- S: PPS high temperature body material
- 22: Copper nickel alloy Soldertail
- XX: Number of positions
- 37: Overall gold plate
- X: Series Features — Q—Auto unloadable; P—High density mounting; M—Shrink 0.070 centers
- C: TI Socket Series

BURN-IN/TEST DIP SOCKETS

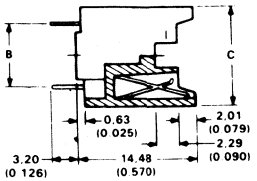




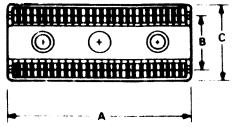
CQ37 SERIES

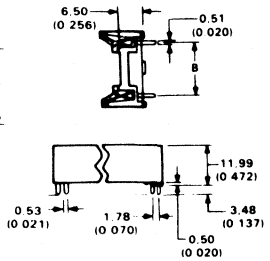


CP37 SERIES



CM37 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

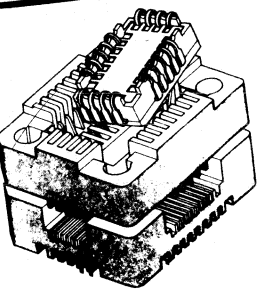
Number of Positions	A max Length	B ±0.02	C max Width
8	11,68 (0.460)	7,62 (0.300)	12,70 (0.500)
14	17,78 (0.700)		
16	20,32 (0.800)		
18	22,86 (0.900)		
20	25,40 (1.000)		
24	30,48 (1.200)	15,24 (0.600)	20,32 (0.800)
28	35,56 (1.400)		
40	50,80 (2.000)		

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

IC SOCKETS
PLCC BURN-IN/TEST



JRES

- op actuated insertion or press-in
- manually or automatically
- to high pressure contact point
- n stand-off design provide high efficiency
- on
- to 10,000 cycles

E SPECIFICATIONS

-)0 cycles
- erature: 180°C max

- 1.0 A per contact
- ce: 30 mΩ max
- nce: 1000 MΩ min
- ding voltage: 500 V ac rms min

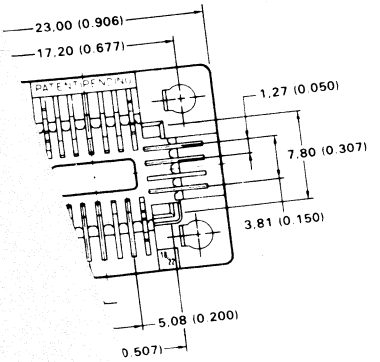
- s filled (U/L 94 VO)
- lloy
- d plate

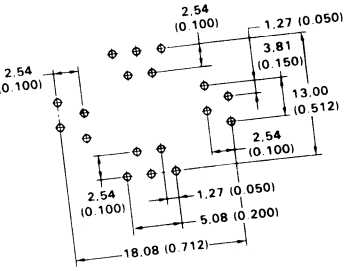
PART NUMBER SYSTEM

CPJ AA33A – XXX B

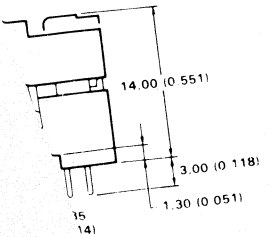
- XXX: Number of positions
- CPJ: TI series socket

T SOCKETS CPJ SERIES





SIZES: 18 PIN
22 PIN



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

ntact factory for detailed information

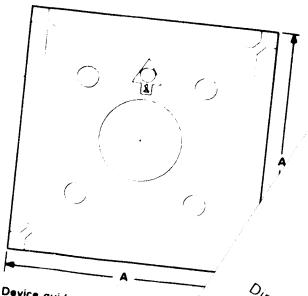
EASILY AUTO INSERTED

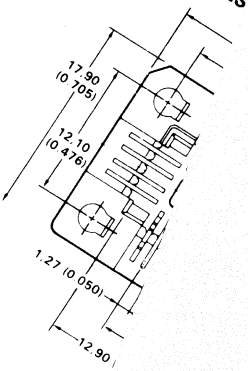
Hig
Com
PERFO
Mechani
Durability:
Operating Te
Electrical
Contact rating:
Contact resistan
Insulation resistan
Dielectric withstan
MATERIALS
Body – ultem glas
Contact – copper
Plating – overall go

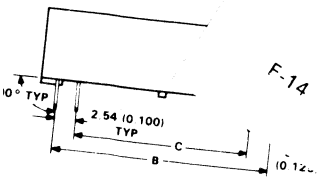
PLCC BURN-IN/TES

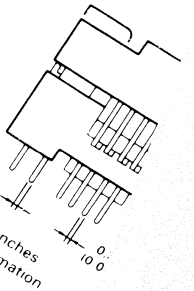
PART
CF

ASTIC LEADER CHIP CARRIER CPR SERIES









Dimensions in parentheses are inches
Contact factory for detailed information

stals

This section lists the commonly used crystal frequencies, crystal specification requirements, and the names of suitable vendors.

Table F-1 lists the commonly used crystal frequencies and the devices with which they can be used.

Table F-1. Commonly Used Crystal Frequencies

FREQUENCY	DEVICE
16.000 MHz	TMS32010-16
18.432 MHz 20.000 MHz 20.480 MHz	TMS32010/C10, TMS320C15/E15, TMS320C17/E17
25.000 MHz	TMS32010/C10, TMS320C15, TMS320C17-25

A crystal connected across X1 and X2/CLKIN on the TMS320 processor enables the internal oscillator, as shown in Figure F-1. The frequency of CLKOUT is one-fourth the crystal fundamental frequency. Crystal specification requirements are listed below.

Load capacitance = 20 pF
Series resistance = 30 ohm
Power dissipation = 1 mW
Parallel resonant
16-MHz and 20-MHz crystals use fundamental mode.
25-MHz operation may require third-overtone crystal.

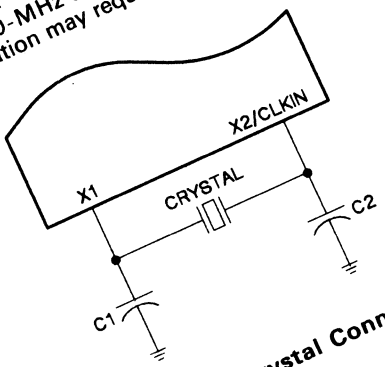


Figure F-1. Crystal Connection

Appendix F - Crystals

Vendors of crystals suitable for use with TMS320 devices are listed

RXD, Inc.
Norfolk, NB
(800) 228-8108

CTS Knight, Inc.
Contact the local distributor

N.E.L. Frequency Cont
Burlington, WI
(414) 763-3591

Programming the TMS320E15/E17 EPROM Cell

The TMS320E15/E17 includes a 4K x 16-bit EPROM implemented using an industry-standard EPROM cell for prototyping, early field testing, and production. The TMS320C15/C17 with a 4K-word masked ROM then provides a migration path for cost-effective production. An EPROM adaptor socket (part # RTC/PGM320A-06), shown in Figure G-1, is available to provide 40-pin to 28-pin conversion for programming the TMS320E15/E17.

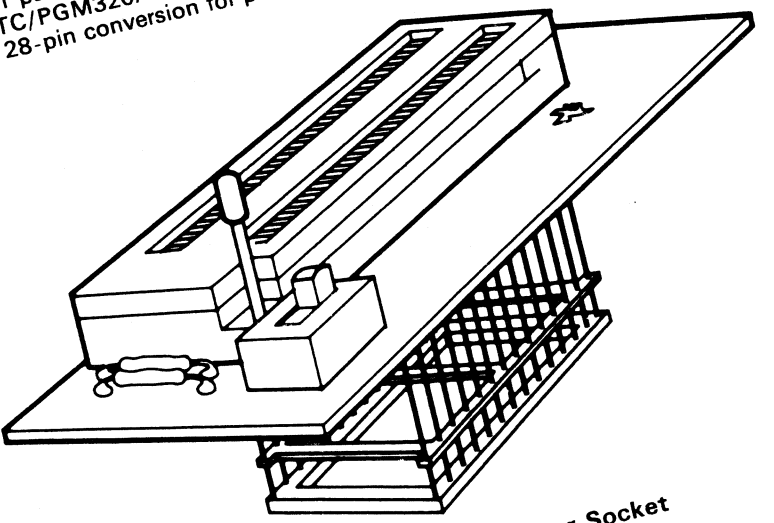


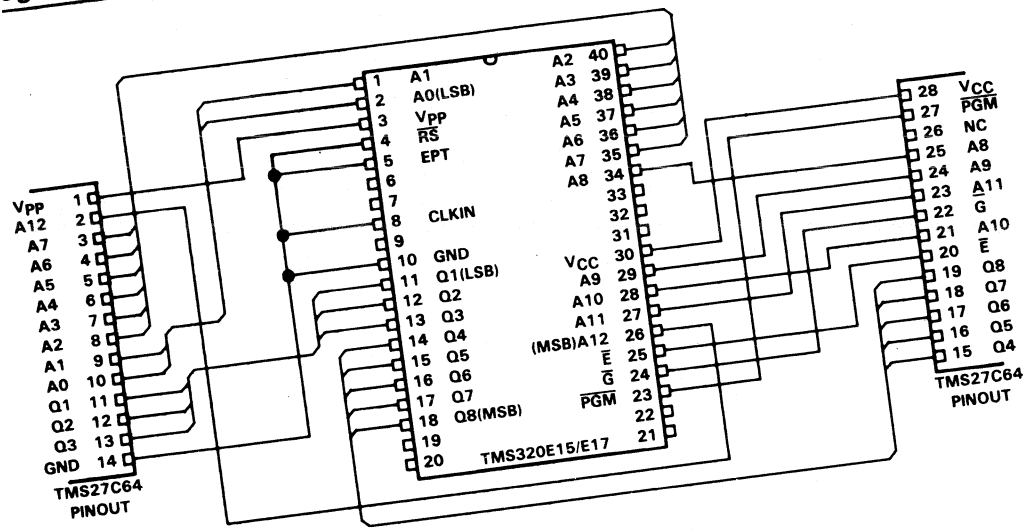
Figure G-1. EPROM Adaptor Socket

Key features of the EPROM cell include standard programming and verifi
tion. The EPROM cell also includes a code protection feature that allows c
to be protected against copyright violations. The protection feature ca
used to protect reading the EPROM contents. This appendix describes era
fast programming and verification, and ROM protection and verification

G.1 Fast Programming and Verification

The TMS320E15/E17 EPROM cell is programmed using the same family
device codes as the TMS27C64 8K x 8-bit EPROM. The TMS27C64 EPR
series are ultraviolet-light erasable, electrically programmable read-only me
ories, fabricated using HVCMOS technology. The TMS27C64 is pin-comp
tible with existing 28-pin ROMs and EPROMs. The TMS320E15/E17, like th
TMS27C64, operates from a single 5-V supply in the read mode; however,
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. When
programmed in blocks, the data is loaded into the EPROM cell one byte at a
time, the high byte first and the low byte second.

Figure G-2 shows the wiring conversion to program the TMS320E15/E17
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. The TMS320E15/E17 uses 13 ad-
dress lines to address the 4K-word memory in byte format.



PIN NOMENCLATURE (TMS320E15/E17)

SIGNAL	I/O	DEFINITION
A12(MSB)-A0(LSB)	I	On-chip EPROM programming address lines
CLKIN	I	Clock oscillator input
$\overline{E}$	I	EPROM chip select
EPT	I	EPROM test mode select
$\overline{G}$	I	EPROM read/verify select
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 power supply
V_{PP}	I	12.5-V power supply

Figure G-2. TMS320E15/E17 EPROM Conversion to TMS27C64 EPROM Pinout

Programming TMS320E15/E17 EPROM – Fast Programming/Verification

Table G-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 G-1. TMS320E15/E17 Programming Mode Levels

SIGNAL NAME	TMS320E15/ E17 PIN	TMS27C64 PIN	PROGRAM	PROGRAM VERIFY	PROGRAM INHIBIT	READ	OUTPUT DISABLE
$\overline{E}$	25	20	V_{IL}	V_{IL}	V_{IH}	V_{IL}	V_{IL}
$\overline{G}$	24	22	V_{IH}	$\overline{PULSE}$	X	$\overline{PULSE}$	V_{IH}
$\overline{PGM}$	23	27	$\overline{PULSE}$	V_{IH}	X	V_{IH}	V_{IH}
V_{PP}	3	1	V_{PP}	V_{PP}	V_{PP}	V_{CC}	V_{CC}
V_{CC}	30	28	$V_{CC}+1$	$V_{CC}+1$	$V_{CC}+1$	V_{CC}	V_{CC}
V_{SS}	10	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
CLKIN	8	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
$\overline{RS}$	4	14	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
EPT	5	26	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
Q8-Q1	18-11	19-15,13-11	D_{IN}	Q_{OUT}	HI-Z	Q_{OUT}	HI-Z
A12-A10	26-28	2,23,21	ADDR	ADDR	X	ADDR	X
A9-A7	29,34,35	24,25,3	ADDR	ADDR	X	ADDR	X
A6	36	4	ADDR	ADDR	X	ADDR	X
A5	37	5	ADDR	ADDR	X	ADDR	X
A4	38	6	ADDR	ADDR	X	ADDR	X
A3-A0	39,40,1,2	7-10	ADDR	ADDR	X	ADDR	X

LEGEND:
V_{IH} = TTL high level; V_{IL} = TTL low level; ADDR = byte address bit
V_{PP} = 12.5 ± 0.5 V; V_{CC} = 5 ± 0.25 V; X = don't care
$\overline{PULSE}$ = low-going TTL level pulse; D_{IN} = byte to be programmed at ADDR
Q_{OUT} = byte stored at ADDR; RBIT = ROM protect bit.

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 TMS320E15/E17, the window should be covered with an opaque label.

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 G-3, 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 TMS320E15/E17 can be programmed when the devices are connected in parallel. Locations can be programmed in any order.

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 15 times. The final programming pulse is 4 ms 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.

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 G-4 shows the timing for the program and verify operation.

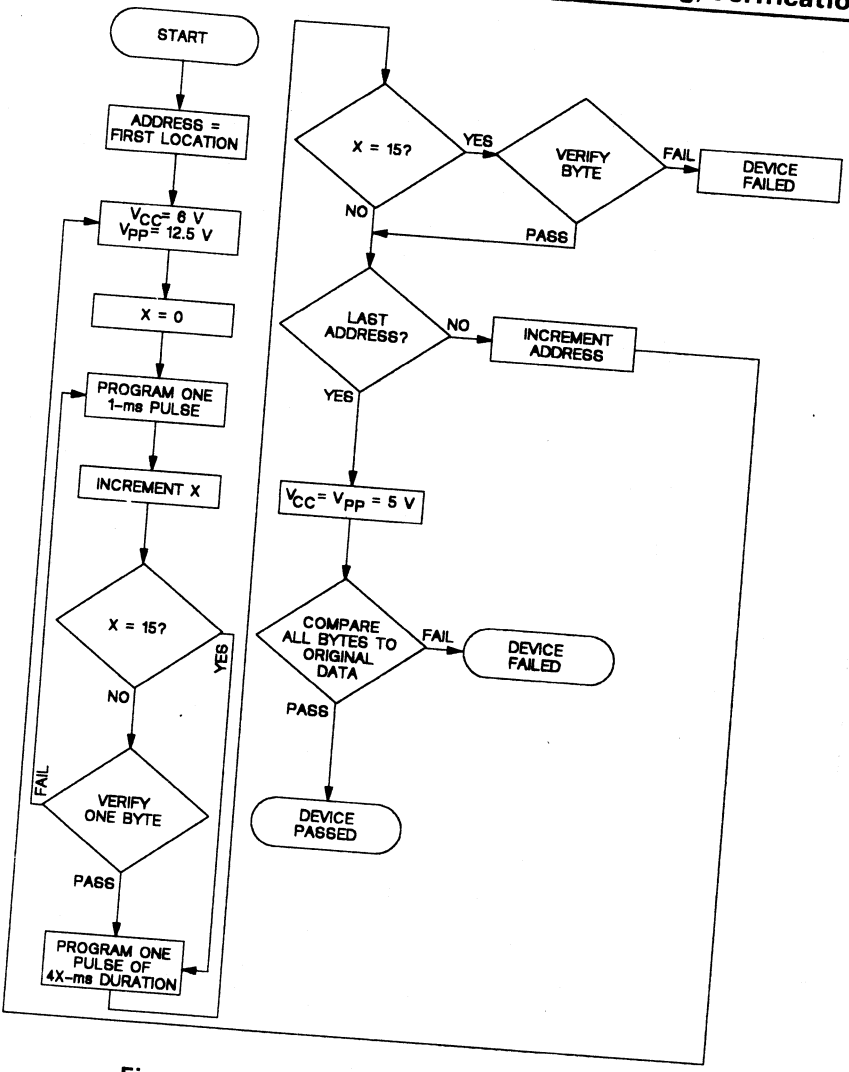


Figure G-3. Fast Programming Flowchart

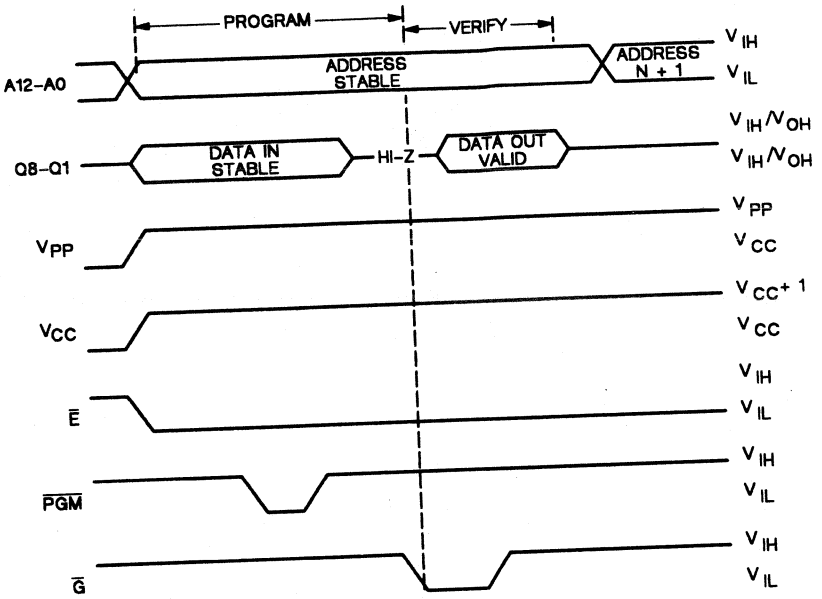


Figure G-4. Fast Programming Timing

Program Inhibit

Programming may be inhibited by maintaining a high level input on the $\overline{E}$ pin or $\overline{PGM}$ pin.

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.

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 $\overline{G}$ and $\overline{PGM}$ pins high. While output disable is selected, Q8-Q1 are placed in the high-impedance state.

G.2 ROM Protection and Verification

This section describes the code protection feature included in the EPROM cell, which protects code against copyright violations. Table G-2 shows the programming levels required for protecting and verifying the ROM. The paragraphs following the table describe the protect and verify functions.

Table G-2. TMS320E15/E17 Protect and Verify ROM Mode Levels

SIGNAL	TMS320E15/E17 PIN	TMS27C64 PIN	ROM PROTECT	PROTECT VERIFY
$\overline{E}$	25	20	V_{IH}	V_{IL}
$\overline{G}$	24	22	V_{IH}	V_{IL}
$\overline{PGM}$	23	27	V_{IH}	V_{IH}
V_{PP}	3	1	V_{PP}	V_{CC}
V_{CC}	30	28	$V_{CC}+1$	V_{CC}
V_{SS}	10	14	V_{SS}	V_{SS}
CLKIN	8	14	V_{SS}	V_{SS}
$\overline{RS}$	4	14	V_{SS}	V_{SS}
EPT	5	26	V_{PP}	V_{PP}
Q8-Q1	18-11	19-15,13-11	Q8=$\overline{PULSE}$	Q8=RBIT
A12-A10	26-28	2,23,21	X	X
A9-A7	29,34,35	24,25,3	X	X
A6	36	4	X	V_{IL}
A5	37	5	X	X
A4	38	6	V_{IH}	X
A3-A0	39,40,1,2	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; X = don't care
$\overline{PULSE}$ = low-going TTL level pulse; RBIT = ROM protect bit

ROM Protect

The ROM protect 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 ROM protect 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 G-5. The required setups in the figure are detailed in Table G-2.

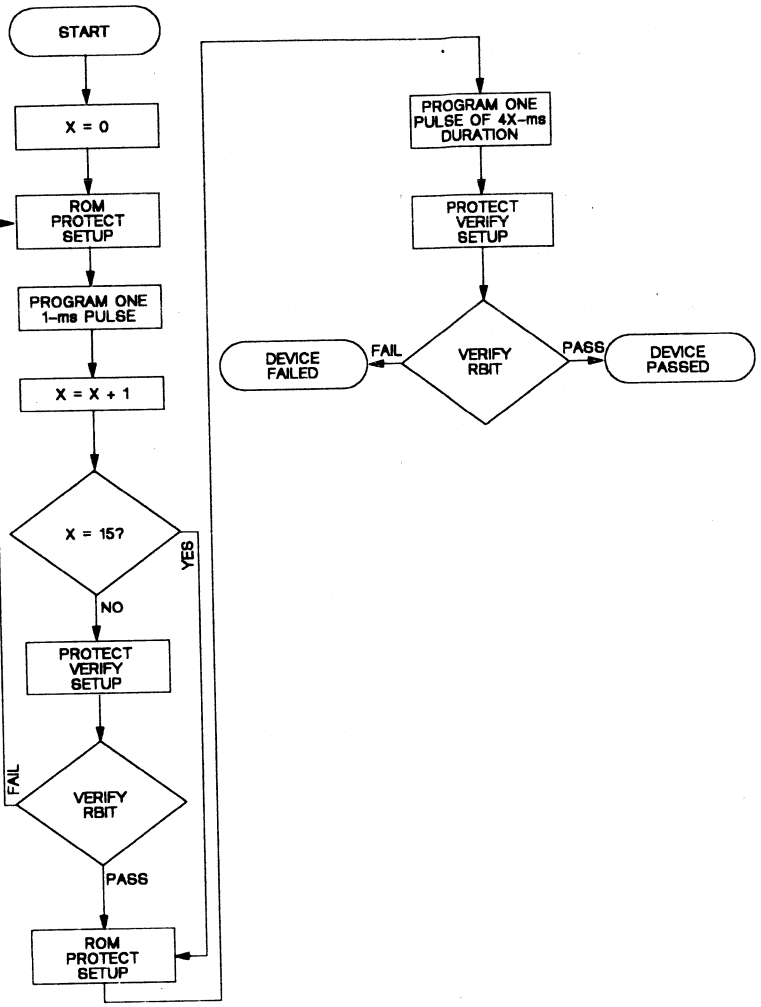


Figure G-5. ROM Protect Flowchart

Protect Verify

Protect verify is used following the ROM protect to verify correct programming of the RBIT (see Figure G-5). 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 ROM protect and verify timings are shown in Figure G-6.

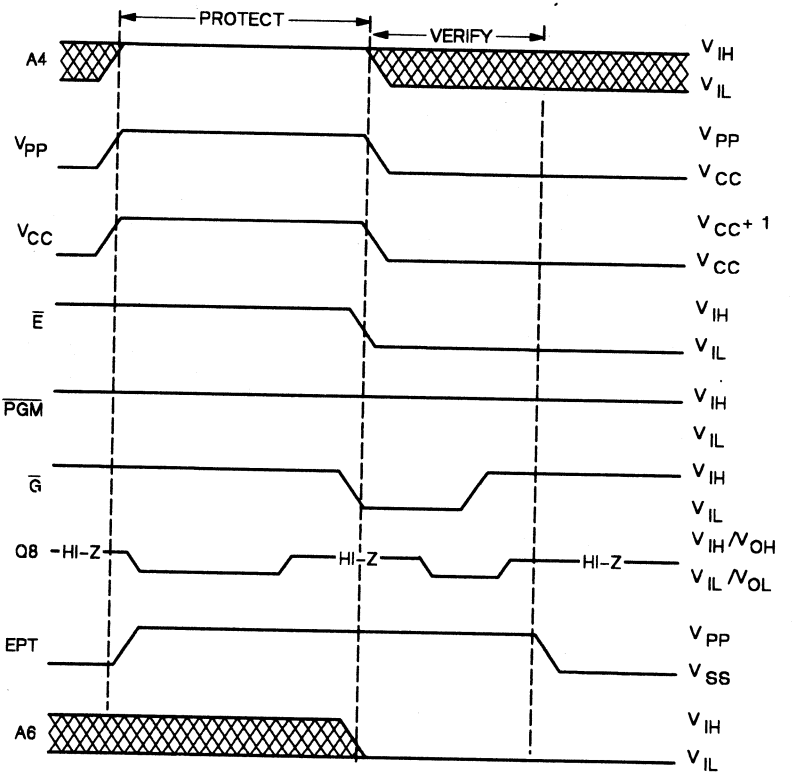


Figure G-6. ROM Protect Timing

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

V

X

Z

TI Sales Offices

BELGIQUE/BELGIË
S.A. Texas Instruments Belgium N.V.
11, Avenue Jules Bordetlaan 11,
1140 Bruxelles/Brussel
Tel : (02) 242 30 80
Telex : 61161 TEXBEL

DANMARK
Texas Instruments A/S
Marielundvej 46E
2730 Herlev
Tel : 02 91 74 00
Telefax : 02 91 84 00
Telex : 35123 TEXIN

DEUTSCHLAND
Texas Instruments
Deutschland GmbH.
Haggertystraße 1
8050 Freising
Tel : 0 81 61/80-0 od. Nbst
Telex : 5 26 529 texin d
Btx : *280505#

Kurfürstendamm 195-196
1000 Berlin 15
Tel : 030/8 82 73 65
Telex : 5 26 529 texin d

Düsseldorfer Straße 40
6236 Eschborn 1
Tel : 06196/80 70
Telex : 5 26 529 texin d

Ill. Hagen 43/Kibbelstraße 19
4300 Essen 1
Tel : 0201/24 25-0
Telex 5 26 529 texin d

Kirchhorster Straße 2
3000 Hannover 51
Tel : 0511/64 80 21
Telex : 5 26 529 texin d

Maybachstraße II
7302 Ostfildern 2 (Nellingen)
Tel. 0711/34 03-0
Telex 5 26 529 texin d

EIRE
Texas Instruments Ireland Ltd
7/8 Harcourt Street
Dublin 2
Tel : (01) 78 16 77
Telex : 32626

ESPAÑA
Texas Instruments España S.A.
C/José Lázaro Galdiano No. 6
28036 Madrid
Tel : (1) 458 14 58
Telex : 23634
Fax : (1) 457 94 04

C/Diputacion, 279-3-5
08007 Barcelona
Tel : (3) 317 91 80
Telex : 50436
Fax : (3) 301 84 61

FRANCE
Texas Instruments France
8-10 Avenue Morane Saulnier - B.P. 67
78141 Vélizy Villacoublay cedex
Tel : Standard : (1) 30 70 10 03
Service Technique : (1) 30 70 11 33
Telex : 698707 F

HOLLAND
Texas Instruments Holland B.V.
Hogehilweg 19
Postbus 12995
1100 AZ Amsterdam-Zuidoost
Tel : (020) 5602911
Telex : 12196

ITALIA
Texas Instruments Italia S.p.A.
Divisione Semiconduttori
Viale Europa, 40
20093 Cologno Monzese (Milano)
Tel : (02) 25300 1
Telex : 332633 MITEX I

Via Castello della Magliana, 38
00148 Roma
Tel. (06) 5222651
Telex 610587 ROTEX I

Corso Svizzera, 185
10100 Torino
Tel. (011) 774545

Via Amendola, 17
40100 Bologna
Tel. (051) 554004

NORGE
Texas Instruments Norge A/S
PB 106
Refstad (Sinsenveien 53)
0585 Oslo 5
Tel : (02) 155090

ÖSTERREICH
Texas Instruments Ges.m.b.H.
Hietzinger Kai 101-105
A-1130 Wien
Tel : 0222/9100-0
Telex : 136 796

PORTUGAL
Texas Instruments Equipamento
Electronico (Portugal) LDA.
R. Eng. Frederico Ulrich, 2650
Moreira Da Maia
4470 Maia
Tel : (2) 948 1003
Telex : 22485

SCHWEIZ/SUISSE
Texas Instruments Switzerland AG
Riedstraße 6
CH-8953 Dietikon
Tel : (01) 740 22 20
Telex : 825 260 TEXIN

SUOMI FINLAND
Texas Instruments OY
Aherrajantie 3
P.O. Box 81,
0201 Espoo
Tel : (90) 0-461-422
Telex : 121457

SVERIGE
Texas Instruments
International Trade Corporation
(Sverigefilialen)
Box 30,
S-163 93 Stockholm
Visit address : Isafjordsgatan 7, Kista
Tel : (08) 793 91 70
Telefax : (08) 751 97 15
Telex : 10377 SVENTEX S

UNITED KINGDOM
Texas Instruments Ltd.
Manton Lane,
Bedford,
England, MK41 7PA
Tel : (0234) 270 111
Telex : 82178

Technical Enquiry Service
Tel : (0234) 223000

TEXAS INSTRUMENTS

TI Regional Technology Centres

DEUTSCHLAND
Texas Instruments
Deutschland GmbH.
Haggertystraße 1
8050 Freising
Tel : (08161) 80 40 43

Frankfurt/Main
Düsseldorfer Straße 40
6236 Eschborn
Tel : (0 61 96) 80 74 18

Kirchhorster Straße 2
3000 Hannover 51
Tel : (0511) 64 80 21

Maybachstraße 11
7302 Ostfildern 2 (Nellingen)
Stuttgart
Tel : (0711) 34 03-0

FRANCE
Centre de Technologie
Texas Instruments France
8-10 Avenue Morane Saulnier, B.P. 67
78141 Vélizy Villacoublay cedex
Tel : Standard : (1) 30 70 10 03
Service Technique : (1) 30 70 11 33
Telex : 698707 F

Texas Instruments France
B. P. 5
06270 Villeneuve-Loubet
Tel : 93 22 20 01
Telex : 470127 F

HOLLAND
Texas Instruments Holland B.V.
Hogehilweg 19
Postbus 12995
1100 AZ Amsterdam-Zuidoost
Tel : (020) 5602911
Telex : 12196

ITALIA
Texas Instruments Italia S.p.A.
Divisione Semiconduttori
Viale Europa, 40
20093 Cologno Monzese (Milano)
Tel : (02) 25300 1
Telex : 332633 MITEX I

SVERIGE
Texas Instruments
International Trade Corporation
(Sverigefilialen)
Box 30
S-163 93 Stockholm
Visit address : Isafjordsgatan 7, Kista
Tel : (08) 793 91 70
Telefax : (08) 751 97 15
Telex : 10377 SVENTEX

UNITED KINGDOM
Texas Instruments Ltd.
Regional Technology Centre
Manton Lane,
Bedford,
England, MK41 7PA
Tel : (0234) 270 111
Telex : 82178

Technical Enquiry Service
Tel : (0234) 223000