SPRU005A

TMS32010 Evaluation Module

User's Guide

Digital Signal Processor Products

TEXAS INSTRUMENTS

TMS32010 Evaluation Module 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.

TI warrants performance of its semiconductor products, including SNJ and SMJ devices, to current specifications in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems such testing necessary to support this warranty. Unless mandated by government requirements, specific testing of all parameters of each device is not necessarily performed.

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 device might be or are used.

Contents

Illustrations

Tables

1. INTRODUCTION

1.1 DESCRIPTION

The RTC/EVM320A, part number RTC/EVM320A-03, is a TMS32010 Digital Signal Processor Evaluation Module, referred to in this manual as the EVM. It provides the ability to develop and debug programs and test them prior to production release. Figure 1-1 shows the EVM.

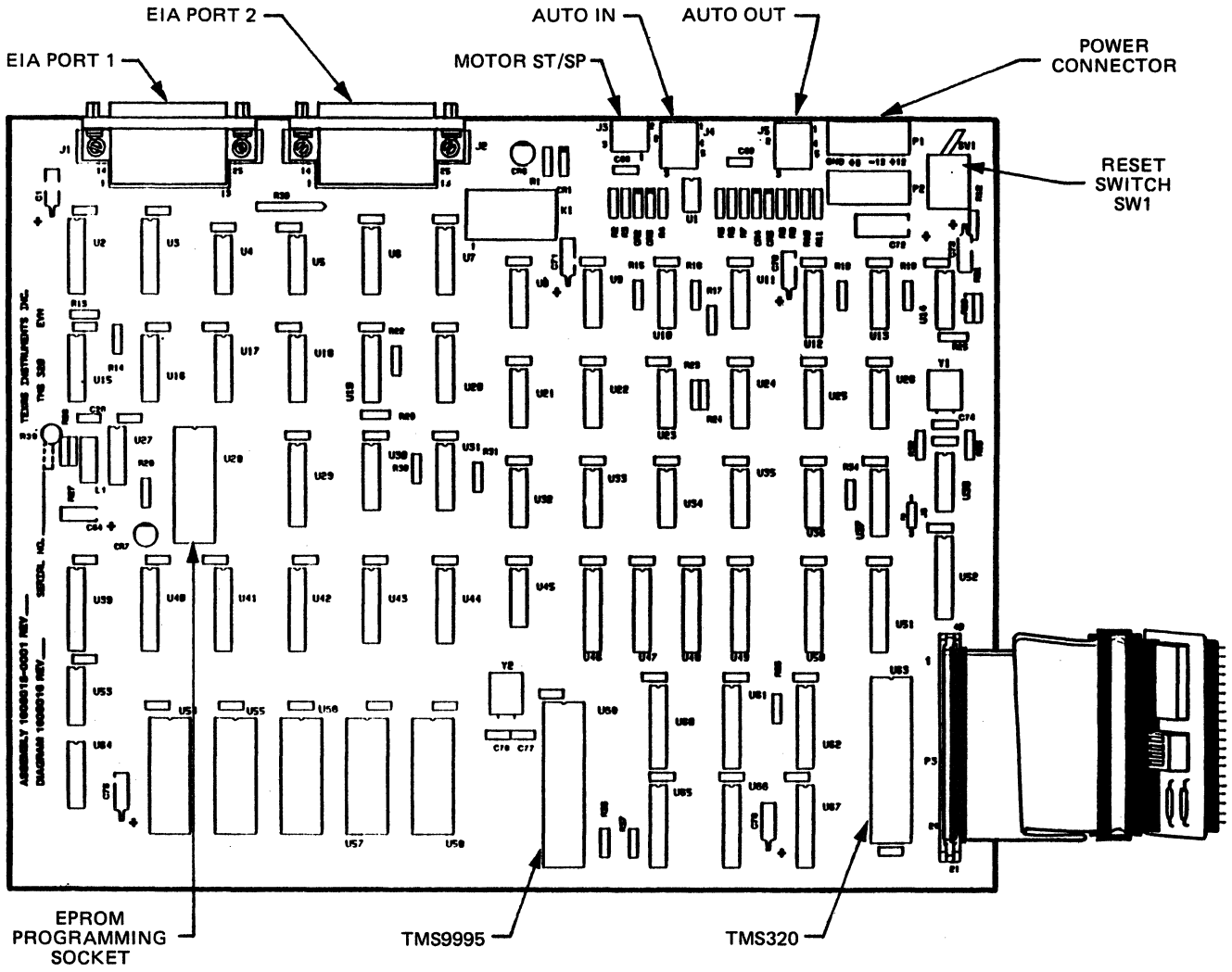


Figure 1-1. TMS32010 Evaluation Module

This manual is organized so that the reader can easily become familiar with the operation of the EVM. Detailed procedures for operating the EVM are specified, and include power-up procedures, monitor operation, instruction set execution, programming, and in-circuit emulation. Example operations are included to aid in understanding some of the more complex procedures.

1.1.1 Functional Overview

The TMS32010 EVM is a single-board development system for the TMS32010 digital signal processor. The EVM can stand alone as a development system, using the on-board full-feature text editor for the creation of TMS32010 assembly language text files, and the audio cassette tape interface (with a limited directory and file search capability) as a mass storage media. Or, the EVM can accept text files from a host CPU through one of the two EIA ports. In either situation, the resident assembler will convert the incoming text into executable code in just one pass by automatically resolving labels after the first assembly pass is complete. This object code is stored in a 4K-word memory space, allowing the utilization of the entire TMS32010 address space for developing programs.

1.1.2 Operating System Firmware

The EVM operating system firmware resides in EPROM and can be divided into four main segments:

1) The debug monitor (see Section 3)
2) The assembler/reverse assembler (see Section 4)
3) The text editor (see Section 5)
4) The TMS2764 PROM utility (see Section 6)

The EVM contains two processors configured in a master-slave relationship. The TMS9995 processor, functioning in the role of master, executes the operating firmware; the on-board TMS32010 is used to execute the user's code in real-time.

1.1.3 EVM Board Configuration

The EVM firmware supports three ports for the operations of inputting and outputting data (text and object code) for storage and/or display. Two of the ports conform to EIA RS-232C specifications and are called Port 1 and Port 2. The third port, Port 3, is an audio tape connection. Instructions for connecting devices to the ports are contained in Section 2. The ports function as follows:

Port 1: User terminal
Port 2: Host CPU uplink/donwlink or line printer connection
Port 3: Audio tape

The EVM supports baud rates of 110, 300, 600, 1200, 2400, 4800, 9600, and 19200.

The baud rate of Port 1 (terminal port) is set automatically at power-up by pressing the carriage return <CR> on the terminal after toggling the RESET switch on the EVM. This feature is called auto baud. The baud rate of Port 2 (up/down link to host) defaults to 9600 baud at RESET. The baud rates of both ports are changeable with monitor commands. (see BAUD1 and BAUD2 commands, Section 3.)

When the EVM is used in conjunction with a host system, the transparency mode provides a flexible means of both editing and transferring text files from the host system to the EVM assembler, using only the terminal connected to the EVM.

The EVM is equipped with a 40-pin emulation cable for connection to a target system. The on-board TMS32010 can be clocked internally at 20 MHz (default), or externally. The user can also select between on-board or external target memory (see the INIT command).

1.2 OTHER APPLICABLE DOCUMENTS

Other documents which may be helpful when operating the EVM include:

- TMS32010 USER'S GUIDE, part number SPRU001A
- TMS32010 ASSEMBLY LANGUAGE PROGRAMMER'S GUIDE, part number SPRU002B
- Tektronix Development System User's Guide

1.3 CUSTOMER ASSISTANCE

For help in using the TMS32010 EVM Board, call the TI Regional Technology Center nearest you. The centers are staffed with applications engineers ready to answer your questions.

ATLANTA	(404) 662-7945
BOSTON	(617) 890-6671
CHICAGO	(312) 640-2909
DALLAS	(214) 680-5066
NORTHERN CALIFORNIA	(408) 748-2220
SOUTHERN CALIFORNIA	(714) 660-8140

2. INSTALLATION AND OPERATION

2.1 INTRODUCTION

This section details the procedures for setting up the TMS32010 EVM for operation. Figures 2-1 and 2-2 illustrate the configurations for stand-alone operation and for connection with a host computer in order to use the host CPU's mass storage.

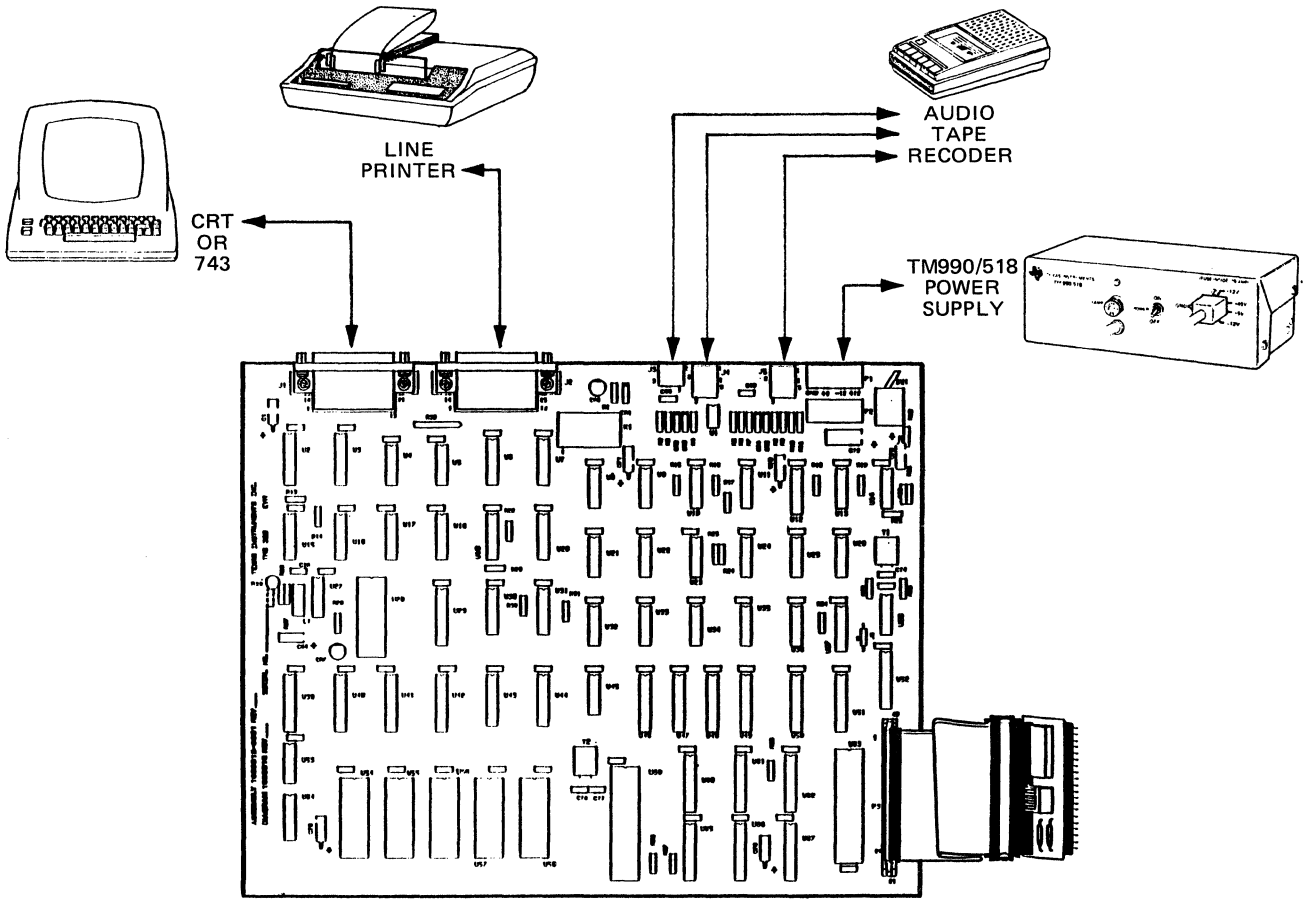


FIGURE 2-1 - STAND-ALONE CONFIGURATION OF THE EVM

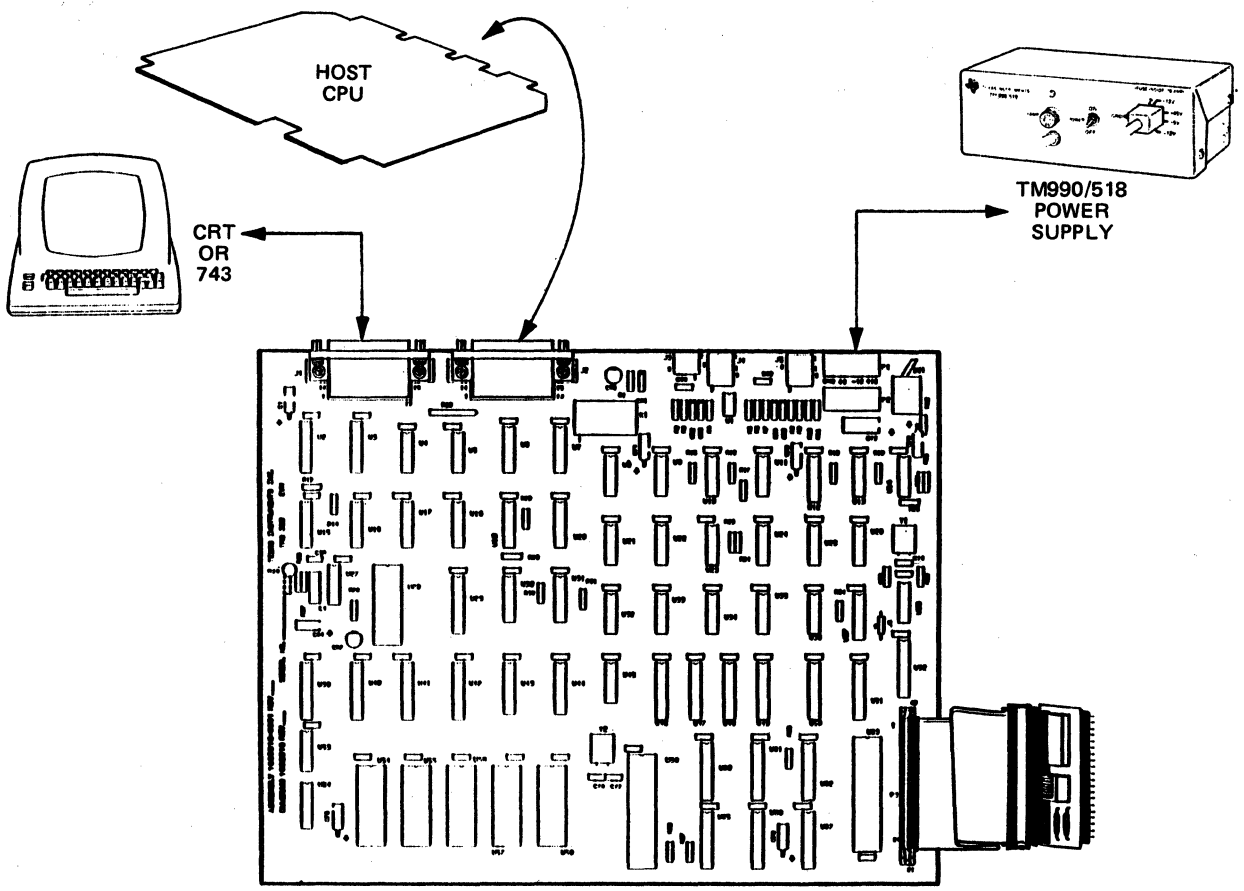


FIGURE 2-2 - EVM CONFIGURED TO USE HOST CPU AS MASS STORAGE

2.2 INSTALLATION

The following paragraphs describe connections and procedures for setting up the EVM board. The basic equipment required, along with appropriate options, is also explained.

2.2.1 Power Supply

The power supply must be UL approved, with current limitations on all outputs. A +5 volt and a +/-12 volt power supply capable of well-regulated and noise-free output is recommended. Additionally, the power supply should have a minimum current capability of +5 volts at 3 amps, -12 volts at 0.1 amps, and +12 volts at 0.1 amps. An additional connector (P2) is provided for daisy chain power connection to a target system. AMP-type connectors 1-480702-0 join to Pins P1 and P2 (pins are 350550-1). The power connectors are wired as follows:

P1/P2 PIN:	1	2	3	4
	GND	+5V	−12V	+12V

An on-board chopper circuit produces +21 V from +12 V for use during EPROM programming operations.

2.2.2 Terminal, Cables, and Tape Recorder

Any standard RS-232C-compatible terminal with a 25-pin RS-232C male plug (type DB25P) will work. For using the audio tape facility, two standard mini-to-mini cables and one sub-mini-to-sub-mini cable are required. The recommended tape recorder is a Radio Shack CTR-41 or its equivalent.

2.2.3 I/O Connections

The EVM has two EIA RS-232C port connections (labeled J1 and J2 in Figure 1-1), and a connection for attaching an audio tape recorder to be used as a mass storage device (labeled J3 in Figure 1-1).

The Evaluation Module does not support a 20mA current loop interface, but adapters can be purchased from outside vendors to perform the conversion.

The EVM supports both hardware and software handshaking protocols for terminals operating in the full-duplex mode. Section 2.7 details the EVM EIA hardware.

2.2.3.1 Terminal Connection

The user's terminal is connected to the EVM at connector J1, hereafter referred to throughout the manual as Port 1 (see Figure 1-1).

2.2.3.2 Host/Printer Connection

The EVM uplink/downlink or line printer connection is made at connector J2, hereafter referred to throughout the manual as Port 2 (see Figure 1-1). The connections for host and printer are illustrated in Figures 2-1 and 2-2.

The following are examples of connections to both host computers and printing devices:

(1)

TMS32010 EVM		820 KSR
PIN 2 RX	←	PIN 2 TX
PIN 3 TX	→	PIN 3 RX
PIN 8 DTR	→	PIN 6 DSR
PIN 7 GND	—	PIN 7 GND
PIN 6 PDCD *	→	PIN 8 DCD
PIN 20 DSR	←	PIN 11 SCA (BUSY)
	→	PIN 4 RTS
	→	PIN 5 CTS

* PDCD is +12 V when EVM power is ON.

(2)

TMS32010 EVM		743 KSR
PIN 1 GND	—	PIN 9 GND
PIN 2 RX	←	PIN 13 TX
PIN 3 TX	→	PIN 12 RX
PIN 7 GND	—	PIN 1 GND
PIN 6 PDCD *	→	PIN 11 DCD
PIN 20 DSR	←	PIN 15 DTR

* PDCD is +12 V when EVM power is ON.

(3)

TMS32010 EVM		810 LINE PRINTER
PIN 1 GND	—	PIN 1 GND
PIN 3 TX	→	PIN 3 RX
PIN 8 DTR	→	PIN 6 DTR
PIN 7 GND	—	PIN 7 GND
PIN 6 PDCD *	→	PIN 8 DCD
PIN 20 DSR	←	PIN 11 DTR (BUSY)

* PDCD is +12 V when EVM power is ON.

(4)

TMS32010 EVM		990 EIA CARD
PIN 2 RX	←	PIN 3 TX
PIN 3 TX	→	PIN 2 RX
PIN 8 DTR	→	PIN 20 DSR
PIN 7 GND	—	PIN 7 GND
PIN 6 PDCD *	→	PIN 18 DCD
PIN 20 DSR	←	PIN 8 RTS

* PDCD is +12 V when EVM power is ON.

NOTE:

The PDCD signal is pulled to +12 volts through a 3.3 Kilohms resistor. 765-type terminals may require a resistance value in the 1 Kilohm range.

(5)

TMS32010 EVM		FS990/4
PIN 2 RX	→	PIN 3 TX
PIN 3 TX	→	PIN 2 RX
PIN 8 DTR	—	PIN 20 DSR
PIN 7 GND	—	PIN 7 GND
PIN 20 DSR	←	PIN 8 RTS

TMS32010 EVM		"DUMB" CRT **
PIN 1 GND	—	GND
PIN 2 RX	←	TX
PIN 3 TX	→	RX

** "DUMB" CRT has all handshaking disabled.

2.2.3.3 Audio Tape Recorder Connection

When used in a stand-alone configuration (see Figure 2-1), the EVM supports one audio tape recorder as a mass storage device. Connections to a tape recorder are detailed in Table 2-1. The connection is illustrated in Figure 2-1.

TABLE 2-1 - AUDIO TAPE RECORDER TO EVM CABLE CONNECTIONS

FUNCTION	EVM PLUG	CASSETTE PLUG
Audio Tape Motor	J3	Remote
Audio Data In	J4 (in)	Ear (monitor)
Audio Data Out	J5 (out)	Microphone

Three connections are provided between the EVM and the tape recorder. Data-out and data-in are accessed with standard dual-ended male mini cables. The motor control connection is made with a dual-ended male sub-mini cable. All cables are easily obtained from local vendors.

The tape cassette motor control is provided by the EVM to insure proper tape starting during dump of multiple-block operations, and this connection is required. The proper setting of the volume control when reading data from the tape is between 6 and 8 on a scale of 10. If the tape recorder has a tone control, it should be set to high.

2.2.3.4 Target Board Connection

A target board may be connected to the EVM by means of a 91605021 target connector cable, as shown in Figure 2-3.

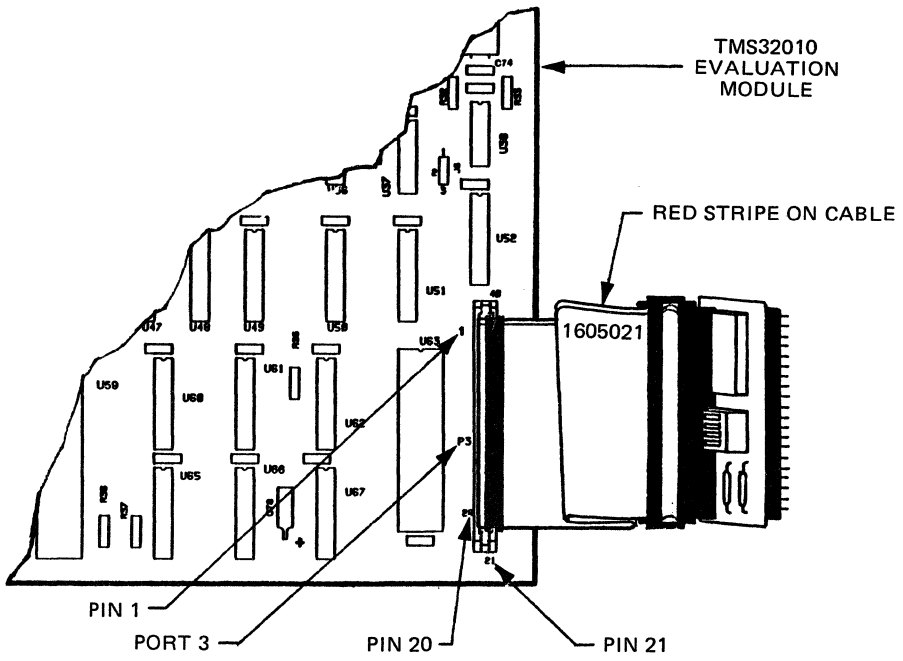


FIGURE 2-3 - CORRECT CONNECTION OF TARGET BOARD TO EVM

CAUTION

The target connector cable MUST be plugged into both the TMS32010 EVM and the target system with Pin 1 connected to Pin 1, or damage may result. Turn the power off when plugging or unplugging the connector cable.

2.3 OPERATION

The following paragraphs describe various functional features of the EVM board.

2.3.1 EVM Memory

EVM memory is divided into three main parts: operating system EPROM, operating system RAM, and user RAM. The EVM memory is discussed in Section 9.2. The memory map is illustrated in Figure 2-3.

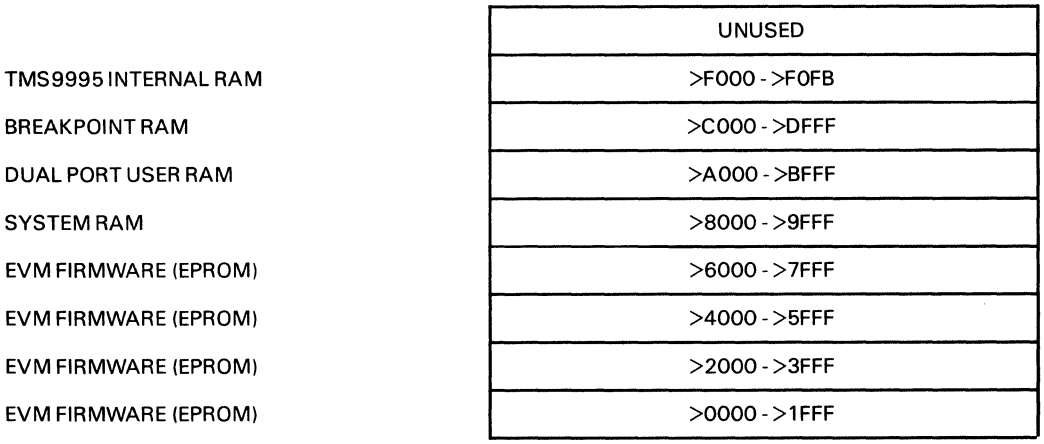


FIGURE 2-4 - EVM MEMORY MAP

2.3.1.1 RAM

The TMS32010 can address 4K words of memory. Four RAM chips (U46-U49) are designated for on-board RAM. Operating system RAM is contained in one 8K x 8 RAM chip (U54).

2.3.1.2 EPROM

The EVM firmware is contained in four TMS2764 EPROMs on the board. The socket addresses are illustrated in Table 2-2, and are addressable only by the EVM host processor (a TMS9995).

TABLE 2-2 - EPROM SOCKET ADDRESSES

CHIP	START ADDRESS	STOP ADDRESS
U58	>0000	>1FFF
U57	>2000	>3FFF
U56	>4000	>5FFF
U55	>6000	>7FFF

2.3.2 The RESET Switch

The RESET switch (SW1) is located at the upper right corner of the EVM board and provides the source of the RESET pulse that initiates the EVM operation and provides a restart for the EVM software at any time. Because of the auto-baud feature of the terminal at Port 1, the user must press the carriage return on the terminal after performing RESET in order to initiate the EVM operating system. If the EVM initializes properly, the RESET banner will appear as follows:

```
          *** TMS32010 EVM OPERATING SYSTEM ***
                      REVISION 1.X

    .
    .
    .
"HELP"  MONITOR COMMANDS
    .
    .
    .
 ?
```

The question mark is the EVM monitor prompt. For baud rates less than or equal to 1200 baud, the RESET banner consists of the revision level, monitor banner, and HELP prompt only. For baud rates greater than 1200 baud (up to 19200 baud), the RESET banner includes initialization messages for breakpoints, event counter, trace, clock, and program memory in addition to the monitor menu. The monitor menu is available at any baud rate at any time (see Display monitor Menu Command, Section 3.2.4).

The EVM senses whether power has been cycled between RESETs. If power has not been cycled and/or the reset is not a "warm" RESET as described in Section 2.3.10, the breakpoints, event count, and trace locations are cleared and the program memory/clock sources are set to internal. If power has been cycled, the following additional functions are performed:

- The text editor is initialized
- The assembler table is reset
- Terminal tabs are reset
- All TMS32010 registers are zeroed

If the user wants to perform a power cycle RESET without actually cycling power, the monitor RESET command is provided. For example:

```
?RESET
ARE YOU SURE? (NO) Y
ENTER <CR>
.
.           [RESET Banner]
.
```

Port 2 defaults at RESET to 9600 baud. The default is changeable (see User-Changeable Functions, Section 2.3.9).

If the EVM does not initialize properly, check that the terminal is properly connected and that power is properly applied.

2.3.3 Keyboard Entry Aids (Special Function Keys)

The EVM supports a flexible set of character entry aids (special function keys) which permit editing of the current input line prior to entering the data (with a carriage return). They are modeled after Lear/Siegler, Adds, Televideo, and Hazeltine dedicated keys. VT52 <ESC> sequences are also supported (see Section 2.3.9.3.2). These entry aids allow editing of the current input line prior to entering a <CR>. Spare locations are provided in the monitor EPROM to expand both the single control character set and any <ESC> sequences (see User-Changeable Functions, Section 2.3.9). Table 2-3 lists these aids (function keys) and their equivalent control characters. The keyboard entry aids are also summarized in Appendix B.

TABLE 2-3 - KEYBOARD ENTRY AIDS (SPECIAL FUNCTION KEYS)

KEY/KEYS†	FUNCTION
ESC	Delete Line/Create New Line
RUB/DEL	Delete Last Character on the Line
CNTL/N	Insert a Character
CNTL/D	Delete a Character
CNTL/F	Move Cursor to the Right (———>)
CNTL/L	Move Cursor to the Right
CNTL/P	Move Cursor to the Right
CNTL/H	Move Cursor to the LEFT (<———)
CNTL	Redisplay Line
CNTL/A	Cursor Home
CNTL/^	Cursor Home
ESC/CNTL/R	Cursor Home
CNTL/E	Move Cursor to End of Text
CNTL/X	Delete All Characters on Line
CNTL/Y	Delete to End of Line from Cursor
CNTL/I	Tab Right (Tab)
ESC/I	Tab Left (Back Tab)

† In the case of multiple keys, press simultaneously.

With baud rates from 110 to 1200, the insert (CNTL/N) and delete (CNTL/D) functions will accept a character from 1 to 9 after entry, redisplay the altered line on the next line with the insertion, or deletion of the specified number of characters and the cursor at the original position. With baud rates from 2400 to 19200, these functions are instantaneous and are performed on the same line.

With fast printing terminals, a "redump line" command (CNTL is provided. When this command is executed at the prompt level with the input buffer empty, it will alternately display the previously executed command STRING or the previously executed CHAIN. For example:

```
? [ CNTL ]              Display most recent command STRING
?<ESC>[ CNTL ]          Display most recent CHAIN
```

Alternating <ESC> key entry will toggle the buffer displayed by the CNTL key. When displayed, each buffer is ready for execution when a <CR> is entered. Each command/STRING and/or CHAIN is held intact in a separate buffer until a new command/STRING or CHAIN is executed. This allows the user to quit execution of a long chain, execute some commands to alter the machine state, and then recall and execute the chain without having to retype it.

The EVM input software converts lower-case alpha characters to their upper-case equivalents except during text entry in the text editor mode. Therefore, either lower- or upper-case letters will be accepted as command inputs.

2.3.4 Command Concatenation (STRINGS And CHAINS)

At the command entry level during EVM activity, it is legal to fill the input buffer with characters that will satisfy a command, thereby satisfying a command on one line. This is called a command STRING. Commands in a string are separated by a semicolon (;) which acts as a <CR> when the buffer is read. All command responses from the EVM will be displayed. For example:

```
?PC<CR>
  PC = 055 0,R<CR>
  PC = 000  <CR>
?

or:

?PC 0,R;<CR>
  PC = 055
  PC = 000
?
```

Note that the use of the "R" redisplay command provides display of the newly altered program counter (the old PC is displayed before being changed to the value entered on the command line). A semicolon (;) after the "R" terminates the subcommand, allowing the carriage return <CR> to terminate the entire command string. Without the semicolon, the command will halt after redisplaying the new PC value, waiting for input. The semicolon after the last command in the string is optional, as are trailing spaces.

The only limitation on command strings is the length of the line buffer, 72 characters. Occurrence of any error will terminate the concatenated command string. If a subcommand is included in the concatenated command string, an additional semicolon must be included to terminate the subcommand. For example:

```
?CLEAR;PC 3,R;;STATE<CR>
```

will clear all TMS32010 locations, load the program counter with 3, then display the TMS32010 register set. While it was executing, the normal EVM responses to all inputs will be printed on the monitor.

Another example:

```
?PROM;READ;QUIT;PC 0,R;;RUN<CR>
```

In this case, the command string will enter the PROM utility, read the EPROM into RAM, quit the PROM utility, set the program counter to zero, and execute the program without breakpoints.

2.3.4.1 Chain Terminators

When using command strings, four terminators are provided to execute the strings either a fixed number of times and/or until entry of <ESC> from the terminal. When one of these terminators is used as the last character in a command or string, it is called a CHAIN. During execution of a chain, display of prompts is suppressed, allowing more efficient use of the terminal screen and making the display easier to read. The four terminators are:

- `*` Execute continuously and scroll down
- `%` Execute continuously and scroll up
- `!` Execute once and scroll up
- `#` Execute continuously with output disabled

The scroll down terminator (*) continuously executes the chain and displays each result on a new line. During execution, entry of a <SP> will start/stop the display after one complete execution of the chain, and entry of <ESC> will terminate execution.

The two scroll up terminators (% and !) continuously execute the chain, but after one execution, print the number of cursor-up characters to the screen equal to the number of line feeds in the display, and then continue the execution. The effect is to freeze the display. the display must not have more lines than are on the terminal screen.

The cursor-up character used by the EVM may be changed by the user (see Section 2.3.9). If the terminal used accepts a different character for cursor-up, the user must make a new EPROM with the correct character or character sequence (up to three characters long) for use with that terminal (see Section 6).

When using the % terminator, entry of a <SP> will start/stop the display after one complete execution. When using the ! terminator, a <SP> entry is required to perform one execution of the chain, and entry of <ESC> will terminate execution.

The output off terminator (#) will execute the chain with all output disabled until entry of <ESC>.

Several monitor commands aid in the use of command chains. The first is the Chain Library command (CLIB). The format is:

```
?CLIB ‹output port›
```

where the output port is 1 (default) or 2. This allows a listing of the library functions. The library lists six preprogrammed chains that can be loaded into the chain buffer by typing the command:

```
Cx
```

where 0 <= x <= 9. The chain buffer is then displayed for execution by entering:

```
?‹ESC›[ CNTL‹ ]
```

The chain library display also lists the four types of terminators and displays the character(s) currently being used for cursor up. Chains are not limited to those contained in the library. Chains of the user's creation can also be entered at any time by the user from the terminal and will be saved in the chain buffer when executed. The addition of chains of the user's creation in those slots designated as UNUSED in the library display is discussed in Section 2.3.9. Execution of a Cx command for an unused slot will leave the previous contents of the chain buffer intact.

2.3.4.2 Chain Counts

To limit the duration of execution of a chain, a count can be entered with the C command as follows:

```
?C
COUNT (0) XXXXX                0 <= xxxxx <= 65535
```

The count is a decimal number with the above range. Entry of 0 (default at RESET) yields infinite execution. If a non-zero number is entered, the count remaining is displayed after one execution of the chain (except when the # terminator which suppresses output is used). If the count is 0 (infinite), it is not displayed. When the count expires, chain execution terminates but the count is left intact for future chain execution and is used as the default for the next execution of the C command.

2.3.5 Input/Output Features

The following paragraphs describe some of the EVM I/O features applicable to Ports 1 and 2, the EIA RS-232C communication links.

2.3.5.1 EVM Character Format

The EVM character format for both ports is:

- 1 start bit
- 7 data bits
- 2 stop bits
- no parity bit

The character format is changeable for either port (see Table 2-6).

2.3.5.2 EIA Communications Protocol

Four commands are provided to enable and disable DEC and Tektronix software handshaking protocols. At cold RESET, both protocols are disabled. Hardware handshaking is always enabled.

Software handshake characters are transmitted out the download port during buffer input. The commands function as follows:

XON: Enable transmission of XON (DC1, >11)
Enable transmission of XOFF (DC3, >13)

XOFF: Disable XON/XOFF transmission

ACK: Enable transmission of ACK (0<CR>)

NACK: Disable transmission of ACK (NOTE: the EVM never transmits a NACK (7<CR>)

To enable software handshaking with any system except Tektronix, enter: XON<CR>. If for some reason it should be desired to disconnect the software handshaking facility, enter: XOFF<CR>. To enable software handshaking with a Tektronix system, enter: ACK<CR>; to disable, enter: NACK<CR>. If both types of software handshaking are enabled, the XON is transmitted before the ACK.

During uplink, after the transmission of each character, the EVM will check for reception of an XOFF character. If an XOFF character is received, the EVM will wait for an XON character before resuming transmission. During downlink, the EVM will transmit an XON prior to starting to fill the downlink buffer, and will transmit an XOFF while receiving the 480th character in the 512-byte buffer. Any additional characters are then received before a character timer determines that transmission has stopped and the buffer is unloaded internally. The value used for XON is DC1, and for XOFF, DC3; these values are changeable (see Section 2.3.9).

2.3.5.3 Fixed/Variable Length Records

The EVM supports both fixed and variable length record formats during downlink. The only constraint is that a variable length record transmission to the EVM must not be greater than the current length of the fixed record setting. The default fixed record length at RESET is 82 characters; it is configurable from between 50 to 142 characters inclusive. (see $DRL monitor command).

2.3.5.4 Hardware Handshaking

The EVM uses handshaking in transmitting and receiving data through Ports 1 and 2. Handshaking is done on a character-by-character basis when interacting with a terminal at Port 1. With 3-wire connections to dumb terminals, unused lines are self-satisfying. During download from Ports 1 and 2, handshaking is done on a buffer-by-buffer basis, 512 bytes at a time.

Handshaking begins with the EVM placing + 12 V on pin 8 of the selected port. the EVM will then wait until pin 20 of the selected port rises above + 4 V (+ 4 V to + 12 V). There is a twenty-second timeout when waiting for this signal from Port 2. The duration of the timeout can be changed; see Section 2.3.9. After the handshake signal is received, the data is transmitted. The EVM will check the handshake line before each byte of data is transmitted. The signal at pin 8 will go from 4 V to -12 V after character reception during character handshaking, but will wait until 480 characters have been received during buffer handshaking. At that time, any additional characters are received before a character timer determines that transmission has stopped and unloads the buffer internally. The EVM will pull all handshake input line to + 12 V through a 3.3 kohms resistor.

The EVM checks the handshake line before each byte of data is transmitted. If at any time the DSR line is not satisfied, the EVM will wait in a loop until it is satisfied. If a terminal without handshaking facility is being used, do not connect pin 20; the EVM will pull the handshake line to + 12 V through a 2.2 K resistor.

2.3.6 Transparency Mode Support

Transparency mode provides a means of communication between a host system connected to the EVM at Port 2 and the EVM downlink software by allowing the user to log on to a host CPU and the EVM from one terminal. Each time the toggle character is entered from the keyboard, transparency mode is toggled between monitor control and transparency mode control, thereby switching the terminal between host and EVM operating systems. By using transparency mode, the EVM terminal is able to emulate a host terminal at Port 2 of the EVM and stimulate the host to upload/download files to/from the EVM. Figure 2-4 illustrates the transparency mode configuration.

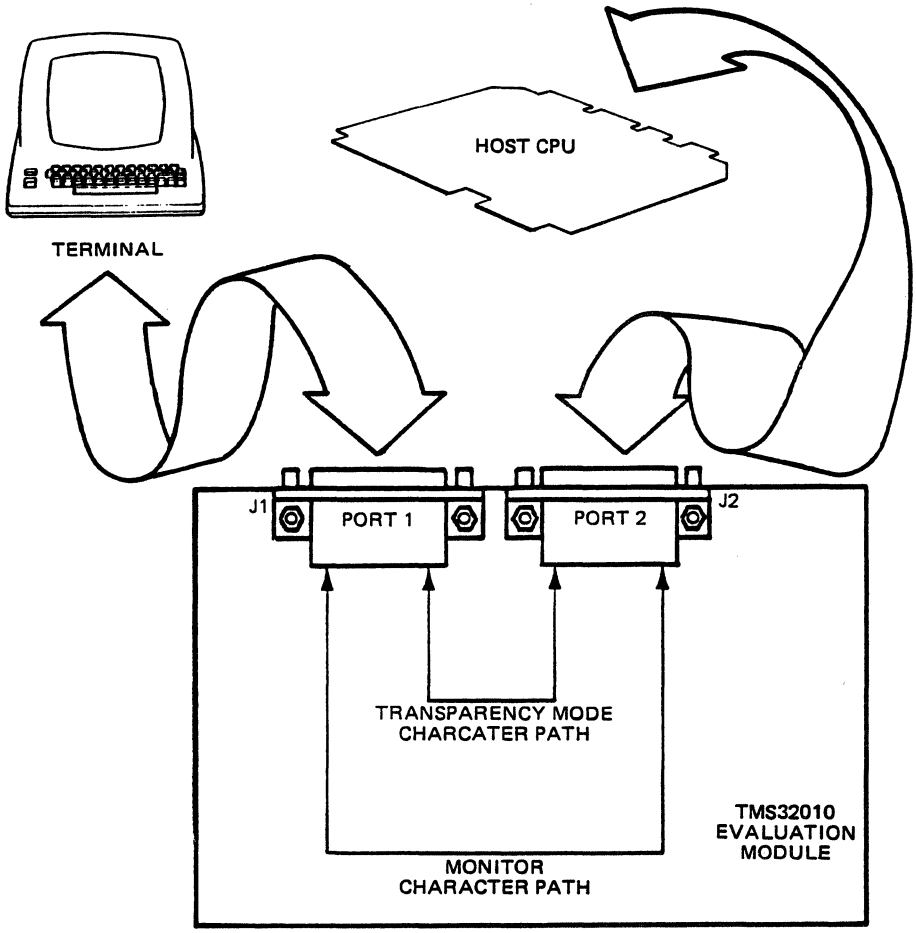


FIGURE 2-5 - TRANSPARENCY MODE CONFIGURATION

2.3.6.1 Display/Modify Toggle Character Command (COMM)

The transparency mode toggle defaults at power-on initialization to the CNTL/C character. If the host system uses this character extensively and the "PASS" command (discussed in Section 2.3.6.2) does not allow for enough flexibility in transmitting this character, the COMM command may be used to change the toggle character. Once the toggle character is changed, only cycling the power to the EVM will cause the default to be reloaded.

When using the COMM command, the ASCII value of the current toggle character is displayed. Entry of the character is in ASCII and is achieved by simply typing in the character at the terminal. If the character is reserved for use by the EVM (see Table 2-4), the terminal will "beep" and wait for another input. If the character is accepted, it is redisplayed. The COMM command is terminated by a carriage return. An example of how to change the toggle character is:

```
?COMM
TOGGLE CHARACTER = CNTL/C [ CNTL/U ]
TOGGLE CHARACTER = CNTL/U <CR>
```

In this example, the user entered CNTL/U, which was accepted and redisplayed on the next monitor line. The user terminated the session with a carriage return.

Appendix B also includes a list of control characters with their ASCII values and notes whether or not they are reserved by the EVM. Illegal toggle characters include the keyboard entry aids listed in Tables 2-3 and 2-4.

TABLE 2-4 - CONTROL CHARACTERS RESERVED FOR EVM USE

KEYS	FUNCTION
CNTL/[	ESCAPE
CNTL/M	CARRIAGE RETURN
CNTL/J	CURSOR DOWN
CNTL/K	CURSOR UP
CNTL/V	CURSOR DOWN
CNTL/Z	CURSOR UP

2.3.6.2 Transmit Toggle Character to Port 2 Command (PASS)

Since the transparency mode toggle character is intercepted (and not transmitted), the PASS command transmits the toggle character out of Port 2. This command is identical to the P command in the text editor. The toggle character used in the following example is the default (CNTL/C). To transmit the toggle character, the following sequence or (equivalent) must be executed:

```
[ TRANSPARENCY MODE ACTIVE ]
[ CNTL/C ]

[ MONITOR ACTIVE ]

?PASS

?[ CNTL/C ]
[ TRANSPARENCY MODE ACTIVE ]
```

In the example above, notice that leaving the transparency mode caused the monitor prompt (?) to be printed. Since the PASS command will only execute from the monitor top level, it can be used as needed to log on to a host system before entering transparency mode from within an active load command.

2.3.6.3 Downloading in Transparency Mode

The following example demonstrates the download procedure while the EVM is in transparency mode. It is based on interaction between the EVM and a Texas Instruments 990 minicomputer. (The LPM command (Load Program Memory) is explained in Section 3.2.17)

```
? LPM 2
[ CNTL/C ]
[ <ESC>! ]
SYSTEM COMMAND INTERPRETER - PLEASE LOG IN
   USER ID: VDT101
  PASSWORD:
[ST22] SF
SHOW FILE
  FILE PATHNAME: DS01.TEST [ CNTL/C ]
14      IDT: TEST

?
```

The sequence shown above begins with the user entering the monitor command to load program memory from Port 2. This means that the EVM is expecting object code to be input from Port 2. After terminating the command line, the EVM will idle as it waits for the input.

A CNTL/C is then entered and the monitor displays the new line. The EVM is now in transparency mode and any characters (except CNTL/C) entered at the terminal keyboard will be sent out from Port 2 to the host system. Likewise, any characters received in this mode will be sent to the terminal and displayed. The baud rates of Ports 1 and 2 do not have to be the same.

The next sequence, [<ESC>!], is the log-on character sequence for this particular command interpreter. After logging on to the host, a command is executed on the host to show a file on the "terminal" that the EVM has now become. After entering the pathname, instead of entering a <CR> to terminate the command normally and show the file to the terminal, the user should instead toggle out of the transparency mode.

In any instance, entry of the CNTL/C should be in place of the last <CR> before the dump begins. This does two things: (1) sends a <CR> to the host to start the dump, and (2) returns control to the current monitor command, in this case, "LPM 2".

Finally, download is initiated and the EVM responds by counting the number of blocks as a way of monitoring the progress of the download.

The following examples are the command strings for the entry of the Load Machine State command (Section 3.2.15), the assembler (Section 4), and the text editor (Section 5) via the transparency mode. After entry of the first CNTL/C, user response is identical to the example above. Remember that the LPM command expects object file, while the ASM and EDIT commands expect source files, and the LMS command expects machine-state files.

```
(1)     ?LMS 2
        [ CNTL/C ] ...

(2)     ?ASM 2 1
        LINE NUMBERS? (N)
        [ CNTL/C ] ...

(3)     ?EDIT 2
        LINE NUMBERS? (N)
        [ CNTL/C ] ...
```

2.3.6.4 Uploading in Transparency Mode

The procedure for upload from the EVM to a host system is similar to the download procedures described in the previous section. The example below describes an upload from the EVM text editor (see Section 5) and assumes that the user has already logged on to the system. The transparency mode toggle character used in the example is the default CNTL/C.

```
*[ CNTL/C ]

[ST22]CC
COPY/CONCATENATE
  INPUT ACCESS NAME(S): ME
    OUTPUT ACCESS NAME: .TEST.SOURCE
               REPLACE?: NO/YES
 MAXIMUM RECORD LENGTH: <CR>
[ CNTL/C ]

*QUIT 2

** TMS320 EVM MONITOR **
.
.
.

?
```

This example uses the Copy/Concatenate command. Input is specified as "ME" (the EVM terminal), output is specified as the file name the user wants the text to be stored in on the host system. The file is to be replaced if it already exists (this could be an older version), and the record length is defaulted.

After execution of the CC command, CNTL/C is entered to toggle the mode, and the text editor prompt (*) is displayed. the user simply has to quit the text editor to Port 2, thereby dumping its contents to the host file.

2.3.7 Terminal Emulation Support

The command strings for the assembler (ASM), the text editor (EDIT) entry, and the Load Machine State (LMS) and Load Program Memory (LPM) commands allow for users running terminal emulation software on a host computer connected to the EVM at Port 1. This configuration allows the addition of a line printer at Port 2, while still supporting upload/download capability with a host system supporting a mass storage device. While downloading from Port 1, reception of an <ESC> character will terminate the load and return control to the monitor.

The transparency mode toggle character can also be used to allow a line printer at Port 2 to function as a system line printer when using terminal emulation. By sending the toggle character to the EVM from the controlling keyboard, subsequent characters sent to the EVM will be dumped by the EVM to Port 2. After the transmission is finished, sending the toggle character again will return normal control to the monitor.

2.3.8 Dual EVM Master/Slave Operation

The cable connection shown below is to connect Port 2 of a "master" EVM to the Port 1 (terminal) of a "slave" EVM and to control both (via the transparency mode described in Section 2.3.6, and the terminal emulator mode described in Section 2.3.7) using only one terminal connected to Port 1 of the master EVM. The pin connections are:

TMS32010 EVM (Master)		TMS32010 EVM (Slave)
Pin 2 RX	←	Pin 3 TX
Pin 3 TX	→	Pin 2 RX
Pin 8 DTR	→	Pin 20 DSR
Pin 7 GND	—	Pin 7 GND
Pin 20 DSR	←	Pin 8 RTS

Complete functioning of the EVMs as host peripherals or in stand-alone modes is possible. There are two primary uses for a master/slave configuration:

1) Use one EVM as a text storage media in stand-alone mode in order to use the other EVM to assemble/execute the program without destroying the text file on the first EVM.

2) Use the slave EVM as a second TMS32010 emulator in a dual-processor design. This mode allows the download of object/text files from a host (connected at Port 2 of the slave) to the master, or the download of text/source from an audio tape recorder connected to either the master or the slave EVM. In this mode, no medium exists to start both EVMs synchronously.

When using transparency mode on the master EVM to toggle the terminal between master and slave, the EVM prompts will be displayed each time a different EVM is accessed. To allow the user to easily distinguish between the EVMs, a monitor command is provided to append a user-selectable character to all EVM prompts, thus making all prompts from one EVM distinguishable from the same prompts on the other EVM. The command is :

```
?$ID <ID character>
```

Legal values for the ID character are letters, numbers, and all symbols. Entering more than one character will use the first character entered. Defaulting the character (entering a <CR> immediately after the command) will return the ID to its default value. The ID character defaults at RESET to >00. An ID with this value is not displayed with the prompt or in all single EVM modes of operation.

2.3.8.1 Initialization

The following sequence is required to initialize both EVMs prior to master/slave operation. The master is initialized first, followed by the slave. (In all examples in this section, the default transparency mode toggle character (CNTL/C) is used.)

```
[ Press RESET on Master EVM ]
[ Press RESET on Slave EVM ]
<CR>                          From terminal autobauds master EVM
.
.    (Master EVM banner)
.
?$ID M<CR>                    Set master ID
[ Optionally change Port 2
  baud rate to 19200. see
  Section 2.3.9 to change
  power-up default ]
M?[ CNTL/C ]
.
.    (Slave RESET banner)
.
?$ID S<CR>                    Set slave ID
S? [ CNTL/C ]                 Return to master
M?
```

After initialization, the ID values will remain intact until a power cycle RESET.

Just as entering CNTL/C from the terminal puts the master in transparency mode, use of the PASS command on the master will put the slave in transparency mode. Execution of a subsequent PASS command will toggle the slave back into monitor mode. Communication between a host computer and the master EVM is accomplished by placing the slave EVM in transparency mode and then entering transparency mode on the master with a CNTL/C. Once that is done, the terminal is in communication with the host system, and download/upload between the master EVM and the host system is accomplished as described in Section 2.7.

Communication between master and slave EVMs is simpler. In this mode, the slave is treated as the "host", and all the examples in Section 2.7 apply. In this mode, text, object code and machine states can be transferred between EVMs.

2.3.8.2 A Complete Stand-Alone System

The optimal dual-EVM single emulator configuration uses the slave EVM for emulation and the master EVM for text storage. This allows assembled source listings at Port 2 of the slave. The configuration is as follows:

MASTER EVM: Port 1: Terminal
Port 2: Connected to slave EVM Port 1
Port 3: Audio Tape

SLAVE EVM: Port 1: Connected to master EVM Port 2
Port 2: Line Printer (host optional)
Port 3: Audio Tape (optional)

In this configuration, text files can be loaded into the text edior on the master EVM either from the host or from the audio tape connected to the master. If files are loaded from the host, a line printer is then connected to slave Port 2 for assembled source listings. Text can then be modified until ready for assembly. At assembly, the slave EVM must be set up to accept source from its Port 2. For example:

```
M*[ CNTL/C ]                    Leave master text edior
S?ASM 1 2                       Init slave assembler
LINE NUMBERS (NO) Y<CR>         Must have line numbers
[ CNTL/C ]                      Slave ready, return
M*Q 2<CR>                       Q master editor to 2
M?[ CNTL/C ]                    Assembly complete
S?                              ...Go to slave to ...
.
.  (Normal Debug Cycle)         ...Debug
.
.
S?[ CNTL/C ]                    Debug complete
M?EDIT<CR>                      Return to master editor
.
.  (text edior Banner)          Implement changes to text
.
M*                              Edit file and repeat
```

After assembly is complete, the user will have an assembled source listing that was generated at Port 2 of the slave EVM and can execute the program on the slave EVM. If any modifications need to be made to the program, the text file is still intact on the master EVM. The process can be repeated indefinitely.

The line printer connected to the slave at Port 2 can also be used by the master to generate hard copy of object code dumps and text editor contents. To generate a dump of object code, the sequence is:

```
M?PASS                          Enable slave transparency
M?DPM 0 FFF 2                   Dump to printer through slave
FORMAT? (HX)<CR>
M?PASS                          Disable slave transparency
```

To generate a dump of text editor contents (executed from text editor), the sequence is:

```
M*PASS                          Enable slave transparency
M*Q 2                           Quit to printer
M?PASS                          Disable slave transparency
```

2.3.8.3 The Slave EVM as a Companion Emulator

For designs employing two TMS32010 processors, each EVM in a master/slave configuration can be configured as an emulator, either in the stand-alone mode (assembling files from audio tape) or in the download mode. The cable connection is as follows:

MASTER EVM: Port 1: Terminal
Port 2: Connected to slave EVM Port 1
Port 3: Audio tape (stand-alone mode)

SLAVE EVM: Port 1: Connected to master EVM Port 2
Port 2: Host CPU (Printer optional)
Port 3: Audio tape (stand-alone mode)

With the use of transparency mode, the user alternately communicates with each EVM, loading and dumping text and object, assembling programs into program memory, and generally preparing each EVM to execute its own program.

A major limitation in download mode is that the slave cannot be configured to directly assemble a file from the host system. The file must pass through the slave and be assembled by the master. The object code is then transferred from master to slave. Communicating to the host system from the master through the slave is accomplished by the following sequence (in this case, assembling):

```
M?PASS                      Put slave in transparency mode
M?ASM 2 1                   Set up download on master
LINE NUMBERS? (NO) <CR>
[ CNTL/C ]                  Put master in transparency mode
.
.
.
[ CNTL/C ]                  Execute download
M?PASS                      Remove slave transparency
```

The sequence of transferring data (in this case, object code) from the master to the slave is:

```
M?[ CNTL/C ]                Enter slave
S?LPM 1                     Set up slave load
[ CNTL/C ]                  Return to master
M?SPM 0 FFF 2               Set up master dump and execute
M?                          at 96+ baud (approx. 1 minute)
```

At execution time, the following sequence will start both EVMs executing as closely together as possible. No mechanism exists to synchronously start both EVMs executing the user program. Another example:

```
M?
.
(Set master breakpoints, etc.)
.
M?EX[ CNTL/C ]          Type master EX and enter slave
S?
.
(Set slave breakpoints, etc.)
.
S?EX[ CNTL/C ]          Type slave EX and return to master,
                          (slave now executing)
M?EX<CR>                EX on master reappears. Enter <CR>:
                        both EVMs executing
```

Once both EVMs are executing, the master must encounter a breakpoint or be RESET before the status of the slave can be checked. When checking the slave, entry of CNTL/C from the master prompt will cause either a slave prompt to be displayed (if a breakpoint has been encountered) or no display (if a breakpoint has not yet been encountered.) If a breakpoint has been encountered on the slave, executing DB and PC commands will display all the breakpoints and the program counter. The breakpoint matching the PC was the one encountered. The string:

```
S?DB;PC;<CR>
```

will display the needed information.

2.3.8.4 Pseudo-Synchronous Emulation

The monitor commands $EC and $$EC (see Section 3) can be used together to control the master and slave EVMs by allowing them to execute the same commands. Use of the $$EC command is optional and requires a display at slave Port 2. If used, the $$EC command is first entered on the slave, and provides a means of monitoring slave output activity while Port 1 is communicating with the master. In this mode, output sent to the terminal is also sent to Port 2. Input at Port 1 is also sent to Port 2.

The $EC command is then executed on the master. This command causes all subsequent keyboard input to be sent out Port 2. Thus, at that point (until entry of $NEC on both master and slave, which cancels the effect of both commands), commands executed by the master will also be sent to the slave for execution.

Commands executed directly on the slave in this way will not be executed by the master. Reentry of the toggle character will return terminal control to the maser, resuming command echo to the slave. Command echo allows side-by-side single-stepping of programs on both EVMs, and also provides a means of displaying a breakpoint on the slave if both EVMs receive an EX command. The user should note that whenever using either the $EC or $$EC commands, the overall serial I/O performance is reduced to the baud rate of the slowest EIA port.

2.3.9 User-Changeable Functions

The following paragraphs describe how various EVM features and functions may be altered by the user.

2.3.9.1 EVM Firmware Functions

The automatic EVM firmware functions listed in Table 2-5 may be altered by the user to remove them. This process requires that a new EPROM be created to replace the one in socket U58. The procedure is described following Table 2-5.

TABLE 2-5 - USER-CHANGEABLE EVM FIRMWARE FUNCTIONS

FUNCTION	ADDRESS	CHANGE FROM	CHANGE TO
Delete Form Feed	>0086	>0C00	>0000
Delete Terminal Bell	>0088	>0700	>0000
Unused	>008A		
XON Character	>008C	>11	
XOFF Character	>008D	>13	
PORT 1 Character Format	>008E	>42	See Table 2-6
PORT 2 Character Format	>008F	>42	See Table 2-6
PORT 2 Baud Rate	>0090	9600	110 (Error Default), 300, 600, 1200, 2400, 4800, 19200
Baud Rate Display	>0092	1200	Baud rates equal to or less than this default to minimal display.
Cursor Left Sequence	>0094 to >0096	>0800 to >0000	Up to 3 characters, followed by a byte >00.
Cursor Up Sequence	>0098 to >009A	>0B00 to >0000	Up to 3 characters, followed by a byte >00.
Cursor Down Character	>009C	>0A00	One character in MSB.
Timeout Error Delay	>009E	>0019	1 = Approx .75 seconds
XON/XOFF Disabled at at power up	>00A2	>0000	>FFFF
ACK/NACK Disabled at at power up	>00A4	>0000	>FFFF
Autoexecution after Reset	>00A8	>FFFF	>Max Delay
Transparency mode toggle character	>00A0	>0300	New default toggle char
Number of nulls transmitted after<CR>	>00A6	Initial Zero	To nulls transmitted

The procedure for creating a new EPROM is:

> **CAUTION**
>
> When creating a new EPROM, DO NOT ERASE the existing one until the procedure described below is complete.

1) Move the contents of U58 to RAM by entering:

```
?$MOVE 0 1FFF A000
```

2) Use the $MPM command to modify any locations desired. These locations will be equivalent to the addresses in Table 2-5, except that the most significant address nibble will now be: A.

3) Enter the PROM utility with:

```
?PROM
.
```

4) Place an EPROM in the programming socket and verify that it is erased by entering:

```
.VRFY
VERIFY COMPLETE    (Message returned if EPROM is OK)
```

5) Program the EPROM (all parameters can be defaulted) by entering:

```
.PROG
PROGRAMMING COMPLETE    (Message returned)
.QUIT
?
```

6) Exchange EPROMs, retaining the old one.

2.3.9.2 Changing the Port Character Formats

The character formats for Ports 1 and 2, as described in Section 2.3.5.1, may be altered. Refer to Table 2-6, which describes the TMS9902 Asynchronous Communication Control units format control.

TABLE 2-6 - TMS9902 ASYNCHRONOUS COMMUNICATION CONTROL UNIT FORMAT CONTROLS

	BITS			
STOP BITS	1.5	2	1	1
BIT 7 BIT 6	0 0	0 1	1 0	1 1
PARITY	NONE	NONE	EVEN	ODD
BIT 5 BIT 4	0 0	0 1	1 0	1 1
CLOCK DIVIDE (LEAVE AT ZERO)				
BIT 3 BIT 2	0: Divide clock by 3 1: Divide clock by 4 Unused			
	BITS			
CHARACTER	5	6	7	8
BIT 1 BIT 1	0 0	0 1	1 0	1 1

2.3.9.3 Adding Terminal Control Characters

There are two control character tables stored in the U58 EPROM, one for single characters and one for <ESC> sequences. The <ESC> sequence table is interrogated if a character is received within at least two character times of an <ESC> character. Adding entries to either table will allow the EVM to recognize the characters sent from dedicated keys not already recognized and perform the designated function. These characters will not be echoed to the terminal.

The only cursor characters sent to the terminal are cursor left and cursor up. These are shown in Table 2-5. The process for changing them is described in Section 2.3.9.1. Cursor down is controlled by a line feed character, and cursor right is accomplished by overprinting the contents of the line.

Any control characters added to the existing list of control characters recognized by the EVM will automatically be included in the list of illegal characters for the COMM command.

Changing Single Control Characters: The single control character table stored in EPROM U58 is in the form of:

```
DATA <Character in MSB?<00 in LSB>
DATA <Address of handler>
.
.
.
DATA 0
```

Table 2-7 shows each entry as a pair of data statements, the first with the character (indicated with an *) in the MSB and 00 in the LSB. This is followed by the address of the handler function that character is to perform. The table must end with a word of >0000.

TABLE 2-7 - SINGLE CONTROL CHARACTERS IN EPROM U58

ADDRESS	CONTENTS
* >00A8	<CR>
>00AA	LINE TERMINATOR
* >00AC	RUBOUT
>00AE	DELETE CHARACTER AT END OF LINE
* >00B0	CNTL/N
>00B2	INSERT CHARACTER(S)
* >00B4	CNTL/D
>00B6	DELETE CHARACTER(S)
* >00B8	CNTL/F
>00BA	CURSOR RIGHT
* >00BC	CNTL/L
>00BE	CURSOR RIGHT
* >00C0	CNTL/P
>00C2	CURSOR RIGHT
* >00C4	CNTL/H
>00C6	CURSOR LEFT
* >00C8	CNTL
>00CA	REDUMP LINE
* >00CC	CNTL/^
>00CE	CURSOR HOME
* >00D0	CNTL/A
>00D2	CURSOR HOME
* >00D4	CNTL/E
>00D6	CURSOR TO END OF LINE
* >00D8	CNTL/Y
>00DA	DELETE TO END OF LINE
* >00DC	CNTL/X
>00DE	DELETE ALL OF LINE
* >00E0	<SP>
>00E2	NEXT ENTRY COMMAND TERMINATOR
* >00E4	CNTL/J, <LF>
>00E6	NEXT ENTRY COMMAND TERMINATOR
* >00E8	CNTL/K
>00EA	PREVIOUS ENTRY COMMAND TERMINATOR
* >00EC	CNTL/Z
>00EE	PREVIOUS ENTRY COMMAND TERMINATOR
* >00F0	CNTL/I
>00F2	TAB RIGHT
* >00F4	>0000 (Table Terminator)
>00F6->0114	EXPANSION

When adding entries to the table, the new character is placed over the table terminator (>0000), followed by the address of the routine for that character. The address must be one of the handler addresses already in the table. The table terminator must then be put after the new address. Space is provided for up to eight user-added character/handler address pairs.

<ESC> Sequences: The second table stored in the U58 EPROM is for storing the second character in a two-character <ESC> sequence. This table is searched by the line buffer controller after reception of an <ESC> character from the terminal. The handler ad-

dresses used by this table are the same as those in the single control character table described in the previous subsection. This table is actually a tree of compare and branch instructions of the form:

```
     CI   Rx, <character>
     JNE  X1
     B    @HANDLER
X1   CI...
```

The addresses listed in Table 2-8 are for the <character> location (indicated by an *) in the Compare Immediate instruction, and the address of HANDLER in the Branch instruction. Space is provided in the table for eight user-added characters (one spare character, plus one spare address each). No table terminator is needed, since the entire table is searched each time. If no entry is found, program execution simply continues with the input routine.

TABLE 2-8 - <ESC> SEQUENCES

ADDRESS	CONTENTS
* >0118	CNTL/L
>011E	CURSOR UP
* >0122	I
>0128	BACK TAB
* >012C	CNTL/R
>0132	CURSOR HOME
* >0136	A
>013C	CURSOR UP
* >0140	B
>0146	CURSOR DOWN
* >014A	C
>0150	CURSOR RIGHT
* >0154	D
>015A	CURSOR LEFT
* >015E	H
>0164	CURSOR HOME
* >0168	Spare Character 1
>016E	Spare Address 1
* >0172	Spare Character 2
>0178	Spare Address 2
* >017C	Spare Character 3
>0182	Spare Address 3
* >0186	Spare Character 4
>018C	Spare Address 4
* >0190	Spare Character 5
>0196	Spare Address 5
* >019A	Spare Character 6
>01A0	Spare Address 6
* >01A4	Spare Character 7
>01AA	Spare Address 7
* >01AE	Spare Character 8
>01B4	Spare Address 8

2.3.9.4 Adding Command Chains to the Chain Library

The command chains in the library are stored in ten 48-byte section at the high end of the EPROM in U55. The locations are listed in Table 2-9.

TABLE 2-9 - COMMAND CHAIN LIBRARY LOCATIONS

CHAIN	START ADDRESS	END ADDRESS
0	>7FD0	>7FFF
1	>7FA0	>7FCF
2	>7F70	>7F9F
3	>7F40	>7F6F
4	>7F10	>7F3F
5	>7EE0	>7F0F
6	>7EB0	>7EDF
7	>7E80	>7EAF
8	>7E50	>7E7F
9	>7E20	>7E4F

All unused locations must have a byte of >FF in the start address location, and all used locations must end with a byte of >FF. To add chains to the library, first test the chain thoroughly, then copy the EPROM into RAM with the following instruction:

```
?$MOVE 6000 7FFF A000 <CR>
```

Using the $MPM command and remembering that the most significant address nibble is B after the move, use the ASCII subcommand (see Section 3) to put the new chain into RAM, leaving the other chains and the rest of the EPROM undisturbed. The ASCII subcommand allows entry of ASCII values into byte locatins by enclosing the characters in single quotation marks. For example:

```
?$MPM BFD0<CR>
BFD0 = FFF  "AA' <CR>
BFD0 = 4141
```

Entry of the null string: ," will cause display of the contents in ASCII. For example:

```
BFD0 = 4141 ,"<CR>  AA
BFD0 = 4141
```

The unused slots can be used in any order, and when completed, the new chain will automatically be displayed with the library command CLIB, and will be loaded with the Cx command calling that slot. Once the new chain is entered, place an EPROM in the programming socket, enter the PROM utility with the PROM command, verify that it is erased with the VRFY command, and program it with the PROG command. All parameters in the PROG command may be defaulted.

2.3.10 Monitor Operation

The following subsections discuss some general information concerning monitor operation.

2.3.10.1 Object Code Loading and Dumping

The EVM accepts object code in three formats: TMS7000, TMS9900, and Tektronix. Input files must be at absolute, or load module level. For an explanation of each of these formats, see the appropriate manual listed in Section 1. Attempting to input object files at other than load level may cause the EVM to generate an error for that file.

The EVM outputs object code in two formats: TMS9900 (TMS7000 object code will dump in this format) and Tektronix. The following paragraphs explain each output format.

TMS9900 (TMS7000) Dump Format: If a dump is performed from locations >1234 to >1246, then the TMS9900 dump format will appear as follows:

```
00000PROGRAM 91234BDEADBDEADBDEADBDEADBDEADBDEADBDEADBDEAD7F0BF
BDEADBDEAD7FD29F
:<CR><LF>
```

TMS7000 object code contains an additional data tag, * , to denote an 8-bit byte in the data. A dump from an TMS7000 format would be identical to the example above, except that the header: 00000PROGRAM is replaced by: K0000PROGRAM.

Tektronix Dump Format: If a dump is performed from locations >1234 to >1246, then the Tektronix dump format will appear as follows:

```
/1234140FDEADDEADDEADDEADDEADDEADDEADDEADDEADDEADF4
/00000000<CR><LF>
```

2.3.10.2 RESETs

"Warm" RESETs may be initiated during the execution of a user program and during execution of a command . This feature allows the user to quickly halt the TMS32010; however, this halt is uncontrolled and will clear the internal registers and possibly cause some program or data memory locations to be corrupted.

RESET During Program Execution: If a RESET is performed during execution of a user program, control will immediately return to the monitor top level in the following manner:

```
[ RESET ]

USER HALT -PC RESET   BIO=1

?
```

This "warm" start does not reinitialize baud rates, clock and memory sources, etc. RESET During Command Execution: If RESET is performed during execution of a command (a "warm" start), control will immediately return to the monitor top level in the following manner:

```
[ RESET ]

?
```

This warm start does not reinitialize baud rates, clock, memory sources, etc.

The EVM senses whether power has been cycled between RESETs. If the power has not been cycled and/or the RESET is not a warm start, the breakpoints, event count, and trace locations are cleared and the program memory/clock sources are set to internal. If power has been cycled, the power has been cycled, the following additional functions are performed: the text editor is initialized, the assembler table is reset, terminal tabs are reset, and all TMS32010 registers are zeroed.

If the user wants to perform a power-cycle RESET without actually cycling the power, the monitor RESET command is used as follows:

```
?RESET
ARE YOU SURE? (NO) Y
ENTER <CR>
.
.   (RESET Banner)
.
```

Port 2 (J2) defaults at RESET to 9600 baud. The default is changable.

The RESET command is distinguished from the $BOOT command in that the latter also fills program memory with NOPs and fills data memory with zeros.

2.3.10.3 Using <ESC> or <SP> during Load/Dump Execution

Pressing the space bar <SP> during a dump/display of data freezes the display ONLY if the output is assigned to Port 1. Pressing <SP> again reactivates the dump/display.

If the escape key <ESC> is pressed during execution of any dump, display, or load command, control will immediately return to the top level of the currently executing EVM software (monitor, editor, PROM utility, etc.) in the following manner:

```
<ESC>

?
```

This feature is especially useful during time-consuming loads/dumps because it simply aborts the process without the necessity of having to RESET (and reinitialize). Commands that accept this feature are indicated in the command descriptions in Section 3. Commands which allow the <CR> abort also permit use of the <ESC> feature described in the previous paragraph.

2.3.10.4 Autoexecution After Power-Up Reset (Load-And-Go)

Revision 2.0 allows chain 9 (see "CLIB" and "C9" command descriptions) to be executed automatically at power-up after a reset if the autobaud <CR> is not received before a software timer expires. The value of this timer is initially loaded from locations >00A8->00A9 and starts counting down after the reset switch is pressed. When the autobaud <CR> is received the autoexec process is aborted. If the timer expires before the <CR> is received, the terminal baud rate is set to 19200 baud and chain 9 is automatically loaded and executed.

Chain C9 is initially set to read an eprom in the eprom programming socket, set the PC = 000, init the clock to external, and run the program. The contents of chain 9 can be set to do any sequence of commands up to 48 characters.

The prinicpal use of the autoexec is to allow the evm to be used in a limited fashion in field tests without the requirement of a terminal. The contents of chain 9 as shipped with the software allow a user's program to be read from U29 (the eprom programming socket) and executed after some initialization without the need for a terminal at port 1. After the autoexec timer has expired and the chain is executing, normal terminal output is still enabled and can be watched with a terminal at port 1. Since the baud rate is set a 19200, the time for this terminal output is minimal and the user's program is entered approximately 3 seconds after chain execution starts.

This feature can also be used as a simple autoexec at reset to execute commands usually executed at power-up reset by the user (such as "XON," "ACK," "INIT E," etc.). The user should note that since the baud rate is set to 19200 by the autoexec, the "BAUD1 XXX" command should be executed first in the chain if terminal interaction is desired (see "BAUD1" command description).

2.3.10.4.1 Changing The Autoexecution Delay Value

Each count of 1 in the initial timer value approximately equals a second thus:

```
>0000  NO DELAY

>0060  ABOUT A MINUTE OF DELAY

>FFFF  MAXIMUM DELAY (DEFAULT)
```

Changing the initial delay count requires following the procedure outlined in the section 2.3.9.1 for creating a new U58 monitor eprom. The value is located in:

```
>00A8   MSB OF INITIAL DELAY COUNT
>00A9   LSB OF INITIAL DELAY COUNT
```

2.3.10.4.2 Changing The Autoexecution Chain

The procedure for creating a new chain at the C9 location is described in the section 2.3.9.4. The C9 chain extends from >7E20 to >7E4F.

2.3.10.5 Using Program Breakpoints and TMS320 Interrupts

The breakpoint process on the EVM (including single step) involves running small programs in the TMS320 program space. If the user's application involves interrupts, it is possible to interrupt this breakpoint program in addition to the user's application. The application program will continue to execute properly but if the interrupt occurs during the breakpoint program, one position on the stack (the position after the top of the stack) will be corrupted with the value >0CE.

Using the "RUN" command will allow complete emulation of the TMS320 with no stack limitation.

3. DEBUG MONITOR COMMANDS

3.1 INTRODUCTION

The TMS32010 EVM Monitor Program contains approximately 80 commands and subcommands. This section describes those commands, their usage, and lists and describes monitor error messages. The monitor commands are grouped in the following categories and sub-categories:

- MONITOR COMMANDS
  - Display/Modify Monitor Commands
  - Display/Modify Register Commands
    - Display/Modify Register Subcommands
  - Display/Modify Memory Commands
    - Display/Modify Memory Subcommands
  - Single-Step Commands
    - Single-Step Subcommands
  - Breakpoint Commands
    - Breakpoint Subcommands
  - Trace Commands
    - Trace Subcommands
  - Display/Modify Baud Rate Commands
    - Display/Modify Baud Rate Subcommands
  - Miscellaneous Monitor Commands
- MONITOR MENU DISPLAY COMMANDS
- MONITOR SYSTEM ACCESS COMMANDS

Each instruction within each category is explained in Section 3.3.9.

3.2 MONITOR CONVENTIONS AND FORMATS

The following paragraphs describe the various conventions and formats required to properly enter and execute monitor commands. A full listing of all monitor commands is also included in Appendix B.

3.2.1 Register Definition

Upon initial power-up, the TMS32010 Emulator register contents will be set to zero. Afterwards, all user register memory will remain intact regardless of RESET or execution of breakpoint operations.

3.2.2 Numerical Data

Whenever the monitor is accepting numerical input data, it expects the data to be in either hexadecimal or decimal format. In general, all address inputs are expected to be in hexadecimal. Data for the fill memory commands is also expected in hexadecimal. Data for the program counter, stack and auxiliary registers is expected in hexadecimal. Data for the accumulator, T and P registers is expected in decimal (16 or 32 bits). The command description for each command will detail the input format.

Any entry other than valid hexadecimal or decimal characters will cause a command to abort, and no change will be made to effect any memory modification. If data entry is expected in hex, a greater-than sign (>)is optional. Data entered in decimal may be

preceded by a plus sign (+) to indicate positive numbers; the plus sign is optional and is assumed if none is present. Negative decimal numbers must be preceded by a minus sign.

Commands that display numerical data (either in the form of dumps or for modification) can work in one of three data formats: hexadecimal (HX), signed decimal (SD), and unsigned decimal (UD). In general, commands dealing with program memory will default to HX, while commands dealing with data memory will default to SD. Dump commands (DPM, DDM) will prompt the user for the format on entry of the command. Display/modify commands (MPM, MDM) will take the default value and allow change of the format inside the command. When the format is required as input, the prompt:

```
FORMAT? (XX)
```

is displayed, where XX is the current default value. Entering an <SP> will display:

```
FORMATS: "HX", "SD", "UD"
```

and return to the proper format.

3.2.2.1 Scale Factors

Whenever signed decimal data is displayed, the user has the option of scaling the data by a power of two. The default power of two is zero, yielding a divide-by-one. Legal values of scale factors are from zero (default) to 15 (decimal). Entering a number outside this range defaults the scale factor to zero.

To request a scale factor, the user must enter a comma (,) after the SD format entry, followed by the scale factor. If SD is the default format, the user can immediately enter the ",x" scale factor. The scale factor entered is valid until the command is terminated, at which time the scale factor returns to its default value of zero. When data is scaled, the data is accurate to nine digits to the right of the decimal point. Scaled memory dumps accommodate for the additional digits by spreading the data out. The following are examples of scaling data:

```
FORMAT? (HX) SD,15        Scale data by 2E15, the maximum
                          scale factor

FORMAT? (SD) ,10          Format is defaulted and data is
                          scaled by 1024
```

3.2.3 Display/Modify Procedures

Any time that a display/modify type of command is entered, the user has several options on how to proceed: (1) enter new data, (2. enter a <CR> to terminate the command, or (3) enter a character to either display the previous or the next location. Except for data entry into certain TMS32010 register locations that expect data in signed decimal, all data is expected in hexadecimal format. The format expected by any command allows entry of data not preceded by a character defining the type of data. If the user wants to enter data in a format different from the default, the defining character can be entered. If the defining character entered is the same as for the format expected, it is ignored.

During execution of any display/modify command, the Next Entry characters (<SP>, <LF>, CNTL/V, CNTL/J, cursor down) will access the next location, and the Previous Entry characters (cursor up, CNTL/K,Z) will access the previous location. Each line displayed will be indented two spaces. When a Next Entry command is entered, a caret (%) will be displayed to indicate reverse motion.

3.2.4 Command Parameters

If parameters are required for a command, they must be entered in proper order. Failure to enter the required number of parameters in the proper format (unless defaults are acceptable) will result in a "VALUE ERROR" message being displayed.

A command must be separated from the first parameter by a space <SP> or a comma. Each parameter must also be separated from the previous parameter by a space or a comma. Thus, the general format for a command is:

```
?CMD PARM1 PARM2 PARM3 <CR>
```

For example, the Move Memory command (MOVE) requires three param It would be entered as follows:

```
?MOVE 0 10 20<CR>     OR:     ?MOVE 0,10,20<CR>
```

As another example, the Execute command (EX) accepts optional parameters. It may be entered as follows:

```
?EX<CR>
?EX 2<CR>
?EX 2 2<CR>          OR:      ?EX 2,2,<CR>
```

3.2.5 Fill/Find Commands ASCII Parameter Entry

The fill and find commands described in Section 3.3.9 support entry of the last parameter in the default (hexadecimal) format. Signed decimal numbers can also be entered if preceded by a plus or minus sign (+ /-). The monitor commands affected are:

```
FBPM     $FBPM
FWPM     $FWPM
FBDM
FPM
FDM      $FPM
```

These commands also support direct entry of ASCII characters as either the fill or find character. This is accomplished enclosing the character in single quotation marks. For example:

```
?FBPM 0 FFF +10     Will search program memory for decimal 10

?FBPM 0 FFF A       Will search program memory for decimal 10

?FBPM 0 FFF "A'     Will search program memory for >41
```

If several characters are entered between the single quote marks, a byte command will use only the last character entered, and a word command will only use the last two characters entered, using the ASCII value of each entry as a byte of data. If one character is entered for a word command, the MSB of the word is >00 and the character is used as the LSB. If no characters are inserted between the single quotes, the value used is zero.

3.3 MONITOR COMMANDS

The monitor commands allow the user to control the hardware and software functions of the monitor and monitor program. This section contains detailed descriptions of the monitor commands.

3.3.1 Display/Modify Monitor Commands

The display/modify monitor commands cause a display of the specified information on the system monitor for user reference and/or modification. Table 3-1 lists the Display monitor commands. A description of each command is in Section 3.3.9.

TABLE 3-1 - DISPLAY/MODIFY MONITOR COMMANDS

COMMAND	MNEMONIC
Display Monitor Commands Menu	HELP
Display Keyboard Entry Aids	KHELP
Display Monitor Error Messages	EHELP
Display/Modify Cursor Control Characters	MCC
Display Monitor Menu	MENU
Display/Modify Clock/Memory Source	INIT
Display/Modify Tabs	TABS
16-Bit Unsigned Hex-Decimal Conversion	UD16
32-Bit Unsigned Hex-Decimal Conversion	UD32
16-Bit Signed Hex-Decimal Conversion	SD16
32-Bit Signed Hex-Decimal Conversion	SD32
16-Bit Decimal-Hex Conversion	HX16
32-Bit Decimal-Hex conversion	HX32
Display Scaled 16-Bit Decimal Number	SCALE
Save/Show Machine State	SMS
Load Machine State	LMS
Save Program Memory	SPM
Load Program Memory	LPM
Display Program Memory	DPM
Display Data Memory	DDM
Display TMS32010 Register Set	STATE
Clear TMS32010 Register Set	CLEAR
Display ACC/T Reg/P Reg in Hex	HATP

3.3.2 Display/Modify Register Set Commands

The commands to display and/or modify the TMS32010 register set are accessed as follows: once the register name has been entered and the register contents displayed and modified (if desired), subsequent register locations can be accessed with a Next Entry character (<SP>, <LF>, CNTL/J,V or cursor down) or a Previous Entry character (<CNTL/K,Z or cursor up). Thus, all register locations are available in a bidirectional, circular fashion, with entry at any point.

Table 3-2 lists the register display/modify commands and subcommands. An explanation of each command is in Section 3.3.9.

TABLE 3-2 - DISPLAY/MODIFY REGISTER SET COMMANDS

COMMAND	MNEMONIC
Display/Modify Accumulator	ACC
Display/Modify T Register	TREG
Display/Modify P Register	PREG
Display/Modify Auxiliary Register 0	AR0
Display/Modify Auxiliary Register 1	AR1
Display/Modify Program Counter	PC
Display/Modify Overflow Flag	OV
Display/Modify Overflow Mode	OVM
Display/Modify Data Page Pointer	DP
Display/Modify Auxiliary Register Pointer	ARP
Display/Modify Stack Locations	STACK
SUBCOMMAND	
This Menu	M
Redisplay Register	R
Redisplay Opposite Format	,x
Next Entry	<SP>, <LF>, CNTL/J, CNTL/V or Cursor Down
Previous Entry	CNTL/K,Z or Cursor Up

3.3.2.1 Display/Modify Register Set Subcommands

The subcommands may be entered in place of, or in addition to, data. The menu of subcommands is accessed by entering ",M", resulting in the following display:

```
,M      THIS MENU
,R      REDISPLAY REGISTER
,x      REDISPLAY OPPOSITE FORMAT
<SP>,<LF>,CNTL/J, CNTL/V, CURSOR DOWN      Next Entry
CNTL<K,Z>,CURSOR UP                        Previous Entry
<CR>    QUIT
```

If entered in addition to data, the data is first stored and then the subcommand is executed. Occurrence of any error during entry of a subcommand will cause control to return to the monitor top level. The Next Entry/Previous Entry/QUIT commands allow sequencing of locations in any direction and normal command termination (return to the monitor). The subcommands cause the following functions:

,M Displays the subcommand menu and then redisplays the line that the command was entered on.

,R Allows redisplay of the just entered data without having to advance to the next entry and then back up by one.

,x (where "x" is any alphanumeric character except M or R) Allows redisplay of the current register in signed decimal if the default for that register is hexadecimal, and in hexadecimal if the default for that register is signed decimal. Any alphanumeric character except M or R will execute this command. Since the status register locations are only one bit wide and have the same display in both formats, those locations treat this command as an <SP> entry.

3.3.3 Display/Modify Memory Commands

The commands and subcommands to display and/or modify the TMS32010 program and data memory are listed in Table 3-3. Equivalent functions for the separate memories are provided by similar commands. A description of each command is in Section 3.3.9.

TABLE 3-3 - DISPLAY/MODIFY MEMORY COMMANDS

COMMAND	MNEMONIC
Display/Modify Data Memory	MDM
Find Word in Data Memory	FWDM
Find Byte in Data Memory	FBDM
Display/Modify Program Memory	MPM
Find Word in Program Memory	FWPM
Find Byte in Program Memory	FBPM
SUBCOMMAND	
This Menu	,M
Quit 'FIND' Byte/Word in Program/Data Memory	,Q
Change Display Format	,F
Redisplay Current Value	,R
Display Current Value in Hexadecimal	,H
Display Current Value in Signed Decimal	,S
Display Current Value in Unsigned Decimal	,D
Display Current Value in Binary	,B
Enter New 'MPM'/'MDM' Address	,A =
Display Current Value in ASCII	, "
Modify Current Value in ASCII	,'xx'
Next Entry	<SP>,<LF>,CNTL/J, CNTL/V or Cursor Down
Previous Entry	CNTL/K,Z or Cursor Up
Quit MXM/'FIND' Next Entry	<CR>

3.3.3.1 Display/Modify Memory Subcommands

The subcommands may be entered in place of, or in addition to, data. The menu of subcommands is accessed by entering ",M", resulting in the following display:

```
,M      THIS MENU
,Q      QUIT "FIND' BYTE/WORD IN PROGRAM/DATA MEMORY
,F      CHANGE DISPLAY FORMAT
,R      REDISPLAY CURRENT VALUE
,H      DISPLAY CURRENT VALUE IN HEXADECIMAL
,S      DISPLAY CURRENT VALUE IN SIGNED DECIMAL
,D      DISPLAY CURRENT VALUE IN UNSIGNED DECIMAL
,B      DISPLAY CURRENT VALUE IN BINARY
,A=     ENTER NEW MPM/MDM ADDRESS
,"     DISPLAY CURRENT VALUE IN ACII
,'xx'  MODIFY CURRENT VALUE IN ASCII
<SP>,<LF>,CNTL/J, CNTL/V, CURSOR DOWN      Next Entry
CNTL(K,Z),CURSOR UP                        Previous Entry
<CR>    QUIT MXM/'FIND'                    Next Entry
```

If entered in addition to data, the data is first stored and then the subcommand is executed. Occurrence of any error during entry of a subcommand will cause control to return to the monitor top level. The Next Entry/Previous Entry/QUIT commands allow sequencing of locations in any direction and normal command termination (return to the monitor). The subcommands cause the following functions:

,M Displays the menu and then redisplays the line that the command was entered on.

,Q Used to terminate a Find Data command, since the normal command terminator (<CR>) is used to initiate the next find sequence.

,F Allows change of default display/modify format. Defaults are signed decimal for data memory, and hexadecimal for program memory. If signed decimal is specified, the user can specify a scale factor (see Section 3.2.2.1). Execution of the ,F command from a Find Data command will cause the currently executing command to become the equivalent display/modify command. The formats are: HX (hexadecimal), SD (signed decimal), and UD (unsigned decimal).

,R Allows display of the just entered data without having to advance to the next entry and then back up one.

,H Displays the contents of the current location in hexadecimal, regardless of default format.

,S Displays the contents of the current location in signed decimal, regardless of the default format. This subcommand also incorporates the scale factor option available for signed decimal data. Default scale factor is 0, and the legal range is 0 to 15. Entering a scale factor outside this range causes a default to zero. the scale factor is entered immediately after the ,S command as Sx, with no space or comma between. The scale factor is used only for the subcommand with which it is entered; any scale factor entered previously intact. The user must enter the ,F command to change the scale factor for the command.

,U Displays the contents of the current location in unsigned decimal, regardless of default format.

,B Displays the contents of the current location in binary, regardless of the default format. A space is inserted between each nibble.

,A = Allows large jumps in location without having to exit the command and reenter with a new starting location. The user must observe maximum address values for data memory (0 to >8F) if currently executing a data memory command, and for program memory (0 to>FFF) if currently executing a program memory command. Execution of the ,A = command from a Find Data command will cause the currently executing command to become the equivalent display/modify command. For example:

```
FBPM would become: MPM
FWDM would become: MDM
```

," Displays the contents of the current location as two ASCII characters. If either byte is out of the displayable ASCII range >20 to >>F), it is displayed as a space.

,'xx' Changes the contents of the current location to the hexadecimal value of the last two ASCII characters entered. If only one ASCII character is entered then it is right justified with the MSB = 0.

<CR> In a display/modify command, the <CR> returns control to the monitor. In a Find Byte/Word command, the <CR> initiates the search for the next entry.

3.3.4 Single-Step Commands

Table 3-4 lists the single-step commands and subcommands. A description of each command is in Section 3.3.9.

TABLE 3-4 - SINGLE-STEP COMMANDS

COMMAND	MNEMONIC
Fill Data Memory	FDM
Fill Program Memory	FPM
Fill Program Memory with NOPs	NOP
Move Program Memory	MOVE
Execute User Program with Breakpoints	EX
Execute User Program without Breakpoints	RUN
Single-Step User Program	SS
SUBCOMMAND	
This Menu	M
Change Display Type	Tx
Change Display Format	F
Enter Step Count	C
Display Program Counter	P
PC Display Range	R
List to Port 1	1
List to Port 2	2
Execute One Single-Step	<SP>
Return to Monitor	<CR>

3.3.5 Breakpoint Commands

Table 3-5 lists the breakpoint commands and subcommands that display and/or modify breakpoints. A description of each command is in Section 3.3.9.

TABLE 3-5 - BREAKPOINT COMMANDS

COMMAND	MNEMONIC
Set Breakpoints	SB
Clear One or All Breakpoints	CB
Display Breakpoints	DB
Set Event Count For One Breakpoint	EC
SUBCOMMAND	
This Menu	,M
Clear Breakpoint	,C
Redisplay Breakpoint	,R
Display Event Counter	,E
Next Entry	<SP>,<LF>,CNTL/J, CNTL/V, or Cursor Down
Previous Entry	CNTL/K, CNTL/Z or Cursor Up
Quit	<CR>

3.3.6 Trace Commands

Table 3-6 lists the trace commands and subcommands. A description of each command is in Section 3.3.9.

TABLE 3-6 - TRACE COMMANDS

COMMAND	MNENOMIC
Set Single-Step Trace Line Locations	ST
Clear One or All Trace Locations	CT
Display Trace Locations	DT
SUBCOMMAND	
This Menu	,M
Clear Field	,C
Redisplay Field	,R
Next Entry	<SP>,<LF>,CNTL/J,CNTL/V, or Cursor Down
Previous Entry	CNTL/K,Z or Cursor Up
Quit	<CR>

3.3.6.1 Trace Subcommands

The subcommands may be entered in place of, or in addition to, data. The menu of subcommands is accessed by entering ",M", resulting in the following display:

```
,M      THIS MENU
,C      CLEAR FIELD
,R      REDISPLAY FIELD
<SP>,<LF>,CNTL/J,CNTL/V,CURSOR DOWN      Next Entry
CNTL(K,Z),CURSOR UP                  Previous Entry
<CR>    QUIT
```

If entered in addition to data, the data is first stored and then the subcommand is executed. Occurrency of any error during entry of a subcommand will cause control to return to the monitor top level. The Next Entry/Previous Entry/QUIT subcommands allow sequencing locations in any direction and a return to monitor when the command has executed. The other commands function as follows:

,M Displays the menu and then redisplays the line that the command was entered on.

,C Clears the current trace location. Default = 0, not 1, although the function is the same (one step). In addition, any count entered is used as the next default until the SS command is terminated.

,R Allows redisplay of the just entered trace location without having to advance to the next entry and then backing up one.

3.3.7 Display/Modify Baud Rate Commands

The EVM contains two TMS9902 UART chips, one for each EIA port. The baud rate of Port 1 (the terminal) is determined automatically after RESET by striking the <CR> on the terminal keyboard. The baud rate of Port 2 defaults to 9600 baud. Both port baud rates may be changed by using the commands listed in Table 3-7. A description of each command is in Section 3.3.9.

TABLE 3-7 - DISPLAY/MODIFY BAUD RATE COMMANDS

COMMANDS	MNEMONIC
Display/Modify Port 1 Baud Rate	BAUD1
Display/Modify Port 2 Baud Rate	BAUD2
SUBCOMMANDS	
This Menu	,M
Redisplay Baud Rate	,R
Display Current Value of Control Byte of the EIA Port	,D
Modify Control Byte of the EIA Port	,C =

Note: Allowable baud rates will be displayed at the bottom of the menu as follows:
Baud Rates: 110, 300, 600, 1200, 2400, 4800, 9600, 19200

3.3.7.1 Baud Rate Subcommands

The subcommands may be entered in place of, or in addition to, data. The menu of subcommands is accessed by entering ",M", resulting in the following display:

```
,M      THIS MENU
,R      REDISPLAY BAUD RATE
,D      DISPLAY CHARACTER FORMAT
,C=       CHANGE CHARACTER FORMAT
BAUD RATES:  110, 300, 600, 1200, 2400, 4800, 9600, 19200
```

If entered in addition to data, the data is first stored and then the subcommand is executed. Occurrence of any error during entry of a subcommand will cause control to return to the monitor top level. The commands function as follows:

,M Displays the menu and then redisplays the line that the command was entered on.

,R Allows redisplay of the baud rate just entered.

The next two subcommands allow display/modification of the format control byte for each EIA port. Table 2-6 detail how the contents of this byte control the character format. Both EIA ports are initially set to one start bit, seven data bits, two stop bits, and no parity bits. The user has the option of either creating a new EPROM (see Section 2.3.9.1) to change these power-up default values, or of executing the ,C = command each time EVM power is cycled.

,D Causes a display in hex and binary of the current value of the control byte of the port. The control byte dictates the format of characters transmitted/received at that EIA port. The power-up default value of the control byte is shown in Table 2-5.

,C = Allows modification of the control byte of the EIA port. The byte is entered as a hexadecimal value. See Table 2-6 to configure the control byte. No entry aborts the subcommand. Execution of this commands automatically causes execution of the ,D command described above.

3.3.8 Miscellaneous Monitor Commands

Table 3-8 lists the miscellaneous monitor commands. A description of each command is in Section 3.3.9

TABLE 3-8 - MISCELLANEOUS MONITOR COMMANDS

COMMAND	MNEMONIC
Print Formfeed Character to EIA Port	FF
Display Files Stored on Audio Cassette	DIR
Enable Audio Cassette Motor	MO
Display Assembler Label Table	TABLE

3.3.9 Monitor Command Definitions

The monitor commands on the following pages are defined and described in detail. They are listed in alphabetical order.

Operands None

Syntax A[CC]

Purpose To display and/or modify the contents of the accumulator.

Notes

1) The accumulator is displayed as a signed 32-bit decimal number with input expected in signed decimal.

2) Entering a preceding plus sign (+) is optional, since it is assumed. Negative numbers must be preceded by a minus sign (-). Entering values in hex is allowed if a greater-than sign (>) precedes the entry.

Example

```
?ACC                          Accumulator is displayed,
  ACC = +44 -122,R            modified, and redisplayed.
  ACC = -122 <CR>

?
```

Operands None

Syntax AR0

Purpose To display and/or modify the contents of auxiliary register 0.

Notes

1) The auxiliary register is displayed as an 8-bit hexadecimal number.

2) Entry of data is assumed to be in hexadecimal within a range from >0 to >8F (+143), but decimal values can be entered by preceding the entry with a plus (+) sign.

Example

```
?AR0                          The contents of the auxiliary
  AR0 = 008F 60,R                     register are displayed, modi-
  AR0 = 0060 <CR>               fied, and redisplayed.

?
```

Operands None

Syntax AR1

Purpose To display and/or modify the contents of auxiliary register 1.

Notes

1) The auxiliary register is displayed as an 8-bit hexadecimal number.

2) Entry is assumed to be in hex within a range of from >0 to >8F (+143), but decimal values can be entered by preceding them with a plus sign (+).

Example

```
?AR1                    Auxiliary register 1 is displayed
  AR1 = 0040 <CR>       in hexadecimal.

?
```

Operands None

Syntax ARP

Purpose To display and/or modify the auxiliary register pointer.

Note The auxiliary register pointer is a single bit of the status register, and is changed by the least significant bit of the entered value.

Example

```
?ARP                    The auxiliary register pointer bit is
  ARP = 0 1,R           displayed, modified, and redisplayed.
  ARP = 1 <CR>

?
```

Operands None

Syntax BAUD1

Purpose To display and/or modify the baud rate of Port 1 (the terminal).

Note The baud rate is displayed and modified in the actual decimal baud rate values. This is accomplished by first entering the command. The terminal will display the current baud rate, and the user then enters the new baud rate desired, as shown in the example below. Pressing the <ESC> key will cause the monitor prompt to display at the new baud rate.

Example

```
BAUD1                                  The terminal baud
PORT1 BAUD RATE = 19200 9600<CR>       rate is changed
                                       without RESET.
[ CHANGE TERMINAL BAUD RATE ]
<ESC>
?
```

Operands None

Syntax BAUD2

Purpose To display and/or modify the baud rate of Port 2.

Note The baud rate is displayed and modified in the actual decimal baud rate values. This is accomplished by first entering the command. The terminal will display the current baud rate, and the user then enters the new baud rate desired, as shown in the example below. Pressing the <ESC> key will cause the monitor prompt to display at the new baud rate.

Example

```
?BAUD2                          The Port 2 baud rate is
PORT2 BAUD RATE = 9600 300,R    changed.
PORT2 BAUD RATE = 300 <CR>

?
```

Operands Breakpoint number (1 to 8.)

Syntax CB [breakpoint number]

Purpose To allow the user to selectively clear a breakpoint or clearall breakpoints without entering the SB command.

Notes

1) Legal range of breakpoint numbers is from 1 (default) to 8.

2) Entering a zero or an A (for ALL) will clear all breakpoints and the event counter whether any breakpoints are set or not.

3) Clearing a breakpoint will cause a BPx = CLEARED message to display; attempting to clear a breakpoint that is not set will cause a BPx = NOT SET message to display.

4) In the event counter is set for a breakpoint, clearing that breakpoint or all the breakpoints will also clear the event counter.

Example

```
?CB A                          All breakpoints are cleared.
BREAKPOINTS ARE CLEARED
EVENT COUNT CLEARED

?
```

Operands Trace location

Syntax CT [trace location]

Purpose To allow the user to selectively clear trace locations or to clear all trace locations without having to enter the ",C" subcommand of the ST command.

Notes

1) Legal range of trace location numbers is from 1 (default) to 6; entering a zero or an A (for ALL) will clear all trace locations whether or not any are set.

2) Clear a trace location will cause display of a TRACEx = CLEARED message.

3) Attempting to clear a trace location that is not set will cause display of a TRACEx = NOT SET message.

Example

```
?CT A                    All trace locations are
TRACE CLEARED            cleared.

?
```

Operands None

Syntax DB

Purpose To display all set breakpoints on one line of the terminal.

Notes

1) Only breakpoints that are set are displayed; if no breakpoints are set, nothing is displayed.

2) Breakpoints are program memory locations from >0 to >FFF, and are displayed in hexadecimal.

Example

```
?DB                                  All set breakpoints
BP1=040  BP4=503  BP7=033            are displayed.

?
```

Operands Output port (optional) default = 1

Syntax DIR [output port]

Purpose To display the files stored on an audio tape recorder cassette connected to Port 3 at either Port 1 or Port 2.

Notes

1) The audio tape directory is printed by file name and type. As the file passes by the read heads, the number of blocks (512 bytes/block) are counted.

2) Termination of directory listing is by pressing the <ESC> key or by RESET.

3) This command is discussed in detail in Section 8.

Example

```
?DIR 1                    The listing of the audio tape
                          directory is assigned to  and
AUDIO TAPE DIRECTORY      displayed at the terminal.

5      TEST.SOURCE
12     TEST1.OBJECT
3      TEST1.STATE

<ESC> OR < RESET >

?
```

Operands None

Syntax DP

Purpose To display and/or modify the data page pointer.

Note The data page pointer is a single bit of the status register, and is changed by the least significant bit of the entered value.

Example

```
?DP                 The data page pointer bit is displayed,
  DP = 0 1,R        modified, and redisplayed.
  DP = 1 <CR>

?
```

Operands

1) Memory display start address
2) Memory display end address
3) Output port 1 or 2 (optional) default = 1

Syntax

DPM <start address> <end address> [output port]

Purpose

To display the contents of a selected block of program memory for examination by the user.

Notes

1) Legal values for output port are 1 and 2 only.

2) Failure to enter either address parameter will cause a VALUE ERROR message to display. Entering an end address greater than the start address will cause an ADDRESS ERROR message to display.

3) Valid address range is from >0 to >FFF. When entering address parameters, only the last four digits are accepted and leading zeros are assumed.

4) While the dump is in progress, it may be stopped and started by pressing the space bar, and can be aborted at any time by pressing the <ESC> key.

5) The display may be in either hexadecimal (HX), signed decimal (SD), or unsigned decimal (UD) format.

Example

```
?DPM 30 100
FORMAT? (HX) SD

[ PROGRAM MEMORY ]

?
```

User has requested a display to the terminal of a section of program memory in signed decimal format.

Operands None

Syntax DT

Purpose To display all set trace locations and their contents on one line at the terminal.

Notes

1) Only trace locations that are set are displayed.

2) The trace locations are data memory addresses from >0 to >8F, displayed in hexadecimal.

3) If no trace locations are set, nothing is displayed.

Example

```
?DT                                Display all trace locations.

TRACE1=40  TRACE2=41  TRACE4=82

?
```

Operands Breakpoint number

Syntax EC [breakpoint number]

Purpose To allow the user to display and/or modify the event count associated with an existing breakpoint.

Notes

1) The event count for a breakpoint is the number of times that breakpoint is to be encountered before execution is halted and the display given.

2) Entry is expected as a decimal number from 1 to 255. Entering zero or a number larger than 255 clears the event counter. Entering zero for the breakpoint number clears theevent counter whether it is set or not.

3) Entering a breakpoint number will allow an event count to be entered only if that breakpcint already exists. If not, the message BPx NOT SET will display.

4) The SB command must be used to set a breakpoint.

5) Defaulting the breakpoint number will cause the event counter and the associated breakpoint number (if it is set) to display. If an associated breakpoint number is not set, the EVENT COUNT NOT SET message will display (because the event counter will not set if the specified breakpoint does not already exist).

6) When entering the event count, terminating the entry with a <SP> will redisplay the event count. Termination with a <CR> will return control to the monitor command handler.

7) The user should note that since the breakpoint is processed for each occurrence of the event, overall execution is not in real time.

8) If a breakpoint other than the one associated with the event count is encountered before the event count has reached zero, the event counter is reset to its original value. When the event counter reaches zero, the breakpoint is displayed with the message EVENT COUNT to indicate that the event count breakpoint was processed.

Example

```
?EC 1                    An existing event count is
EC1 = 200 100<SP>        displayed and changed.
EC1 = 100 <CR>

?
```

Operands Output port (optional) default = 1

Syntax E[HELP] [output port]

Purpose To display the monitor error messages for reference by the user.

Notes

1) This command may be executed by either typing E, or the full word.

2) Legal values for output port are 1 and 2.

3) Monitor error messages are discussed in Section 3.6.

Example

```
?EHELP              Monitor error messages are displayed
    .                 at the terminal.
    .
( Menu )
    .
    .

  ?
```

Operands Breakpoint display type 0 to 4 (optional) default = 0

Syntax EX [breakpoint display type]

Purpose To execute the user program with the breakpoints specified by the SB command.

Notes

1) The display type selects the display sent to the terminal when a breakpoint is encountered. Display types are:

0 (Default) ACC, T, P
1 All internal registers
2 Type 0 plus trace line set using ST
3 Type 1 plus trace line set using ST
4 Trace line set using ST
5 PC/BIO/Interrupt/mnemonic only

Attempting to enter a display type of 2 to 4 is only allowed if a trace line has been defined by the ST command. If displaying the trace line, the display format is prompted for prior to execution. Since trace line displays data memory locations, default is signed decimal format.

2) If no breakpoints are set, or if none of the set breakpoints is encountered, the EX command functions the same as the RUN command.

3) If a breakpoint is encountered, no matter what display is selected, the following is displayed:

- The PC, in hexadecimal, equivalent to the breakpoint stress (this instruction has not been executed).
- A breakpoint halt banner message.
- The state of the BIO pin.
- Whether or not an interrupt occured (indicated by an INTERRUPT message).
- The mnemonic of the instruction at the breakpoint address

4) If a RESET is performed before an active event count has expired, the remaining event count is desplayed. Use of the RESET function to stop execution causes an uncontrolled halt of the TMS32010 and will save the internal registers except the PC. Display types are:

0 (Default) ACC, T, P
1 All internal registers
2 Type 0, plus trace line
3 Type 1, plus trace line
4 Trace line
5 PC/BIO/INTERRUPT/mnemonic only

Attempting to enter a display type of 2 to 4 is only allowed if a trace line has been defined by the ST command. If displaying the trace line, the display format is prompted for prior to execution. Since trace line displays data memory locations, default is in signed decimal.

Previously, the RESET switch on the EVM was the only way to terminate user program execution when a breakpoint was not set. The escape key (<ESC>) will also terminate program execution in the same way as the RESET switch, allowing full control of the EVM from the keyboard. Since this is an uncontrolled halt of the TMS320, the program counter value is not retrieved but all other contents of the machine state are saved.

Examples

1.
```
?EX                 A breakpoint is  encountered
                    at PC = >400.
BREAKPOINT HALT

PC = 400   BIO=0   MNEMONIC---> ZAC
ACC = +90  TREG = +2  PREG = 2122

?
```

2.
```
 ?EX 4                      A breakpoint is executed and
FORMAT? (SD)<CR>            the trace line is displayed.

BREAKPOINT HALT

PC = FE1   BIO=1   MNEMONIC---> LACK 1
88=-12  44=+27

?
```

3.
```
 ?EX                            User RESET  before  event
                                count expired.
[ RESET ]

USER HALT -PC RESET   BIO=1
REMAINING EVENT COUNT = 44

?
```

Operands

1) Start search address (begin limit: >0)
2) End search address (end limit: >FF)
3) Target byte (optional) default = >FF

Syntax

FBDM <start address> <end address> [target byte]

Purpose

To search data memory for an 8-bit value.

Notes

1) The legal address range is >0 to >8F, with decimal entry allowed if preceded by a plus sign (+).

2) The target word parameter assumes leading zeros for hexadecimal entry. Decimal entry is allowed if preceded by a plus or a minus sign.

3) Data memory is searched on a byte-by-byte basis, two bytes per word. If a match is found within the address range entered, the MDM command is entered. If no match is found, nothing is displayed. Once in the MDM command, the location of the target word or any location around it can be displayed and/or modified.

4) Entering a <CR> will resume the search process, starting at the address last displayed (see Subcommands, Section 3.3.3.1.

5) ASCII target bytes are allowed (see Section 3.2.5).

Example

```
?FBDM  1E 7F 80              The byte >80 is found in the
  27 = 801E<CR>              MSB of location >27, and  in
  70 = 0080<CR>              the LSB of location >70.

?
```

Operands

1) Start search address (start limit: >0)
2) End search address (end limit: >FFF)
3) Target byte (optional) default = >FF

Syntax

FBPM <start address> <end address> [target byte]

Purpose

To search program memory for a 16-bit value contained within a word boundary.

Notes

1) The legal address range is >0 to >FFF, with decimal entry allowed if preceded by a plus sign (+).

2) The target word parameter assumes leading zeros for hexadecimal entry. Decimal extry is allowed if preceded by a plus or a minus sign.

3) Data memory is searched on a byte-by-byte basis, two bytes per word. If a match is found withing the address range entered, the MPM command is entered. If no match is found, nothing is displayed. Once in the MPM command, the location of the target word or any location around it can be displayed and/or modified.

4) Entering a <CR> will resume the search process, starting at the address last displayed (see Section 3.3.3.1.

5) ASCII target bytes are allowed (see Section 3.2.5).

Example

```
?FBPM 0 200 7F        The byte >7F is located in the NOP
  088 = 7F80 <CR>     opcode MSB and also in a data
  1FE = 117F <CR>     statement LSB.

?
```

Operands

1) Memory start address (start limit: >0)
2) Memory end address (end limit: >8F)
3) Fill word (optional) default = >FFFF

Syntax

FDM <start address> <end address> [fill word]

Purpose

To fill data memory between the specified addresses with a given value.

Notes

1) Legal address range is >0 to >8F. Decimal addresses are allowed if preceded by a plus or a minus sign.

2) The fill word parameter entry is expected in hexadecimal, but decimal entry is allowed if preceded by a plus or minus sign.

3) ASCII fill words are allowed (see Section 3.2.5).

Example

```
?FDM 0 +143 0              Data memory is cleared.

?
```

Operands Output port 1 or 2 (optional) default = 2

Syntax FF (output port)

Purpose To allow continuous forms to be used on a line printer connected to Port 2.

Note The character sequence is:

<FF> = >0C
<CR> = >0D

Example

```
?FF                    The formfeed character is sent
                       to Port 2.
?
```

Operands

1) Memory start address (start limit: >0)
2) Memory end address (end limit: >FFF)
3) Fill word (optional) default = >FFFF

Syntax

FPM <start address> <end address> [fill word]

Purpose

To fill program memory between the specified addresses with a given value.

Notes

1) Legal address range is >0 to >FFF. Decimal addresses are allowed if preceded by a plus sign or a minus sign.

2) Since it fills memory, this command initializes the text editor.

3) ASCII fill words are allowed (see Section 3.2.5).

Example

```
?FPM 0 FFF 0          Program memory is cleared.

?
```

Operands

1) Start search address (start limit: >0)
2) End search address (end limit: >8F)
3) Target word (optional) default = >FFFF

Syntax

FWDM <start address> <end address> [target word]

Purpose

To search data memory for a 16-bit value contained within a word boundary.

Notes

1) The legal address range is >0 to >8F, with decimal entry allowed if preceded by a plus sign (+).

2) The target word parameter assumes leading zeros for hexadecimal entry (i.e., entering >FF will search for >00FF). Decimal entry is allowed if preceded by a plus sign or a minus sign.

3) If a match is found within the address range entered, the MDM command is entered. Once in the MDM command, the target word or any location around it can be displayed and/or modified.

4) Entering a <CR> will resume the search process, starting at the address last displayed (see subcommands, Section 3.3.3.1.

5) ASCII target words are allowed (see Section 3.2.5).

Example

```
?FWDM 10 70 -1          All occurrences of -1 between
  64 = -00001 <CR>      the addresses specified have
  6C = -00001 <CR>      been found.

?
```

Operands

1) Start search address (start limit: >0)
2) End search address (end limit: >FFF)
3) Target word (optional) default = >FFFF

Syntax

FWPM <start address> <end address> [target word]

Purpose

To search program memory for a 16-bit value contained within a word boundary.

Notes

1) The legal address range is >0 to >FFF, with decimal entry allowed if preceded by a plus sign (+).

2) The target word parameter assumes leading zeros for hexadecimal entry (i.e., entering >FF will cause a search for >00FF).

3) If a match is found within the address range entered, the MPM command is entered. Once in the MPM command, the location of the target word or any location around it can be displayed and/or modified.

4) Entering a <CR> will resume the search process, starting at the address last displayed (see Section 3.3.3.1.

5) ASCII target words are allowed (see Section 3.2.5).

Example

```
?FWPM 0 FFF 7F80          All occurrences of the NOP
  2F0 = 7F80 <CR>         opcode in program memory are
  38C = 7F80 <CR>         found.
  441 = 7F80 <CR>

?
```

Operands Output port 1 or 2 (optional) default = 1

Syntax H[ELP] [output port]

Purpose To display the EVM monitor commands and their functions for user reference.

Note This command can be executed by either typing H, or entering the full word.

Example

```
?HELP 1          Monitor Commands Menu is displayed at
  .              Port 1 (the terminal).
  .
( Menu )
  .
  .
?
```

Operands None

Syntax HX16

Purpose To convert a 16-bit decimal number to its hex equivalent.

Notes

1) The HX16 command prompts the user for decimal input.

2) User must not enter more than five digits and leading zeros are assumed.

3) Decimal numbers outside the range +32767 to -32768 are not allowed. Either violation will cause a VALUE ERROR message.

4) Data may be entered with either a <CR> or a <SP>. The converted number is then displayed on the same line, and the user is prompted for another input. If no entry is made, <CR> or <SP> terminates the command.

Example

```
?HX16
DEC INPUT? 23456<CR> = >5BA0<CR>
DEC INPUT?<CR>

?
```

Operands None

Syntax HX32

Purpose To convert a 32-bit decimal number to its hex equivalent.

Notes

1) The HX32 command prompts the user for decimal input. No more than ten digits can be entered and leading zeros are assumed.

2) Decimal numbers outside the range +2137483647 to -2147483648. Either violation will cause a VALUE ERROR message to display.

3) Data entered may be terminated with either a <CR> or a <SP>. The converted number is then displayed on the same line, and the user is prompted for another input. If no entry is made, <CR> or <SP> terminates the command.

Example

```
?HX32
DEC INPUT? 15732735<SP> = >0F00FFF<SP>
DEC INPUT? <SP>

?
```

Operands None

Syntax INIT

Purpose To set the EVM emulator clock source and/or program memory to either internal or external.

Notes

1) The clock and memory source default at RESET to the internal setting. This means that the EVM is supplying the clock used by the target system and the 4K words of program memory are on-board the EVM.

2) If external memory is specified, the EVM program memory display/modify commands (MPM, DPM, FBPM, FWPM, SPM, LPM, FPM AND NOP) cannot be used with the memory on the target system.

3) Selecting external clock also selects external BIO and INT, thus enabling BIO and INT signals to reach the on-board TMS32010 from the target connector.

4) If external CLOCK is selected just to get external BIO and INT, the internal 20 MHz clock can be jumpered to the external clock position in the EVM by connecting positions 2-3 of the jumper above the emulation cable. If this is done then power must not be applied to the Vcc pin (pin 30) of the emulation cable target connecter.

Example

```
?INIT                                    External clock
CLOCK SOURCE    = INTERNAL  External<SP>  is enabled.
PROGRAM MEMORY = INTERNAL  <CR>          (Note: only the
                                         E  in  EXTERNAL
?                                        is needed;  all
                                         other characters
                                         are ignored.)
```

Operands None

Syntax INITO

Purpose To enable the automatically memory initialization associated with the monitor commands LPM (Load Program Memory) and ASM (Execute Assembler). Memory initialization involves filling memory with >7F80 (NOP Opcode).

Notes The user cannot tell whether memory initialization is enabled or disabled when this command is executed. Executing this command when initialization is already enabled will continue to enable initialization. memory initializaion enabled at system reset.

Example

```
?INITO<CR>

?
```

Operands None

Syntax INTIF

Purpose To disable the automatically memory initialization associated with the monitor commands LPM (Load Program Memory) and ASM (Execute Assembler). Memory initialization involves filling memory with >7F80 (NOP Opcode).

Notes The user cannot tell whether memory initialization is enabled or disabled when this command is executed. Executing this command when initialization is already disabled will continue to disable initialization.

Example

```
?INITF‹CR›

?
```

Operands None

Syntax INTM

Purpose To display/modify the interrupt mask bit.

Note The interrupt mask is a single bit in the status register and is changed by the least significant bit of the entry.

Example

```
?INTM
    INTM = 0 <>
```

Operands Output port 1 or 2 (optional) default = 1

Syntax K[HELP] [output port]

Purpose To display the EVM keyboard entry aids (special function keys) menu for reference by the user.

Notes

1) This command can be executed by either typing K, or entering the full word.

2) It is functionally equivalent to the text editor's K command.

3) Legal values for output port are 1 and 2.

Example

```
?KHELP 1                Menu of special function keys
   .                    is displayed at the terminal.
   .
( Menu )
   .
   .
?
```

Operands Input port 1 or 2 (optional) default = 3

Syntax LMS [input port]

Purpose To load machine state data in the format created by the SMS command (see Section 3.3.1.14.).

Notes

1) The machine state can be created by the SMS command, or can be created as a text file on a host system with the same format as in SMS and downloaded through Port 2.

2) For users with terminal emulation software downloading through Port 1, the output port is set to 2.

3) If the machine state data is loaded from tape, a filename input is prompted for, and the tape is searched for a "state" file with that name.

4) The operation can be aborted while in progress with the <ESC> key.

5) The data memory portion of the machine state can only be loaded if stored in hex (HX) format (i.e., the audio tape default condition), or in signed decimal (SD) format. State files uploaded through Ports 1 or 2 to a host system must also use one of those two formats; never the signed decimal(SD) format.

6) During downloading from Ports 2 or 3, the input buffer count is displayed at the terminal. During downloading from Port 1, the input buffer count is displayed to Port 2.

Example

```
?LMS<CR>                    The machine state saved in
                            TSTAT has been restored.
FILENAME: TSTAT
3

?
```

Operands Input port 1 or 2 (optional) default = 3

Syntax LPM [input port]

Purpose To load program memory with object code stored either in TMS9900, TMS7000, or Tektronix format.

Notes

1) The LPM command automatically distinguishes between TMS9900, TMS7000, and Tektronix format. All object data is loaded at the absolute (not relocatable level.

2) When loading from Port 3 (audio tape), only file types with the correct file name will be loaded (see Section 3.3.1.16. . The LPM command derives the load address from the object file itself.

3) The operation can be aborted while in progress with the <ESC> key.

4) For users with terminal emulation software, downloading through Port 1, the output port is set to 2. During downloading from Port 1, the input buffer count and the IDT (for TMS9900 and TMS7000 formats) are displayed to Port 2.

5) During downloading from Ports 2 or 3, the input buffer count is displayed to the terminal. During downloading from Port 2, the IDT is displayed to the terminal for TMS9900 and TMS7000 formats.

6) When the load command executes, program memory is initially filled with the NOP opcode (>7F80).

7) The revision 1.2 firmware used a non-standard word count for the Tektronix object code download format. The revision 2.0 firmware has been changed to use the standard byte count in each object code record. This will allow use of standard Tektronix object code without the need for loader software to adjust the byte count to a word count before the file is downloaded to the EVM.

Example

```
?LPM<CR>                    User has loaded a saved object
                            file from Port 3.
FILENAME; PROG1
3

?
```

Operands None

Syntax MCC

Purpose To modify the character or character sequence for controlling the cursor and cursor left functions.

Notes

1) The % and ! chain terminators (see Section 2. must send a cursor up character or character sequence to the user's terminal in order to provide a fixed display. If the cursor up character or character sequence recognized by the user's terminal is not equivalent to the cursor up character loaded by the EVM at RESET (see Table 2-5), then the MCC command must be executed to modify the character or sequence, which can be one, two or three characters in length.

2) The MCC command also inputs the character or sequence for cursor left, which is sent to the terminal during normal editing operations.

3) Since cursor right is handled by overprinting, and cursor down is accomplished with a line feed, these characters need only be in the library of recognized control characters received from but never sent to the terminal. Control sequences in the library can only be two characters in length.

4) The MCC command cannot be aborted by an <ESC>, since this character is legal in cursor control sequences.

5) After execution of the MCC command, execution of the CLIB command will cause a display of the new cursor up character.

6) Executing the $DEFC command reloads the power-up default values of the cursor up and cursor left control characters. See Section 2 for procedures on creating a new EPROM to change the power-up default values of the control characters.

Example

```
?MCC
>0B  : ENTER CURSOR UP<CR>
>08  : ENTER CURSOR LEFT<CR>

?
```

When MCC is entered, the current value(s) for each cursor control is displayed in hexadecimal. The first entry prompt is the cursor up control. Entry of a <CR> as the first character will leave the current value intact and continue execution of the command. Nothing is echoed, and entries other than a <CR> (up to three characters) will change the value of the cursor up control. If a <CR> is not received before the fourth character, the command is aborted and the power-up default value is reloaded.

The second prompt in the display is for the cursor left control. Entry here is the same as for the cursor up control.

Operands Start address >0 to >8F (optional) default = 0

Syntax MDM [start address]

Purpose To display and/or modify locations in data memory.

Notes

1) Legal address range is >0 to >8F. Decimal addresses are allowed if preceded by a plus sign.

2) Since operation is in data memory, data display/modify format is assumed to be in signed decimal, but can be changed by using one of the subcommands (see Section 3.3.3.1). Hexadecimal data entry is allowed if preceded by a greater-than sign (>).

Example

```
?MDM +100
  64 = +00000 -1,R          Start at location >64 (+100),
  64 = -00001 <SP>          step to location >65, and
  65 = +00021 ,H   >0015    displayed in hexadecimal.
  65 = +00021 <CR>

?
```

Operands None

Syntax MENU

Purpose To display the monitor menu to the terminal.

Notes

1) This command is functionally equivalent to the PROM Utility MENU command.

2) The monitor menu can be sent to Port 2 (the line printer) with the /MON command (see Section 3.4.).

3) The monitor menu is displayed automatically on return from the text editor and any of the assemblers, if the terminal baud rate is greater than 1200 baud. At any baud rate, it can be requested by the MENU command.

Example

```
?MENU                                    (Command)

** TMS320 EVM MONITOR **       -- ---+

"HELP"   MONITOR COMMANDS
"MENU"   THIS MENU
"EDIT"   TEXT EDITOR                 >   (Display)
"ASM"    ASSEMBLER
"LBLA"   LINE-BY-LINE ASSEMBLER
"PASM"   PATCH ASSEMBLER
"RASM"   REVERSE ASSEMBLER
"PROM"   PROM UTILITY          -- ---+

?
```

Operands None

Syntax MO

Purpose To permit positioning of the cassette tape without having to disconnect the control cable at the tape drive.

Notes

1) With the cassette motor under program control, the user can use this command to force the motor to turn on for rewinding, positioning of the tape past the leader, etc.

2) Monitor execution stops when this command is activated; the MO command is terminated with either a <CR> or <SP>, at which time monitor execution recommences.

3) This command is discussed in detail in Section 8.

Example

```
?MO                    The tape recorder motor is enabled.
<CR>                   Control is returned to the monitor
                       with the <CR>.
?
```

Operands

1) Memory block start address (>0 to >FFF)
2) Memory block end address (>0 to >FFF)
3) Destination block address (>0 to >FFF)

Syntax

MOVE <start addr> <end addr> <dest addr>

Purpose

To move section of program memory to higher or lower parts of program memory.

Notes

1) Legal address range for all three addresses is >0 to >FFF.

2) The memory block start address parameter must be less than the block end address parameter or an ADDRESS ERROR message will display.

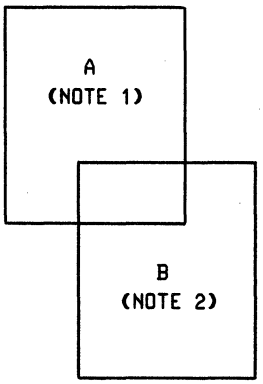
3) Memory blocks can overlap (see the example below), since moves down in memory start at the low address, and moves up in memory start at the high address (see Figure 3-1.

Example

```
?MOVE 100 200 170

?
```

Block 100-200 (A) is moved to begin at address 170 (B), as illustrated in Figure 3-1.



NOTE 1: If the move is performed from block A to block B, transfer starts from location >200 to >270 and continues backwards toward locations >100 and >170.

NOTE 2: If the move is performed from block B to block A, transfer starts from location >170 to >100 and continues forward toward locations >270 and >200.

Operands Start address >0 to >FFF (optional) default = 0

Syntax MPM [start address]

Purpose To display and/or modify locations in program memory.

Notes

1) The starting address allows the user to enter an offset address from >0 to >FFF. Entry is expected in hexadecimal, with decimal entry allowed if preceded by a plus sign (+) for addresses or a minus sign (-) for data.

2) Since the operation is in program memory, the data display/entry format is assumed to be hex, but can be changed with a subcommand. Decimal data entry is allowed if preceded by a plus sign or a minus sign.

Example

```
?MPM 800                  The opcode at >800 is  changed and
  800 = 6880 6881,R       displayed at <SP>, then causes the
  800 = 6881 <SP>         opcode at >801 to be displayed.
  801 = 7F80 <CR>

?
```

Operands None

Syntax NOP

Purpose To fill program memory with >7F80.

Note Since it fills memory, this command initializes the text editor.

Example

```
?NOP                Program memory is filled with >7F80.

?
```

Operands None

Syntax OV

Purpose To display and/or modify the overflow flag mode.

Note The overflow flag is a single bit of the status register and is changed by the least significant bit of the entry.

Example

```
?OV                    The overflow flag is displayed.
  OV = 0 <CR>

?
```

Operands: None

Syntax: OVM

Purpose: To display and/or modify the overflow flag mode.

Note: The overflow mode is defined by a single bit of the status register and is changed by the least significant bit of the entry.

Example

```
?OVM                    The overflow mode bit is displayed,
  OVM = 0 1,R           modified, and redisplayed.
  OVM = 1 <CR>

?
```

Operands None

Syntax PC

Purpose To display and/or modify the contents of the program counter.

Note The program counter is displayed and modified as a hexadecimal number. The legal range is from >0 to >FFF.

Example

```
?PC                 The contents of the program counter
  PC = 088 0,R      are displayed, modified, and redis-
  PC = 000 <CR>     played.

?
```

Operands None

Syntax P[REG]

Purpose To display and/or modify the contents of the auxiliary register.

Notes

1) The P register is displayed as a signed 32-bit decimal number, with input expected in signed decimal.

2) Entering a preceding plus sign (+) is optional, since it is assumed. Negative numbers must be preceded by a minus sign (-). Entering values in hex is allowed if a greater-than sign (>) precedes the entry.

3) Prime numbers greater than 2 to the sixteenth power are not allowed. Entering them will not cause an error, nor will it change the emulator's P register, but it will not load the TMS32010's P register correctly upon running or single-stepping.

Example

```
?PREG                    The contents of P register are
  PREG = -94066 1,R      displayed,    modified,    and
  PREG = 1 <CR>          redisplayed.

?
```

Operands None

Syntax RUN

Purpose To execute the user program with no breakpoints without having to clear all existing breakpoints.

Note The only way to recover from a RUN condition is with RESET. This is referred to as a "warm" RESET, and does not initialize baud rates, clock, memory sources, etc. When a warm RESET is performed, all TMS32010 registers except for the PC are saved. The state of the BIO pin is displayed and, if an interrupt occurred during execution, the INTERRUPT message is displayed.

Previoiusly, the RESET switch on the EVM was the only way to terminate user program execution when a breakpoint was not set. The escape key (<ESC>) will also terminate program execution in the same way as the RESET switch, allowing full control of the EVM from the keyboard. Since this is an uncontrolled halt of the TMS320, the program counter value is not retrieved but all other contents of the machine state are saved.

Example

```
?RUN                Execute without breakpoints. Execution
                    continues until  a RESET is  performed.
[ RESET ]           An interrupt occurred during execution.

USER HALT -PC RESET   BIO=0   INTERRUPT

?
```

Operands Initial breakpoint (optional) default = 1

Syntax SB [number 1 to 8]

Purpose To permit user to selectively display and/or modify up to eight breakpoints.

Notes

1) Legal range of breakpoint numbers is from 1 (default) to 8; entering a zero is equivalent to entering a 1.

2) A breakpoint that is not set (i.e., one that is clear) is displayed as XXX.

3) Legal address range for a breakpoint is >0 to >FFF. Entering a value larger than >FFF causes a non-fatal ADDRESS ERROR message to display.

4) Breakpoints can be entered in any order, and breakpoint locations can be skipped. For example, only SB5 and SB7 might be used.

5) When a program is executed to a breakpoint, the instruction at the breakpoint is not executed. Subsequent execution will start at the location of the breakpoint.

6) When displaying a breakpoint, the terminal will beep (if so equipped) if an event counter is set for that breakpoint.

7) A special check is built into the SB command to check for breakpoints set on illegal locations. If this occurs, a non-fatal BREAKPOINT ERROR message will display as follows:

```
BREAKPOINT ERROR  XXXX  YYY
```

where XXXX is one of the opcodes listed below, and YYY is the address of the opcode. An illegal location is defined as the location after the location containing one of the following mnemonics (opcodes):

```
TBRD    >67XX    PUSH   >7F9C      (Where X equals
TBWR    >7DXX    POP    >7F9D       any value)
CALA    >7F8C    IN     >4XXX
RET     >7F8D    OUT    >4XXX
```

8) If a breakpoint is set to occur at a location in program memory that is read by a TBLR instruction, the program will be halted when either the instruction at that locations is executed or when that location is read by the TNLR instruction. If the halt occurs because of the TBLR instruction, the program counter and the accumulator will contain the address of the breakpoint. However, the last instruction actually executed was the TBLR instruction to recover from this inadvertant breakpoint, the user must make the program counter point to the instruction following the TBLR instruction, and then continue execution with the EX command.

Example

```
?SB                          Breakpoint is set to 1.
  BP1 = XXX 88,R
  BP1 =088 <CR>

?
```

SET BREAKPOINT SUBCOMMANDS: The subcommands described here can only be used with the SB command. The subcommands may be entered in place of, or in addition to, data. The menu of subcommands is accessed by entering ",M", resulting in the following display:

```
,M      THIS MENU
,C      CLEAR BREAKPOINT
,R      REDISPLAY BREAKPOINT
,E      DISPLAY EVENT COUNT
<SP>,<LF>,CNTL/J,CNTL/V,CURSOR DOWN      Next Entry
CNTL<K,Z>,CURSOR UP                  Previous Entry
<CR>    QUIT
```

If entered in addition to data, the data is first stored and then the subcommand is executed. Occurrence of any error during entry of a subcommand will cause control to return to the monitor top level. The Next Entry/Previous Entry/QUIT commands allow sequencing of locations in any direction and normal command termination (return to the monitor). The other commands cause the following functions:

,M Causes display of the breakpoint subcommand menu.

,C Clears the current breakpoint.

,R Allows redisplay of the just entered breakpoint without having to advance to the next entry and then back up one.

,E Displays the event count currently set for the breakpoint most recently displayed. If no event count is set, this command functions the same as the ,R command.

Operands None

Syntax SCALE

Purpose To display a 16-bit decimal number scaled by the specified power of 2 from 0 to 15.

Notes

1) The SCALE command prompts the user for both the scale factor and the decimal input.

2) No more than five digits of decimal input may be entered and leading zeros are assumed.

3) Entry of a scale factor of 0 or of one larger than 15 will terminate the command.

4) If a number outside the range +32767 to -32768 is entered, a VALUE ERROR message will display.

5) The decimal input sequence repeats until entry of either <SP> or <CR> is made in place of a decimal number. Entry of <SP> returns to scale factor input. Entry of <CR> terminates the command.

Example

```
?SCALE
SCALE FACTOR? 2<CR>
DEC INPUT? 16<CR> = +4.000000000
DEC INPUT? <SP>
SCALE FACTOR? 1<CR>
DEC INPUT? 4<CR> = +2.00000000
DEC INPUT? <CR>

?
```

Operands None

Syntax SD16

Purpose To convert a 16-bit hex input to its signed decimal equivalent.

Notes

1) The SD16 command prompts the user for hex input and prints a preceding greater-than sign (>).

2) The user can enter as many digits as desired; however, the command will accept only the last four digits entered and assume leading zeros.

3) Data entered may be terminated with either a <CR> or <SP>. The converted number is then displayed on the same line, and the user is prompted for another input. If no entry is made, <CR>, or <SP> terminates the command.

Example

```
?SD16
HEX INPUT? >FFFF<CR> = -1
HEX INPUT?><CR>
?
```

Operands None

Syntax SD32

Purpose To convert a 32-bit hex input to its signed decimal equivalent.

Notes

1) The SD32 command prompts the user for hex input and prints a preceding greater-than sign (>).

2) The user can enter as many digits as desired, however, the command will only accept the last eight digits entered and assume leading zeros.

3) Data entered may be terminated with either a <CR> or <SP>. The converted number is then displayed on the same line, and the user is prompted for another input. If no entry is made, <CR> or <SP> terminates the command.

Example

```
?SD32
HEX INPUT? >F00FFF<SP> = 15732735
HEX INPUT? <SP>

?
```

Operands Output port 1, 2, or 3 (optional) default = 3

Syntax SMS [output port]

Purpose To dump the machine state data to the specified output port to be displayed or stored for later retrieval.

Notes

1) The machine state consists of the STATE command display and the contents of the data memory.

2) If the output port is specified as 1 or 2, the display format for the data memory is prompted for format. The choices are: hexadecimal (HX), signed decimal (SD), or unsigned decimal (UD). If the output port is specified as 3 (audio tape) the data memory is stored in hex format and a state descriptor is assigned to the file so that only the LMS command can load that type file.

3) The operation can be aborted while in progress by using the <ESC> key.

4) If the state data is assigned to Port 1 or Port 2, the <SP> bar will start and stop the dump. If the state data is assigned to either Port 2 or 3, the less-than sign (<) which is the end-of-dump character sign is appended. This character is required for indicating the end of the file for proper LMS command execution.

Example

1.
```
?SMS<CR>

FILENAME: TSTAT<CR>
READY TO RECORD? <CR>

(Data Dumped To Port 3)

?
```
The current machine state has been saved to audio tape. READY TO RECORD prompts to put cassette in record mode.

2.
```
?SMS 1
[ TMS32010 REGISTERS ]

[ TMS32010 DATA MEMORY ]

?
```
Machine state has been displayed to the terminal.

Operands

1) Memory dump start address >0 to >FFF
2) Memory dump end address >0 to >FFF
3) Output port number >0 to >FFF (optional) default = 3

Syntax

SPM <address 1> <address 2> [output port]

Purpose

To dump memory to Ports 1, 2 or 3. Data will be dumped in either 9900 format or Tektronix format.

Notes

1) The object data is dumped at the absolute (not relocatable) level.

2) Entering a dump end address that is less than the start address will cause an ADDRESS ERROR message to display. Failing to enter one of the address parameters will cause a VALUE ERROR message to display.

3) The range of address values is >0 to >FFF. When entering address parameters, only the last four characters entered are accepted and leading zeros are assumed.

4) The FORMAT prompt allows the choice of TMS9900 dump format (the default) or Tektronix format. Tektronix format is chosen by entering a"T". (The entire word may be entered, but only the T affects the decision.) Any other entry causes a default to TMS9900 dump format.

5) The operation can be aborted while in progress with the <ESC> key.

6) When stored to audio tape, object files in either format contain an "object descriptor" so that only the LPM command can load them.

Example

```
?SPM 0 FFF<CR>
FORMAT? (9900)<CR>

FILENAME: PROG1
READY TO RECORD? <CR>

(Data dumped to Port 3.)

?
```

The entire 4K-word program memory space has been saved in 9900 format to audio tape. READY TO RECORD prompts the user to put the cassette in Record Mode.

Operands

1) Display type 0 to 4 (optional) default = 0
2) Initial display port 1 or 2 (optional) default = 1

Syntax

SS [display type] [initial display port]

Purpose

To single-step the user program.

Notes

1) Legal values for display type are 0 (default) to 4. Legal values for output port are 1 (default) and 2.

2) All parameters are changeable (see subcommands, Section 3.3.5.7).

3) Attempting to enter a display type of 2 to 4 is only allowed if a trace line has been defined by the ST command. If displaying the trace line, the display format is asked for before execution begins. Since the trace line displays data memory locations, default is signed decimal.

4) The only line common to all display types is the first line. This line displays the PC (in hexadecimal) of the instruction to be executed next, the state of the BIO pin, whether or not an interrupt occurred (indicated by the INTERRUPT message), and the mnemonic of the instruction just executed.

5) Once inside the SS command, the prompt ("SS") is displayed when the command is ready to accept a subcommand. Entering a <CR> will return control to the monitor.

Examples

```
?SS              A single-step instruction is executed.
PC = 020   BIO=1   MNEMONIC---> NOP
ACC = +5  TREG = -1  PREG = -99
(SS)<CR>

?
```

SINGLE-STEP USER PROGRAM SUBCOMMANDS: The subcommands may be entered after the SS command prompt is displayed. The menu of subcommands is accessed by entering "M", resulting in the following display:

```
M      THIS MENU
Tx     CHANGE DISPLAY TYPE
F      CHANGE DISPLAY FORMAT
C      ENTER STEP PROGRAM
P      DISPLAY PROGRAM COUNTER
R      PC DISPLAY RANGE
1      LIST TO PORT1
2      LIST TO PORT2
<SP>   EXECUTE ONE SINGLE STEP
<CR>   RETURN TO MONITOR
```

Execution of any of the commands will cause the following:

M Causes the subcommand menu to display.

Tx (where x is the display type selected) Causes a change in the display type, as specified by x. The display types are:

0 (default) ACC,T,P
1 All Internal Registers
2 Type 0 Plus Trace Line set using ST
3 Type 1 Plus Trace Line set using ST
4 Trace Line set using ST
5 PC/BIO/Interrupt/mnemonic Only

If no locations are set in the trace line, attempting to enter a display type of 2 to 4 will force a TRACE NOT SET error message and default the display type to 0. This command automatically calls the F command for types 2 to 4 displays. The PC and the mnemonic of the instruction just executed are always displayed.

F Selects format for the display of the trace line (data memory locations). The user will be prompted for a format if the display type selected was from 2 to 4 and trace locations have been entered. Formats are: HX (hexadecimal), SD (signed decimal), and UD (unsigned decimal). Since trace locations are in data memory, default value for format is SD (signed decimal). This command automatically calls the C command. If a signed decimal format is specified, the user can also specify a scale factor.

C Allows the user to specify a count of steps to be taken the next time a <SP> is entered. Default is 0; entering a zero is equivalent to entering a 1. The maximum value allowed is 254. Entry is in decimal format. Any count entered is used as the default until the Single-Step command is quit.

This mode will continue until entry of a <CR>. An example of a step count entry is:

```
(SS) C
COUNT (0) 20
[single steping begins]
```

P Allows display of the current program counter value.

R Allows display/modification of the PC range within the limits:

PC LO = >000
PC HI = >FFF

The PC LO range is displayed first and can be changed or skipped. If the PC low range entry is terminated with a Next Entry character (<SP>, <LF>, Cursor Down), the PC high range is displayed and can then be changed or skipped. If the PC high entry is terminated with a Previous Entry character (<CNTL/K,Z or Cursor Up), the PC low entry is repeated. The display/modify process is terminated by entry of a <CR>. Address values between 0 and >FFF can be entered, with entry expected in hexadecimal. The values entered define a bracket for program counter values outside of which the SS command will inhibit display. This allows the user to single-step through a particular part of a program and then execute to that part of the program again when outside the range due to the display being inhibited. Should

enter a range of values and no display results during subsequent single steps, the P command will allow display of the current PC without leaving the single-step command mode. Execution for restricted PC range is not in real time.

1 Causes subsequent single-step displays to be sent to Port 1 (the terminal). This is the default condition upon entry into the SS command. The 1 command is intended to reset the condition caused by the 2 command.

2 Causes subsequent single-step displays to be sent to Port 2 for either printing on a line printer or logging to a host computer. For each single-step display sent to Port 2 that was caused by the user pressing the space bar, the SS prompt is repeated at the terminal. If a large step count is entered and all display is sent to Port 2 while display is active, keyboard input is inhibited except for <CR> and <SP> as described below.

<SP> Entering a <SP> from the SS prompt causes execution of one instruction. If a step count (subcommand C) different than 1 has been previously entered, that many steps will be taken. While multiple steps are being taken, entering a <SP> will freeze the display and the single-step process between instructions; entering another <SP> allows the process to proceed. Entry of a <CR> at any time will clear the count, stop the display, and return to the SS prompt. The user should note that while the single-step command is executing outside the PC display range established by a P command, entering a <SP> will not cause the command to freeze.

<CR> Entry of a <CR> from the SS prompt will return control to the monitor. Entry of a <CR> during display of single instruction stepa will return control to the single-step command with a prompt of SS, and clear any step count values remaining.

Operands Initial trace number 1 to 6 (optional) default = 1

Syntax ST [initial trace number]

Purpose To allow the user to selectively display/modify up to six data memory trace locations.

Notes

1) Legal range of trace numbers is from 1 (default) to 6; entering a zero is the same as entering a 1.

2) A trace that is not set (clear) is displayed as: XX.

3) Legal address range for a trace is >0 to >8F. Entering a value larger than >8F causes a non-fatal ADDRESS ERROR message to display.

4) Address entry is allowed in decimal if preceded by a plus sign (+).

5) See also the subcommands that may be used with this command is Section 3.

Example

```
?ST                                    Set TRACE1.
  TRACE1 = XX +143,R<CR>
  TRACE1 = 8F <CR>

?
```

Operands Stack starting location 0 to 3 (optional) default = 0

Syntax STACK [starting location]

Purpose To display and/or modify locations in the stack.

Notes

1) Stack location values are from 0 to 3, with 0 being the "top of stack" and 3 being "bottom of stack".

2) Stack locations are displayed in hexadecimal, with entry defaulting to hex.

3) Entry can be in decimal if preceded by plus or minus signs.

Example

```
?STACK 1                    Two middle stack  locations
  STACK1 = 000 33<SP>       are displayed and modified.
  STACK2 = 000 44<CR>

?
```

Operands Output port 1 or 2 (optional) default = 1

Syntax TABLE [output port]

Purpose To cause a display or print of the label table created by the most recent execution of the assembler. If no valid label table is contained in memory, nothing will display/print.

Notes

1) Legal values for output port are 1 and 2.

2) Execution of the command displays/prints the label table in two parts: the unresolved references, followed by the entire label table.

3) The table printed/displayed is the equivalent to the one printed when the SYMT assembler directive is executed.

Example

```
?TABLE

NUM     001     LOC   004     NUMZ   002
LOOP    00A     SUB1  09C     SUBZ   AAF
LOOP1   139

?
```

Operands None

Syntax TABS

Purpose To allow the user to display and modify, if desired, one or all of the three terminal tab settings.

Notes

1) Upon execution of the TABS command, the first tab setting is displayed. A tab setting can be changed by entering the new tab setting after the current tab value. After a new tab value is entered, terminating the entry with a comma will cause the tab to be redisplayed.

2) Entering a <CR> will terminate the command and return control to the editor. Entering a Next Entry character (SP, LF, CNTL/J, CNTL/V, cursor down) will display the next tab, and entry of a Previous Entry character (CNTL/K,Z or cursor up) will display the previous tab.

3) Tab settings must be in increasing value and less than or equal to decimal 70. Violation of either constraint will cause the tabs to be reset to their initial values of 8, 14, and 30.

Example

1.

```
?TABS                       Tab settings are
TAB1 = 8 12<SP>             changed.
TAB2 = 14 22<SP>
TAB3 = 30 40<SP>

?
```

2.

```
?TABS                       User changed and re-
TAB1 = 12 10,<CR>           displayed TAB1. The
TAB1 = 10<SP>               value entered for
TAB2 = 22 99<CR>            TAB2 is too large;
VALUE ERROR - TABS RESET    tabs are reset to
                            original values of
?                           8, 14, 30.
```

Operands None

Syntax T[REG]

Purpose To display and/or modify the contents of the T register.

Notes

1) The T register is displayed as a signed 16-bit decimal number with input expected in signed decimal.

2) Entering a preceding plus (+) sign is optional, since it is assumed. Negative numbers must be preceded by a minus sign (-). Entering values in hex is allowed if a greater-than sign (>) precedes the entry.

Example

```
?TREG                    The contents of T register are
  TREG = -899 >FFFF,R    displayed, modified, and re-
  TREG = -1 <CR>         displayed. Notice user chose
                         to enter -1 as >FFFF.
?
```

Operands None

Syntax UD16

Purpose To convert a 16-bit hexadecimal input to its unsigned decimal equivalent.

Notes

1) The UD16 command prompts the user for hex input and prints a preceding greater-than sign (>).

2) User can enter as many digits as desired; however, the command will accept only the last four digits entered and assume leading zeros.

3) Data entered may be terminated with either a <CR> or <SP>. The converted number is then displayed on the same line, and the user is prompted for another input. If no entry is made, either <CR> or <SP> terminates the command.

Example

```
?UD16
HEX INPUT? >FFFF<CR> = 65535
HEX INPUT?><CR>

?
```

Operands None

Syntax UD32

Purpose To convert a 32-bit hex input to its unsigned decimal equivalent.

Notes

1) The UD32 command prompts the user for hex input and prints a preceding greater-than sign (>).

2) User can enter as many digits as desired; however, the command will accept only the last eight digits entered and assume leading zeros.

3) Data entered may be terminated with either a <CR> or <SP>. The converted number is then displayed on the same line, and the user is prompted for another input. If no entry is made, either <CR> or <SP> terminates the command.

Example

```
?UD32
HEX INPUT? >F00FFF = 15732735<SP>
HEX INPUT? ><SP>

?
```

3.4 DISPLAY MENU COMMANDS

The monitor menu display commands allow the user to call up any menu available within the EVM software for display either to the terminal screen or to a printer. Several of the menus are accessible through the monitor subcommands (usually in the form of a ,M or M), which invokes the menu for a particular command type (display/modify montor, display/modify register set, etc.) Any of the menus may be called by executing the appropriate command in the following format:

```
?[COMMAND] [output port]
```

The output port parameter is optional; the default is 1 (to the terminal). The legal values for the output port parameter are 1 and 2.

All display menu commands are distinguished by being preceeded by a slash (/), and by having a command name similar to the command type to which the menu belongs (i.e., /MON for display/modify monitor commands menu, /MM for display/modify memory commands menu, etc.). The display menu commands are listed in Table 3-9.

TABLE 3-9 - DISPLAY MENU COMMANDS

COMMAND	MNEMONIC
Display All the Menu Display Menus	/HELP
Display the Monitor Menu	/MON
Display the PROM Utility Menu	/PROM
Display the Text Editor Commands Menu	/EDIT
Display/Modify Memory Commands Menu	/MM
Display Register Commands Menu	/REGS
Display Set Breakpoint Subcommands Menu	/SB
Display Set Trace Subcommands Menu	/ST
Display Baud Rate Subcommands Menu	/BAUD
Display Single-Step Subcommands Menu	/SS

The commands function in the following manner:

/HELP Causes a display/printout of all of the menus of display commands listed in Table 3-9. The format is:

```
/ HELP (output port)
```

/MON Causes a display of the monitor commands menu. The format is:

```
/MON [output port]
```

This command is distinguished from the /HELP command in that the /HELP command does not allow an output port parameter and the /HELP command also has a different function when in the PROM utility.

/PROM Causes the display of the PROM utility menu from the monitor command entry level. This display is equivalent to the Menu command when in the PROM utility. The format is:

```
/PROM [output port]
```

/EDIT Causes the display of the text editor command menu from the monitor command level. This command is equivalent to the text editor Help command, as described in Section 5. The format is:

```
/EDIT [output port)
```

/MM Causes the display of the Modify Memory commands menu. May be called at any time while executing one of the following monitor commands:

```
MPM     (Display/Modify Program Memory)
MDM     (Display/Modify Data Memory)
FBPM    (Find Byte in Program Memory)
FWPM    (Find Word in Program Memory)
FBDM    (Find Byte in Data Memory)
FWDM    (Find Word in Data Memory)
```

/REGS Causes the display of the Display/Modify Register commands menu. May be called at any time while executing any of the following monitor commands:

```
ACC     (Display/Modify Accumulator)
TREG    (Display/Modify T Register)
PREG    (Display/Modify P Register)
PC      (Display/Modify Program Counter)
AR0     (Display/Modify Auxiliary Register 0)
AR1     (Display/Modify Auxiliary Register 1.
OV      (Display/Modify Overflow Flag)
OVM     (Display/Modify Overflow Mode)
INTM    (Display Modify Interrupt Mode)
ARP     (Display/Modify Auxiliary Register Pointer)
DP      (Display/Modify Data Page Pointer)
STACK   (Display/Modify Stack)
```

/SB Causes the display of the menu of subcommands which may be used within the Set Breakpoint (SB) command.

/ST Causes the display of the menu of subcommands which may be used within the Set Trace (ST) command.

/BAUD Causes the display of the baud rate subcommands menu from within either the BAUD1 or BAUD2 command.

/SS Causes the display of the Single-Step (SS) command subcommands menu from within the SS command. Use of a comma preceding an /SS subcommand is optional with the SS command only (see Table 3-6.).

3.5 MONITOR PROGRAM SYSTEM ACCESS COMMANDS

The monitor program and the PROM Utility contain a set of system access commands that give the flexibility of using some of the monitor and PROM Utility commands to manipulate the system software resident on the EVM in the EPROM.

The system access commands for the PROM utility are discussed in Section 6. The following paragraphs generally pertain to both the monitor program and the PROM utility program system commands.

All system access commands are distinguished by being preceded by a dollar sign ($) and by having a command name mnemonic the same as the regular commands previously described in this chapter and the PROM utility commands in Section 6.

3.5.1 Format

The format for any system access command is identical to the regular command, except that the RAM restrictions have been changed from the TMS32010 RAM range (>0 to >FFF) to the TMS9995 RAM range of >0 to >FFFF, in which the TMS32010 RAM resides at >A000 to >BFFF.

> **NOTE**
>
> A command executing in TMS32010 RAM is word oriented, while its system access equivalent is byte oriented.

3.5.2 Command Menu

A menu listing all of the system access commands may be called by entering either $HELP.

3.5.3 Monitor Program System Access Commands

The monitor program system access commands and their regular command equivalents, as well as four unique system commands are listed in Table 3-10. The system access commands with regular equivalent commands function in the same manner as their equivalent commands. See the description of the equivalent command for information; a description of each unique command follows the table.

TABLE 3-10 - SYSTEM ACCESS COMMANDS

COMMAND	MNEMONIC	EQUIVALENT
Display/Modify Program Memory	$MPM	MPM
Display Program Memory	$DPM	DPM
Fill Program Memory	$FPM	FPM
Find Byte in Program Memory	$FBPM	FBPM
Find Word in Program Memory	$FWPM	FWPM
Move Memory	$MOVE	MOVE
UNIQUE COMMANDS		
Display/Modify Record Length	$DRL	
Monitor Expansion Command	$USER	
Display Operating System Revision Level	$REV	
Perform a "Power Cycle" RESET	$BOOT	
Assign EVM Identifier	$ID	
Disable EVM Output	$OFF	
Enable EVM Output	$ON	
Echo User Input to Port 2	$EC	
Echo All Characters to Port 2	$$EC	
Disable All Echo	$NEC	

The unique commands function in the following manner:

$DRL — Permits setting of the record length during download operations of fixed-length record input files. The record length may be changed at any time during download. The legal range is from 50 to 142, with 142 being the default. Downloading a variable-length record with record length longer than those allowed by the $DRL command will cause a loading error.

$USER — Gives the capability to branch to any point in the TMS9995 memory space. The starting address of TMS32010 RAM memory space within the TMS9995 RAM memory space is >A000. The return address to the monitor through the monitor menu is >0080.

REV — Causes a display of the revision level of the software resident on the EVM. (The only time the revision level is displayed automatically is on initialization of the system, i.e, on cold RESET.)

$BOOT — Performs a "power cycle" RESET from the keyboard and prompts for a <CR> to execute autobaud. This RESET cycles the power and initializes the baud rates, breakpoints, event counter, trace, clock and program memory sources (both internal), transparency mode toggle character (CNTL/C), the assembler label table, and the text editor.

NOTE

Once the system is initialized, the assembler label table and the transparency mode toggle character are not reinitialized by any other reset; $BOOT must be executed to reset these two.

$ID — Used to distinguish between the master and slave EVMs in a dual-EVM operation. (see Section 2.)

$OFF/$ON — Allows the output to the terminal (response to commands, etc.) to be turned off and on. After execution of the $OFF command, entry from the keyboard is still accepted, but the user sees nothing. Entry of the $ON command enables the display. Execution of the EX and RUN commands will also enable output, allowing display of breakpoints. Any RESET will also enable output.

$EC — Causes transmission of user-entered data from the terminal to be sent to Port 2. If a slave EVM is cnnected at Port 2, it will execute the same commands as the master EVM. If this command is executed on the master with the $$EC command executed on the slave, a printer or CRT connected to Port 2 of the slave will monitor the slave execution of commands from the master.

$$EC Causes echoing of all terminal activity at Port 2. For example, This allows terminal activity to be sent to Port 2 for an overall display for demonstration purposes. It also allows commands being executed by one EVM to be sent to another EVM that has had its output suppressed by the $OFF command.

$NEC Cancels both the $EC and $$EC commands.

3.6 MONITOR PROGRAM ERROR MESSAGES

Table 3-11 lists the monitor error messages and explains the cause of each.

TABLE 3-11 - MONITOR PROGRAM ERROR MESSAGES

ERROR MESSAGE	CAUSE
COMMAND ERROR	The command interpreter was not able to interpret the command as entered.
PARAMETER ERROR	The parameter analyzer associated with the command entered received a character that was not valid for the type of data expected. Examples: a Z for hex input, or, a 4 for a port parameter.
VALUE ERROR	A parameter is out of range for the data expected, such as entering a 4 for a port parameter. It can also indicate that a parameter was expected and not entered.
ADDRESS ERROR	An entered address is out of the legal range for the operation. Legal ranges are: Data memory: >0 to >8F Program memory: >0 to>FFF
TAPE ERROR	An invalid checksum for a block of data loaded from the audio tape is detected. The error includes the approximate RAM address of the error and continues with the load process.
CHECKSUM ERROR	Indicates detection of an invalid checksum during download of either 9900 or TX format object code. Error includes the approximate RAM address of the error and continues with the load process.
TIMEOUT ERROR	DTR signal true was not received within 20 seconds after entry in the write routine to Port 2. When the EVM is writing to EIA Port 2, it waits for DTR to start character transmission. If this signal does not go true in approximately 20 seconds, the process times out. (The duration of the timeout is user-changeable.)

TABLE 3-11 - MONITOR PROGRAM ERROR MESSAGES (Continued)

ERROR MESSAGE	CAUSE
TAG ERROR	Indicates the presence of an invalid object tag during download of either 9900 or TEK formated object code. Error includes the approximate RAM address of the error and continues the load process upon receipt of the next <CR> in the data stream.
BREAKPOINT ERROR	A warning against placing a breakpoint in RAM at a location containing an opcode for aninstruction that the EVM cannot breakpoint on. The error will occur if an attempt is made to set a breakpoint on the location after the location containing one of the following opcodes: TBRD >67xx (where xx is any number) TBWR >7Dxx CALA >7F8C RET >7F8D PUSH >7F9C POP >7F9D IN >4xxx OUT >4xxx The format for the message is: BREAKPOINT ERROR XXXX YYY, where XXXX is one of the opcodes listed above, and YYY is the address of the opcode.
DEVICE ERROR	The user must note that the RAM will only hold 8K bytes of text (including line numbers) while the active text editor holds 15000 bytes. If the 8K bytes is ever exceeded either on read or write a "DEVICE ERROR" message will be issued.

4. THE ASSEMBLER AND REVERSE ASSEMBLER

4.1 INTRODUCTION

This section provides a general discussion of the EVM assembler and reverse assembler.

4.2 ASSEMBLER EXECUTION

The assembler is executed from the monitor. The general format for executing the assembler is as follows:

```
?ASM [input port] [output port]
LINE NUMBERS? (NO)<CR>
```

Whenever the assembler is executed, RAM is filled with >7F80 (NOP) to insure valid execution outside the range of the user's program, and the text editor is initialized, clearing its contents. For information on assembler commands, refer to the TMS32010 Assembly Language Programmer's Guide. Also see Section 4.2 for exceptions to the instruction sets and assembler directives.

The assembler requires the use of an editor to produce text, which is then assembled using the EVM. The editor used can be the one resident on the EVM, or an editor on a host computer system. If the text is sourced from a host system, it must be bracketed with the <> symbols in the file.

For example:

```
*>                          (Insert *> sign at top of file)
LOOP BZ  >100               .....      ----+
     LARP 0                 .....          ¦
     SUB  13,4              .....          ¦
     SACL 10,0              .....          ¦> (Text)
*    COMMENT LINE           .....          ¦
     NOP                    .....          ¦
     END                    .....      ----+
*<                           (Insert *< sign at bottom of file)
```

The greater-than (>) and less-than (<) signs are the beginning-of- and end-of-file markers recognized by the EVM text editor and assembler. For the assembler these characters can be preceded by a comment symbol (*) so that any cross assembler can assemble the file.

The EVM assembler accepts both forward and backward referenced labels. All equate statements must occur before the equated label is used in an instruction. The assembled source listing will show >FFFF for branches for all forward referenced labels; however, the END statement directive will resolve those labels and place the correct value into the appropriate RAM locations. The object code in RAM may be inspected with the monitor commands DPM and MPM, as described in Section 3.3.9.

After the assembler receives the END directive, it lists all unresolved labels. Immediately following the unresolved listing is a list of all labels used in the program if the label table list is requested with the SYMT assembler directive. The output listing will list the label and the address of the label. These labels are printed as they occur in the assembler and label table, three on a line. Lastly, the total count of errors and warnings which occurred during assembly is also printed; this number is a decimal value.

Any time a file is assembled, only the object code is loaded into RAM. The original source and/or source listings are not stored in the EVM RAM.

All files assembled from a host system must enter the EVM through Port 2. The specifics of assembling from an external system are discussed in Section 4.1.2.

4.2.1 Input Port Designation

The input port can be either Port 1, 2, 3, or 4. If Port 2 is selected, the EVM will expect source file(s) to be input from Port 2 with the proper beginning- and end-of-file markers. Selecting Port 3 will cause the EVM to assemble from tape and will prompt the user for a file name. The assembler will only load source files. The prompt:

```
LINE NUMBERS? (NO)
```

is displayed when input is from Port 2 (host computer), and output is to Port 1. This mode is for download of text from a host system. This prompt is also displayed when input is from Port 1 and output is to Port 2 (a line printer). This mode is provided for users running terminal emulation software on a host system connected to the EVM at Port 1 (terminal).

The mode entered with both ports equal to two is a special mode for host system interaction, allowing text to be sent to the EVM and the listing returned to the host via one full duplex EIA link. The default for the prompt is NO, since most host text editors have no line numbers. If the file was generated by the EVM text editor and saved on the host system via uplink, then it would have line numbers, and the correct response in this case would be YES.

Entry of a Y or an N in response to the line numbers prompt is acceptable since only the first character is read; entry of any character other than Y is treated as an N.

Revision 1.2 requires a file to be uploaded from the text editor prior to assembly since there is not enough RAM on the EVM to support both programs at the same time. With Revision 2.0, it is possible to place an 8K byte static RAM in the EPROM programming socket (U28) and read/write to it as PORT 4 using any of the download/upload commands. Two types of RAM devices may be used (any access time will work):

ELECTRONIC DEVICES INC.	EDI8808
HITACHI	HM6264

The EDI8808 can be placed in U28 and used with no modification to the EVM. The HM6264 requires a solder jumper between pins 28-26 of U28 since it has an additional (active high) enable that is a no-connect on the EDI8808. This jumper does not affect EPROM programming since pin 26 is a no-connect on TMS2764 devices and, most importantly, inadvertantly turning on the programming voltate (+21 volts) with a static RAM device in U28 will cause no damage since pin 1 (VPP) is a no-connect on both of these chips.

Examples:

```
?EDIT 4<CR>          EDIT A FILE FROM PORT 4
*Q 4 <CR>            QUIT THE EDITOR TO PORT 4
?ASM 4 1<CR>         ASSEMBLE FROM PORT 4
```

4.2.2 Output Port Designation

The output port can be either Port 1, 2, or 0; it can never be Port 3. Entering the output port as a inhibits the listing. (see UNL Directive, Section 4.3.2.8). When the lising is inhibited in this way, the number of errors and warnings and the label table (see SYMT Directive, Section 4.3.2.9), if called for, are displayed if the input port is 2 or 3, and sent to Port 2 if the input port is 1.

If both port parameters are set to 1, the assembler functions as a line-by-line assembler (LBLA), with input and output at the terminal. If no port parameters are entered, the assembler defaults to the LBLA. (Section 4.2.5 details the LBLA.)

When the output port is 1 (the terminal) during download from Ports 2 or 3, the assembly listing can be stopped in progress and restarted any number of times by pressing the space bar <SP>. In this situation and when the output port is 1 or 2 during during download from Port 3, the <ESC> key will abort the assembly. Any time the input port is 1, reception of an <ESC> character will terminate the assembly.

Neither the assembler nor the LBLA support concatenated command strings (see Section 2).

4.2.3 Assembling Files from a Host System

Once a file is edited with the proper beginning- and end-of-file markers (see Section 4.2), the EVM is ready to assemble the file as it is entered into the system from the host.

4.2.3.1 Obtaining Hard Copy

If the user wants a hard copy of the assembled source listing of the program, a printing terminal attached to Port 1 is required (the TI 820 KSR terminal, for example). Another method of obtaining a hard copy of the assembled source listing is to load the file through Port 2 into the Editor, save the file to audio tape, and then assemble from the tape with a printer attached at Port 2. The sequence of commands for this last option is as follows:

```
?EDIT 2                          Downlinks a text file that has
LINE NUMERS? (NO)  Yes           a line number for each line.
```

or,

```
?EDIT 2                          Downlinks a text file with  no
LINE NUMBERS? (NO) <CR>          line numbers and generates the
                                 line numbers.
```

NOTE

Use the LIST command in the Editor to verify proper loading.

or,

```
?QUIT 3                          Saves the file to audio tape.
```

NOTE

Disconnect the external system from Port 2 and connect the printer to Port 2. Use the monitor BAUD2 command to set up the baud rate for the printer.

or,

```
?ASM 3 2                         This will  assemble  the  file
                                 from tape and dump the listing
                                 to Port 2.
```

NOTE

Refer to Section 5 for a complete description of the Editor commands.

The user can also bypass the step of saving the file to tape by downloading the file directly into the assembler through Port 2. The following examples demonstrate how this is accomplished:

Example 1: Assembly with line numbers

```
?ASM 2 1
LINE NUMBERS? (NO) Yes

** TMS320 EVM ASSEMBLER **

[ TRANSPARENCY MODE INITIATES DOWNLOAD ]  (see Section 2 for
                                          transparency mode)
*>
00001  000  7F82  INIT   EINT
00002  001  7F8A         ROVM
00003  002  F500         BV    CLROV
       003  FFFF
00004  004  6880  CLROV  LARP  0
00005  005  6E00         LDPK  0
00006   006              END
*<

LABEL TABLE ...
```

If there is a label in the label field, there can be only one space between the label and the last digit in the line number. There must also be at least one space between the mnemonic and the label, or at least two spaces between the mnemonic and the last digit in the number field.

Example 2: Assembly with no line numbers

```
?ASM 2 1
LINE NUMBERS? (NO) <CR>

** TMS320 EVM ASSEMBLER **

[ TRANSPARENCY MODE INITIATES DOWNLOAD ]

*>
00001  000  7F82  INIT   EINT
00002  001  7F8A         ROVM
00003  002  F500         BV    CLROV
       003  FFFF
00004  004  6880  CLROV  LARP  0
0)005  005  6E00          LDPK  0
00006  006               END
*<

LABEL TABLE ...
```

4.2.3.2 Suppressing the Assembly Listing

Another option available is the ability to suppress the assembled listing. There are two ways to do this:

1) Specify zero (0) for the output port parameter. This disables the entire listing but still loads the assembled object code.

2) Use the LIST and UNL assembler directives (see Sections 4.3.2.7 and 4.3.2.8)

> **NOTE**
>
> Since the source listing is not stored by the EVM, suppressing it during assembly will require another assembly to generate it.

4.2.4 Assembling Files from Audio Tape

The EVM assembler can accept source files from a tape if they were loaded to that tape by the EVM text editor. When the text file is loaded to tape the EVM will automatically include the beginning- and end-of-file markers described in Section 4.2. The format of the tape assembly command is as follows:

```
ASM 3 ‹output port›
```

The output port can be either Port 1 or Port 2. There is no line number option in the ASM command because, since they originally came from the text editor, all files assembled from tape already have line numbers. The text editor also automatically provides the space after the line numbers required by the assembler.

4.2.5 Concatenation of Audio Tape Files

A file concatenation feature exists for developing software with the EVM text editor and audio tape which allows text files of any size to be created, stored, manipulated, and assembled. When the text editor issues the RAM FULL error message, the user may save the file to tape, reinitialize the editor, then continue entering text. This process may be repeated as often as necessary, but each time, the last file created must have the END assembler directive. When the assembler is executed with input from the tape, it will assemble until it finds the END directive. If an end-of-file mark (<)is encountered first (as automatically provided by the text editor), it assumes file concatenation and responds with:

```
FILENAME:
```

at the terminal, accompanied by a beep. The user may then specify the file name of the next section of text. If the user has forgotten to put the END directive in the text, the <ESC> key should be pressed to force the assembler termination routine the same as if an END directive had been present. By using this method of file concatenation, long text files can be broken up into smaller more manageable parts, edited and stored in any order, then assembled in the proper order automatically using the built-in file search capability of the EVM.

4.2.6 The Line-by-Line Assembler (LBLA)

Revision 1.2 firmware enters the LBLA mode with the "ASM" command when both port parameters are 1:

```
?ASM 1 1        ENTER LBLA ON REV 1.2 FIRMWARE
```

Revision 2.0 firmware uses the command "LBLA" for the line-by-line assembler and reserves the format "ASM 1 1" for download assembly from a PC or intelligent terminal.

```
?ASM 1 1        DOWNLOAD ASSEMBLER FROM PC WITH LISTING TO PC

?LBLA           ENTER LINE-BY-LINE ASSEMBLER (LBLA)
```

The line-by-line assembler (LBLA) may be entered any time the user wants to assemble code a line at a time through Port 1 (the terminal). The LBLA is entered with the following command format:

```
?LBLA
```

Upon entering the LBLA, a banner message will appear, followed on the next line by a line number and memory location. The cursor will then be positioned for entry of a line of code. For example:

```
?LBLA
** TMS320 EVM LINE-BY-LINE ASSEMBLER **

00001  000          [] <--- Cursor
```

At this point, code may be entered. The LBLA will indicate errors as they occur. If an error is detected, an error message is printed on the next line, and the entire line containing the error is ignored. After printing the error number, a new line number and memory location are printed on the next line, but the memory location is unchanged from the line wih the error. For example:

```
00001  000              LOOP  CLB
*******ERROR 02
00001  000          [] <--- Cursor
```

CLB is an illegal mnemonic. Error 02 indicated the error (see Section 4.4). The label LOOP is not stored as a label because the entire line has been ignored.

As each line is entered and properly terminated, it is assembled and stored in RAM. All forward referenced labels will assemble as FFFF, but will be corrected in RAM as they are resolved. If the user references a label previously defined by its use in the label field, the LBLA will place the value in RAM and indicate it in the listing. For example:

```
00001  000  7F82  INIT   EINT
00002  001  7F8A         ROVM
00003  002  F500         BV    CLROV
       003  FFFF
00004  004  6880  CLROV  LARP  0
00005  005  6E00         LDPK  0
00006  006              END
```

4.2.7 The Patch Assembler (PASM)

Once the program is entered and completed with the END assembler directive, the user can change object code two ways: (1) using the MPM monitor command to directly change bytes in RAM (The MLP object code tables must be used in order to alter the program), or (2) by using a unique command (PASM) that will allow use of the LBLA to add code to the existing program, the referencing by the new code to any labels used in the initial program, and to bypass the automatic fill of program memory with NOPs. This subsection discusses the PASM command.

The command format is:

```
?PASM<CR>

** TMS320 EVM LINE-BY-LINE ASSEMBLER **

00001  000        [] <--- Cursor
```

The EVM will respond in the same manner as it does to an LBLA command. The only difference is that the EVM does not reset the label table. To use the PASM command successfully, two items are essential:

1) Patching can only be done on the most recent program assembled, and only if the text editor has not been entered. If a label table does not exist in system RAM, the results will be erratic, and a RESET will probably have to be performed.

2) The first line of code must be an AORG assembler directive. The AORG directive points to the location in the program where the patch is to be inserted. The AORG directive is discussed in Section 4.3.2.1.

Remember that the existing code is written over by the patch. If the user wants to insert a block of code in the program, a branch (BR) statement is patched in at the appropriate location in the existing program; it should branch to a location outside of the existing program limits. The patch is coded at that location and branched back into the existing program. Any instructions overwritten by the first branch must be included at the start of the patch. See example below.

```
                .
                .
                .

     (EXISTING PROGRAM)

                .
                .
00035  A34  7F89        ZACK
00036  A35  5008        SACL  SIGN
       A36  5006        SACL  ROOT
00037  A37  7E01        LACK  1
00038  A38  5007        SACL  ONE
00039  A39  2F07        LAC   ONE,15
                .
                .
00097  B03       END
                .
                .
```

If the user wants to insert a patch between the LACK 1 and the SACL ONE, it would be done as follows:

1) Enter the LBLA patch mode with the PASM command.

2) Use the AORG assembler directive to place the branch (B) statement at PC location >0A38.

3) Code in the branch statement to branch past the end of the existing program (in this case, >0B03 or greater).

4) Use the AORG assembler directive again to place the PC location at the point you chose to execute the branch.

5) Code in the patch, remembering to include the code eliminated by the branch statement and include another branch statement to return to the correct place in the original program.

CAUTION

Care must be taken not to start a patch on the second word of a two-word instruction.

6) Use the END assembler directive to terminate the patch. The following example demonstrates the patch process:

```
00001  000                AORG  >0A38
00002  A38  F900           B     >0B03
       A36  0B03
0003   A38                AORG  >0B03
00004  B03  5007           SACL  ONE
00005  B04                   .
                             .
                          (PATCH CODE)
                             .
                             .
00045  B94  F900         B     >0A39
       B95  0A39
00046  B97               END
```

The patching process can be repeated as necessary.

4.3 ASSEMBLER CONVENTIONS AND FORMATS

All TMS32010 assembler instructions are described in the TMS32010 Assembly Language Programmer's Guide, Part Number SPRU002B. All exceptions are discussed in this section.

4.3.1 Constants

Constants must be intered in signed decimal. If a number is entered in hexadecimal, it must be preceded by a greater-than sign (>).

4.3.2 Assembler Directives

The EVM assembler only supports the directives outlined in this section. The IDT and TITL directives are not supported, but do not cause an error message. Table 4-1 lists the directives.

TABLE 4-1 - ASSEMBLER DIRECTIVES

MNEMONIC	DEFINITION
AORG	Absolute Origin
BSS	Block Starting With Symbol
BES	Block Ending With Symbol
EQU	Define Assembly-Time Constant Directive
DATA	Initialize Word
WORD	Initialize 32 Bit Double Word
TEXT	Initialize text editor
LIST	List Source
UNL	No Source List
SYMT	List Label Table
PAGE	Eject Page
END	End Program

4.3.2.1 Absolute Origin Directive (AORG)

The purpose of the AORG directive is to change the contents of the location counter over the address range of the TMS32010 (>000 to >FFF, or 4K words). The assembler will place the object code in RAM in a position relative to the AORG value. Any time the

location counter exceeds the 4K maximum value, an error occurs. If the greater-than symbol (>)is not entered, the AORG statement defaults to decimal. The syntax is:

```
[<label>]   AORG   <adddress>   [comment]
```

4.3.2.2 Block Starting with Symbol Directive (BSS)

Executing BSS first assigns the label (if present) a decimal (default) or hexadecimal value, then increments the location counter by the value of the expression. The syntax is:

```
[<label>]   BSS   <expression>   [<comment>]
```

A directive entered with no label will advance the location counter by the value of the expression. The expression may be a hexadecimal or a decimal (default) value.

If a label is used, the label is assigned the value of the location of the first word in the block, and the location counter is advanced by the value. For example:

```
Label   BSS   >20                Save a buffer of 32 words.
```

4.3.2.3 Block Ending with Symbol Directive (BES)

BES first increments the location counter by the value of the expression, then assigns the label, if present. The syntax is:

```
[<label>]   BES   <expression>   [<comment>]
```

Not entering a label will adavance the location counter by the value of the expression. The expression may be a hexadecimal or a decimal value. If the command includes a label, the label is assigned the value of the location of the last word in the block, and the location counter is advanced by the value.

4.3.2.4 Define Assembly-Time Constant Directive (EQU)

EQU assigns a label a decimal (default) or hexadecimal value (an assembly-time constant). The syntax is:

```
[<label>]   EQU   <value>   [<comment>]
```

For example:

```
TEMP1   EQU   14
TEMP2   EQU   +14
TEMP3   EQU   >E
TEMP4   EQU   -14
```

A directive entered with no label will advance the location counter by the value of the expression. The expression may be a hexadecimal or a decimal (default) value.

With a label, the label is assigned the value of the location of the last word in the block, and the location counter is advanced by the value. For example:

```
Label   BES   >20                Saves a buffer of 32 words.
```

4.3.2.5 Initialize Word Directive (DATA)

The DATA directive specifies that the operand that follows is a constant to be loaded into the next word of memory. Only the last four valid hexadecimal digits entered are stored. Data may be entered as a positive or negative decimal (default) value. DATA statements can also be the sum of previously defined EQU statements, with an unlimited number of terms, and can also have as a last term the sum of a hexadecimal or decimal value. DATA in hexadecimal format will accept the last four characters entered, with leading zeroes assumed. Data entered in signed decimal must be in the range -32768 to +32767. A

data statement can also have a label as the operand, if the label is already defined in the program. For example:

```
900  DDAD        DATA >FDDAD
901  0020        DATA 32
902  FFFF        DATA -1
```

4.3.2.6 Initialize 32 Bit Double Word Directive (WORD)

A 32 bit word directive has been added to the EVM assembler that allows the user to directly specify a 32 bit value without using two 16 bit data statements. The power of the word directive is not evident when working with hexadecimal numbers, since any 32 bit value can directly be split into two 16 bit MS and LS words. But if the user wants to enter a table of 32 bit decimal values, it would be necessary to first convert each number to hexadecimal (with a monitor command like "HX32") and then enter them as pairs of data statements:

```
004  FFFE   LABEL   DATA  -2
005  FFFF   LABEL   WORD  -2
     FFFE
007  0000   LABEL   WORD  >A
     000A
009  0000   LABEL   WORD  10
     000A
```

Entries for the word directive are assumed to be positive decimal values unless preceded by a '-' (negative decimal value) or a '>' (hexadecimal value). All negative values are sign extended. The most significant word is stored in the lower address and the least significant word is stored in the higher address.

4.3.2.7 Initialize Text Directive (TEXT)

The TEXT directive specifies that a string of arbitrary length will follow as the operand. The string must be enclosed in single quotes and follow the format for string entries. Null strings are not allowed. For example:

```
900          TEXT   "<string>"<CR>
906
```

Note that the ASCII values of the TEXT string are not printed. This is a special space-saving measure for listings. The next address printed indicates the next available byte after the TEXT string. The ASCII values are placed in the least significant byte of successive words.

4.3.2.8 List Source Directive (LIST)

The LIST directive enables the listing at the output port, unless the output port is specified as zero when the assembler is entered. When the output port is not zero, the assembler is normally in the list mode, so this directive is only needed to reverse a UNL directive. The syntax is:

```
[<label>]   LIST   [<comment>]
```

4.3.2.9 No Source List Directive (UNL)

The UNL directive suppresses the listing at the output port until either a LIST directive is encountered or the END statement is processed. Once the END statement is processed, the listing is turned back on for listing of any errors, warnings, and the label table are called for. The syntax is:

```
[<label>]   UNL   [<comment>]
```

4.3.2.10 Enable Label Table Directive (SYMT)

The SYMT directive enables the listing of the entire label table after assembly is complete. Once the directive is encountered by the assembler, there is no way to suppress the label table listing. The SYMT directive can be placed anywhere in the text file. If the SYMT directive is not present in the text file, printing of unresolved labels still takes place after assembly is complete. The syntax is:

```
[<label>]   SYMT   [<comment>]
```

4.3.2.11 Eject Page Directive (PAGE)

Executing the PAGE directive causes a form feed sequence to be sent to the output port. The syntax is:

```
[<label>]   PAGE[<comment>]
```

The sequence is:

```
<FF><CR><LF>
```

4.3.2.12 Program End Directive (END)

When an END directive is processed, after unresolved labels and a sum of all warnings and errors are printed. system control is returned to the monitor. The END directive will ignore a label in the label field.

4.3.2.13 The Comment Line

A comment may be entered on a program line if an asterisk (*) is the first character of the comment. For example:

```
000   *  These
001   *  three lines
002   *  are comment lines
003
```

4.4 ASSEMBLER ERRORS

Whether using the assembler or the LBLA, the error listing format is the same, except that in the LBLA mode, the user must correct the mistake before continuing. The error listing will always occur on a new line. Any labels defined on the line containing the error are ignored, along with the assembly mnemonic. In the assembler Mode (not LBLA), the assembler will store two NOPs (>7F80) in RAM. For either mode, unresolved branch instructions will have >FFFF stored in the branch address word (after the B opcode). This allows the user to correct the errors (after the entire file is assembled) with the MPM command or the LBLA patch command (PASM).

```
Errors are indicated by:      ****** XXXXX ERROR YY

Warnings are indicated by:    *** XXXXX WARNING YY
```

where XXXXX is the line number (not displayed in LBLA mode), and YY is the error number. If the listing is suppressed either with the UNL directive or by setting the output port to zero, the error/warning messages will still be displayed as the errors occur. If the input port is 2 or 3, the messages will be sent to the terminal. If the input port is 1, they will be sent to Port 2. Table 4-2 lists the error codes and an explanation of each; Table 4-3 lists the warning codes and an explanation of each.

TABLE 4-2 - ASSEMBLER ERROR CODES

CODE	EXPLANATION
01	Characters after label and mnemonic are not spaces, or, character after mnemonic is not a space or <CR>.
02	Illegal mnemonic.
03	Expected comma at end of field to indicate start of next required field.
04	Comma or <CR> not valid for first character of a field.
05	Incorrect line number read.
06	Field terminator was not space, comma, or <CR>.
07	First character must be alpha.
08	Character immediately following label must be a space.
09	Label already exists.
10	The "x" input for SACL must be a zero.
11	This mnemonic requires an operand field.
12	Incomplete mnemonic.
13	Branch and call statements require label or absolute address.
15	location counter beyond TMS32010 range.
16	Found <CR> instead of start of mnemonic.
17	Field either initiated or terminated by an illegal character.
18	Label must be a previously defined value.
19	The "x" input for SACH must be a 0, 1, or 4.
20	Multiply constant is out of range. Must be −4096 to +4095.
21	The MAR mnemonic may only use indirect addressing.
22	TEXT string is not preceded by a single quote (').
23	EQU statement must have a predefined label.
24	Branch is to address outside TMS32010 range.
25	Number entered is too large.

TABLE 4-3 - ASSEMBLER WARNING CODES

WARNING CODE	EXPLANATION
01	Value truncated.
02	Character terminating last field not a space or <CR>.
04	AORG address is out of range. Must be 0 < = AORG < = 4095.
05	<CR> encountered before closing single quote (') in TEXT string.

4.5 THE REVERSE ASSEMBLER (RASM)

The TMS32010 EVM reverse assembler converts object code between specified addresses into TMS32010 assembly language mnemonics without labels. It is executed from the monitor. The format of the command is:

```
?RASM <start address> <stop address> [output port]
```

The start and stop address parameters must both be entered and cannot be greater than >FFF. The output port parameter is optional, with legal values being 1 (default-terminal) and 2.

CAUTION

When executing the reverse assembler, care must be taken not to start the process on the second word of a two-word instruction.

The reverse assembler will print a line of data in the following format:

```
XXXXX YYY   ZZZZ   MNEMONIC OPERAND(S)
```

where XXXXX is the line number (1-up count), YYY is the address of the opcode, ZZZZ is the opcode, and the mnemonic is the reverse assembled opcode with operands, if any.

If the data at the address YYY cannot be interpreted as a valid form of a mnemonic, it is printed as a DATA statement. If the data at a location can be interpreted as a valid mnemonic (whether it is or not), it is printed as a mnemonic. Any value greater than >FF00 will be printed as a DATA statement. If the opcode is a branch, the destination of the branch is printed on the next line in the form of an absolute address.

5. THE EVM TEXT EDITOR

5.1 INTRODUCTION

The EVM text editor is line-number oriented, with character edit capability. The editor may be used to build assembly language source files as well as general text files. Entering the text editor resets the assembler label table. This section details the operation of the text editor.

5.2 PROCEDURES AND FORMATS

The following subsections describe the procedures and formats for properly executing the text editor.

5.2.1 The EDIT Command

The text editor is executed from the monitor by entering the EDIT command. The format is:

```
?EDIT <input port>
```

The input port parameter can be either 1 (terminal), 2 (downlink), 3 (audio tape) or 4 (See Section 4.2.1 for a description of port 4). The port parameter determines the initial input of text into the editor. If the user intends to edit an existing file, Port 2 or 3 should be specified and the text loaded from that port.

If the file to be loaded was not sourced from the EVM, it will need to be bracketed by the EVM buffer control characters. The greater-than sign (>) must be the first character in the file, followed by <CR>. The file must end with a <CR> followed by a less-than sign (<). When the text editor dumps its contents, these characters are automatically generated.

After the load is complete, control returns to the terminal. If Port 1 is specified, text entry from the terminal is immediately enabled. If no port parameter is specified, Port 1 is assumed. If download to the text editor is through Port 1 via user terminal emulation software, the buffer control characters are not required, since normal terminal input is expected. In this case, the user must include line numbers or one autoincrement line number command in the text (see below). If the input port is specified as Port 2, the user is given an additional prompt as follows:

```
?EDIT 2                          Downloads text file without
LINE NUMBERS? (NO) <CR>          line numbers.

[ TRANSPARENCY MODE INITIATES DOWNLOAD "    (See Section 2
                                             for information
                                             on transparency
                                             mode)
```

Text editors on host systems are usually not line-number oriented as is the EVM text editor. Thus, when the EVM text editor is loaded with text from a host ystem, it must create the line numbers. This is the default condition for the prompt. If the text file being loaded was created with the EVM text editor, it already has line numbers, and te proper response to the prompt would be:

```
?EDIT 2                        Downloads text file with line
LINE NUMBERS? (NO) <Yes>       numbers.

[ TRANSPARENCY MODE INITIATES DOWNLOAD "
```

If the input port specified is 3, the user is prompted for a filename. After entry of the filename, the audio tape controller will begin to search the tape for a source file with that name. It is the responsibility of the user to rewind the tape and/or to insure that the file desired is ahead of the tape position. The text editor will only load source files. The command sequence is:

```
?EDIT 3

FILENAME: TEST
5
```

During downlink from either Port 2 or 3, the number of input buffers loaded is displayed to the terminal as an indication of activity.

5.2.2 The Text Editor Banner

When the text editor mode is entered, the following banner is displayed:

```
** TMS320 EVM TEXT EDITOR **

"H" HELP

XXXXX FREE BYTES
*
```

where XXXXX is the number of bytes (in decimal) available for text storage. (The number displayed above is a maximum.) After a successful download into the editor either from a host system via Port 2 or from tape via Port 3, some of the bytes will have been used and the number display will indicate how much space is left for text storage. Subsequently, as more text is entered, this number will decrease. The prompt is an asterisk (*), and is displayed whenever input is expected. As in the monitor, the Escape <ESC> key is used to abort the current activity and return control to the top level of the editor.

5.2.3 Text Editor Memory

Figure 5-1 illustrates the text editor memory map.

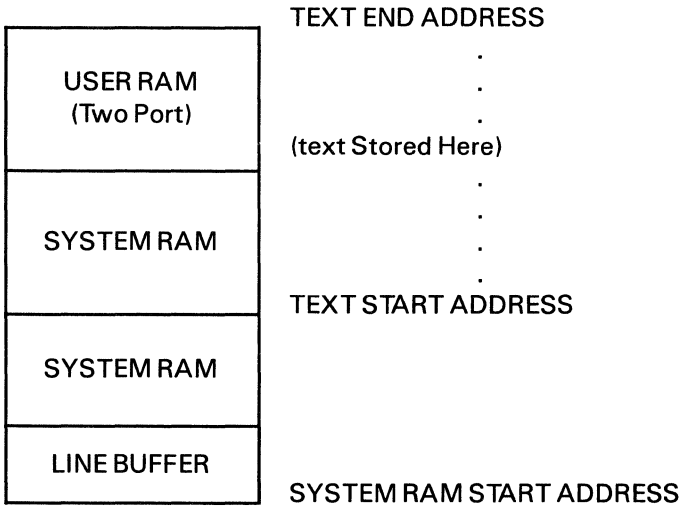


FIGURE 5-1 - EVM TEXT EDITOR MEMORY MAP

The line buffer holds data input from the keyboard, and lines that are currently being edited are terminated by a <CR>. The line is then linked in memory in the proper sequence. The current line is stored in the next open place in memory, and the two bytes before that line number are used to store the address of the next line. The corresponding two bytes in the previous line are changed to point to the new line.

5.3 TEXT EDITOR COMMANDS

All commands in the text editor are executed by the first letter of any command entered. Text editor commands are also summarized in Appendix B.

The text editor supports concatenated command strings (see Section 2), with the exception of autoincrement, edit, and text entry commands, thereby allowing the concatenation character (;) to be used in the text. Table 5-1 lists the text editor commands and definitions.

TABLE 5-1 - EVM TEXT EDITOR COMMANDS

DESCRIPTION	MNEMONIC
Autoincrement Line Number Mode	AUTO
Change Line Number	CHANGE
Save Line/Edit Next Line	CNTL/J, CNTL/V, line feed, or cursor down
Save Line/Edit Previous Line	CNTL/K, CNTL/Z, <ESC>CNTL/L, or cursor up
Duplicate Line	DUPLICATE
Edit Line	EDIT
Find String (8 characters maximum)	FIND
Display Editor Commands Menu	HELP
Display Keyboard Entry Aids Menu	KHELP
List Line(s)	LIST
Display Free RAM Available	MEMORY
Quit Editor and Dump File	QUIT
Resequence Line Numbers to EOF	RESEQUENCE
Display/Modify Tabs	TABS
Clear text and Initialize Editor	ZERO
Delete Line Number	<CR>
Abort Current Activity	<ESC>

A description of each command is presented beginning on page 5-5.

5.3.1 Entering Text Into RAM

Text may be intered into RAM after a line is started with the Enter Line command (LINE<SP>), either manually or when provided by entering the autoincrement mode (see autocommand description). The command format is:

```
XXXXX‹SP›TEXT‹CR›
```

Where: XXXXX means the line number, i.e., all lines begin with a five-digit number from 00001 to 65525; the line number 00000 is illegal. Attempting to enter a line number greater than 65535 will cause the issuance of a LINE NUMBER ERROR message. Any number entered with less than five digits is assumed to be padded with leading zeroes.

<SP>: Entering a space after the line number tells the text editor that text will follow, thereby distinguishing the command from a line number preceding a

command. This space is provided automatically in the Autoincrement mode. This space is not stored with the text.

<CR>: A carriage return signifies the end of the line of text and tells the text editor to store the line from the line buffer to RAM in order of increasing line numbers.

Notes:
1) The text editor fills RAM from the user RAM start address to the user RAM end address (see Figure 5-1. The editor will not allow the text to be stored in the system RAM, since this area is needed if the text is to be dumped to either cassette tape (Port 3. or to another computer (Port 2.). When text storage uses RAM to within 64 characters of the RAM end address, the RAM FULL message is issued after any operation that involves storing text to RAM. The user can use the MEMORY command to make best use of the remaining space. The editor will quit storing lines with they are too long to fit in the remaining RAM, but will still allow the user to type lines as usual, continually issuing the RAM FULL message.
2) Three limitations are placed on the nature of the text. First, the text editor will not allow control characters to be entered as text. If a control character that is not a command (see example below) is entered, an audible beep is issued by the terminal. Second, if the text of a line consists only of spaces, the line will not be stored. Third, the maximum number of characters allowed in a line is 72.
3) While in text entry mode, the user can use the cursor control key, the insert and delete functions, and the tab key to format the files neatly.

An example is shown below.

```
*00010          AORG 0
*00020          B         INIT BRANCH VECTOR FOR RESET
*00030          B         INTR BRANCH VECOTR FOR INTERRUPT
*00040 *
*00050 INIT     EINT      INITIALIZE INTERRUPT MODE
*00060          ROVM      INITIALIZE OVERFLOW MODE
*AUTO<CR>
00070          BV   CLROV CLEAR OVERFLOW FLAG
00080 CLROV    LARP 0     INITIALIZE ARP
00090          LDPK 0     INITIALIZE DATA PAGE POINTER
00100 * INITIALIZATION COMPLETE
00110 <CR>
*
```

In the example above, the user has executed the text editor from the monitor and specified input from Port 1, the terminal (EDIT 1. . After initializing (see ZERO command), the user began entering text. After entering line 60, the user entered autoincrement line numbers mode (see AUTO command).

Parameters: Line number preceding command (optional)

Syntax [line#]A[UTO]

Purpose To automatically print to the terminal sequentially incremented line numbers, followed by a space, for quick and orderly entry of text.

Notes

1) The autoincrement command initially uses the default line number increment (10) installed when a ZERO command is executed. If a RESEQUENCE (line numbers) command is executed, the AUTO command uses that increment. When a line number is entered preceding the AUTO command, Autoincrement mode is entered beginning with that line number. If no line number is entered, the editor positions the autoincrement pointer at the end of the text file and starts with the last used line number plus the current increment.

2) To exit autoincrement mode, enter a carriage return <CR> immediately after the line number. If a line of text with that line number already exists, it will not be deleted. Entering and immediately exiting autoincrement mode will cause the current line to point to the last line in the file.

Examples:

```
1. *AUTO          Text editor is executed, memory is
   00010          initialized with ZERO command, and
                  autoincrement mode is entered.

2. *AUTO          Text editor is executed with a download
   00860          load of text from either Port 2 or 3,
                  then autoincrement mode is entered.
                  Pointer is automatically positioned at
                  the end of the text.

3. *500A           Autoincrement mode is entered beginning
                  at line 500.

4. 00200 <CR>      Autoincrement mode is quit.
   *
```

Parameters: 1) Line number to be changed (optional)
2) New line number

Syntax [line#]C[HANGE] <new line number>

Purpose To allow the user to change the line number of a line in text memory.

Notes

1) This command will only execute properly if two conditions are satisifed: the line number to be changed must already exist in the text file, and the new line number specified must not already exist in the text file. If either condition is not satisfied, the LINE NUMER ERROR message is displayed.

2) If the extra memory the text editor needs to change the line number of a line will cause the text file to exceed the user RAM space, the RAM FULL message will display. Issuance of any error aborts the command.

Examples:

1
```
*50CHANGE 70
LINE NUMBER ERROR
*
```
Attempted to change a line number to an already existing line.

2.
```
55CHANGE
LINE NUMBER ERROR
*
```
Attempted to change the line number of a nonexistent line. Command immediately terminated.

3.
```
*60CHANGE 45
*40CHANGE 65
*
```
Lines 60 and 60 changed to lines lines 45 and 65 respectively. Lines numbers 60 and 40 are automatically deleted from the text file.

Parameters: None

Syntax CNTL/J, CNTL/V, Line Feed, or Cursor Down

Purpose To save the line currently being edited, then display thenext line for editing.

Notes

1) This command gives the user a quick way of scrolling through a file. If the last line is encountered, the last line is continually redisplayed.

2) When this command is used while editing a line from the FIND command, the next line is displayed until a <CR> is entered, returning control to the text editor.

Save Line/Edit Previous Line

CNTL/K
CNTL/Z
<ESC>CNTL/L
CURSOR
UP

Parameters: None

Syntax CNTL/K, CNTL/Z, <ESC>CNTL/L, or Cursor Up

Purpose To save the line currently being edited and to display the previous line for editing.

Notes

1) This command is the opposite of the Edit Next Line command on the previous page.

2) If the first line in the file is encountered, the first line is continually redisplayed until a <CR> is entered to terminate the command.

Parameters: 1) Line number to be duplicated (optional)
2) New line number

Syntax [line#]D[UPLICATE] <new line number>

Purpose To allow the user to duplicate a line of text in memory.

Note This command is identical in operational requirements to the CHANGE command, i.e, the first line number to be duplicated must already exist, and the new line number specified cannot exist.

Examples:

1.
```
*DUPLICATE
LINE NUMBER ERROR
*
```
First line must exist in order to be duplicated.

2.
```
*60D 80
LINE NUMBER ERROR
```
Second line number cannot already exist.

3.
```
*60DUPLICATE 150
*60<CR>
*150CHANGE 60
  *

  *
```
Line 60 is duplicated at line 150, original line 60 is deleted, then line 150 is changed to 60.

Parameters: Line number preceding command (optional)

Syntax [line#]E[DIT]

Purpose To allow the user to insert/delete/replace characters within a line.

Notes

1) If no line number is entered, the current line number is assumed. (The current line is the first line of the file immediately after intering the text editor.) If the user is in the process of entering text from the keyboard, the current line is the most recent line number being displayed. If a line number is specified, that line becomes the current line and is used when the EDIT command executes.

2) When the EDIT command executes, the line of text is dumped to the terminal with the cursor positioned at the start of the line.

3) In the edit mode, the text editor will accept characters from the keyboard and store them at the cursor position, moving the cursor one position to the right for each character entered. The character input routine does not accept control characters as text and will beep the terminal when an illegal character is entered. Edit mode commands available to the user are:

   CNTL/J, CNTL/V, Line Feed, Cursor Down
   (Save Line/Edit Next Line)

   CNTL/K, CNTL/Z, <ESC>, CNTL/L, Cursor Up
   (Save Line/Edit Previous Line)

Example:

```
*10E                        Line 10 is edited.  Cursor is
00010 LABEL   EQU >2000     positioned at first character
      ["  < - cursor        on the line.
```

Parameters: String

Syntax F[IND] <string> (terminals < = 1200 baud)
Or,
F[IND] <prompt> <string> (terminals > 1200 baud)

Purpose To allow the user to quickly locate a string of up to eight characters text file.

Notes

1) The FIND command matches the specified string with its occurrence in the text file and prints any matches to the terminal.

2) The format of the display is the line that the string occurs in, followed by a redisplay of the string as a reminder to the user.

3) FIND returns a find for all occurrences of a string.

4) The maximum length of the string to be searched for cannot exceed eight characters. The characters in the string can only be within the range of >20 to >7E of ASCII values (no control characters). Entering a character out of this range will cause an audible beep to the terminal.

5) If the baud rate of the terminal is greater than 1200 baud, a prompt will be printed for the string input. When the prompt is printed, the string may also be entered on the same line as the command.

6) When an occurrence of the string is found, the program halts with the cursor postioned at the end of the line and waits for input. One of three commands may be given at this point:

<CR>	Terminate search
<SP>	Continue search
E	Edit line (see EDIT command)

If the user chooses to edit the line, termination of the edit command returns to execution of the FIND command at the start of the line just edited.

7) When the FIND command is terminated by a <CR>, control returns to the text editor, and the asterisk (*) prompt is printed. If searching continues to the end of the file by continuation with the <SP> bar after each occurrence, the editor prompt will be displayed signifying that there are no more occurrences, and control returns to the editor.

8) Failure to find a string will cause the NO OCCURRENCES FOUND message to display, and control will be returned to the editor.

Examples:

1.
```
*FIND MASK
FOUND:  00020        AND  MASK     <SP>
FOUND:  00100        LAC  MASK     <CR>

00100         LAC  MASK,15<CR>

END OF TEXT
*
```

In the above example, the search was for MASK. The first occurrence was passed with a <SP>. The second occurrence was edited, with no changes made. Edit was terminated with <CR> and searching continued with edited line. <SP> continues the search, no further occurrences are found, and the search terminates automatically. The prompt will not be displayed since the baud rate is less-than or equal-to 1200 baud.

2. *FIND

```
LEGAL STRING LENGTH IS 1 TO 8 CHARACTERS
ENTER A STRING TO BE LOCATED:  BANZ<CR>
FOUND:  00010        AND MASK     <CR>
*
```

In the above example, the user prompt is displayed because the baud rate is greater-than 1200 baud. The first occurrence of MASK is found. FIND is terminated with a <CR> following the MASK.

3. *FIND SROOT

```
NO OCCURRENCES FOUND
*
```

In the above example, the string SROOT was not found in the text file. The user prompt is not displayed because the baud rate is less-than or equal-to 1200 baud.

Parameters: Output port (optional) default = 1

Syntax H[ELP] [output port]

Purpose To display the valid text editor commands.

Notes

1) This command can be invoked in the editor any time the prompt asterisk is displayed.

2) Upon execution of this command, the valid editor commands will be dumped to the appropriate port specified.

Examples:

```
1.  *HELP                    Displays the editor commands to
                             the terminal.
    [ Menu of commands "

     *

2.  *HELP 2                  Dumps the editor commands to
                             Port 2.
    [ Menu of commands "

    *
```

Parameters: Output port (optional) default = 1

Syntax K[HELP] [output port]

Purpose To display the various special function key commands available to the user to manipulate text.

Notes

1) This command can be invoked in the editor any time the monitor prompt asterisk is displayed.

2) Upon execution of this command, the cursor control and character manipulation commands will be dumped to the port specified.

Examples:

```
1.  *KHELP                  Displays the keyboard entry
                            aid menu to the terminal.
    [ Menu of commands "

    *

2.  *K                      Dumps the menu to Port 2.

    [ Menu of commands "

    *
```

Parameters: 1) Line number preceding command (optional)
2) Number of lines to list (default = 1)

Syntax [line#]L[IST] <# lines>

Purpose To list lines of text in order of ascending count to the terminal.

Notes

1) The number of lines that can be listed is in the range of 1 to 65535.

2) If no line number is entered, the first line listed is the current line. The current line is pointed to by the line number most recently displayed. Immediately after entering the text editor, the current line is the first line in the file. If a line number is entered, listing starts with that line. In either case, listing continues until either the prescribed number of lines are dumped, or until the end of the text file is encountered.

3) While the text is being dumped to the terminal, the user can press the <SP> bar to stop the listing in order to view the text. Another press of the <SP> bar reactivates the listing process, or, pressing the <ESC> key causes an abort of the command.

Examples:

1.
```
*LIST 2
00010 X1      DATA 0
00020 X2      DATA 0
*
```
Two lines are listed beginning with line 1. Since line 1 does not exist, the listing begins with the first line found after line number 1.

2.
```
*10LIST 2


00010 X1      DATA 0
00020 X2      DATA 0
*
```
Two lines are listed, beginning with line 10.

```
LIST
```

3.
```
*LIST
00020 X2      DATA0
*
```
The current line is displayed. (In the previous example, it would be line 20.)

4.
```
*LIST 99
00010 X1      DATA 0
00020 X2      DATA 0
00030 X3ART   DATA 0
00040 X4      DTA 0
00050         PSEG
00060 START   IN   X1,PA1
00070         LT   X4
00080         ZAC
00090         MPYK 6
00100         LTD  X3
00110         MPYK 5
00120         END
*
```
A request was made to list 99 lines, starting at line number 1, but the end of text file was encountered first.

Parameters: None

Syntax M[EMORY]

Purpose To display the number of bytes of RAM available for text storage. The number of bytes is in decimal.

Notes

1) The available RAM space for text storage is a maximum just after execution of the ZERO command. It is equal to the total amount of user RAM detected by the memory sizing routine at power-up/RESET, less the system RAM.

2) When the text editor is entered with a downlink of text from either Ports 1 or 2, the remaining free RAM value is printed out on the line above the * prompt.

Examples:

1.
```
*MEMORY
15000 FREE BYTES
*
```
The text editor is initialized. It has 15008 bytes of usable RAM space available.

2.
```
*MEMORY
14966 FREE BYTES
*
```
The user had entered a line of text 38 characters long. Each line stored requires two bytes for text editor use, two bytes for line number, the text, and the line ending ‹CR›.

Parameters: Output port (optional) default = 0

Syntax Q[UIT] [output port]

Purpose To exit the text editor and re-enter the monitor program. If a port is specified, the text file is dumped to that port prior to re-entering the monitor program.

Notes

1) Because the text editor does not destroy its internal pointers, the user can exit and enter it at will, as long as nothing occurs to alter the contents of RAM. As an example, the QUIT command gives the user the means of saving the text file to cassette tape (Port 3. , then immediately executing the editor again from the monitor, then quiting the editor again to get a complete listing at either the terminal at Port 1 or a printer at Port 2.

2) If a zero (default value) is entered for the port number, no dump takes place and control returns directly to the monitor program.

3) Pressing the <ESC> key during a dump associated with a QUIT will abort the dump and return control to the monitor program. This action will have no effect on the text in RAM.

4) If the dump is to Port 1, pressing the <SP> bar will start and stop the dump.

5) The dump can be to Port 4 if an 8K byte static ram chip as described in Section 4.2.1 is placed in the EPROM programming socket.

Examples:

1.
```
*QUIT 0

** TMS320 EVM MONITOR **

[ Monitor Menu "

?


QUIT
```
The text editor is quit and the monitor program re-entered without a dump of the text. text remains intact in RAM.

2.
```
*QUIT 3

FILENAME: TEST
READY TO RECORD? <CR>

** TMS320 EVM MONITOR **

[ Monitor Menu "

?
```
The text editor is quit and text is dumped to Port 3 (cassette tape) to a file named TEST. text remains intact in RAM.

3.
```
*Q 1

** TMS320 EVM TEXT EDITOR **

>
00010        LAC  X,15     DIVIDEND INTO ACC
00020        LARK 0,15     USE AR0 AS COUNTER
00030        LARP 0        SET ARP TO AR0
00040 LOOP   SUBC Y        SUBTRACT CONDITIONALLY
00050        BANZ LOOP     IF AR0 <> ZERO, DECREMENT
00060 *                    AND GO TO LOOP
<
```

```
** TMS320 EVM TEXT EDITOR **

[ Monitor Menu "

?

4. *Q4                          QUIT THE EDITOR TO PORT 4
```

In the example above, note the > and the < bracketing the text file. These marks are used by the text editor to "sync up" to the incoming ASCII string during the text load from Ports 2 or 3. If the user prepares a text file on another computer for download to either the EVM text editor or the EVM assembler, the file should start with ><CR> and end with <<CR> to insure proper acceptance by the text editor. In addition, the text editor inserts a space between the line number and the text when dumping a file, and strips off the first character after the last line number (expecting it to be a space) during downlink.

Parameters: 1) Line number preceding command (optional)
2) Resequence increment (optional) default = 10

Syntax [line#]R[ESEQUENCE] [increment] (terminals < = 1200 baud)

[line#]RESEQUENCE<CR><prompt>[increment] (terminals >1200 baud)

Purpose To resequence line numbers of text in memory.

Notes

1) If no line number is entered, all lines are resequenced. If a line number is entered, all lines are resequenced with that line number becoming the first line number.

2) If no increment is entered, 10 is assumed. Legal increment range is from 1 to 100. Entering zero for the increment aborts the command.

3) If execution of the command causes a line number to exceed the maximum of 65335, a LINE NUMBER ERROR message is displayed.

4) The prompt will not be displayed for terminals running less than or equal to 1200 baud (see also the FIND command). When the prompt is displayed for high baud rates, an increment may be entered on the same line as the command.

Examples:

1.
```
*RESEQUENCE
*
```
All lines of text are resequenced in memory. Increment defaulted to 10, i.e., first line number will be 00010. The user prompt is not displayed because the baud rate is less-than or equal-to 1200.

```
RESEQUE
```

2.
```
*RESEQUENCE 100
*
```
All lines of text in memory are resequenced in increments of 100. The first line number is 00100. The user prompt is not displayed because the baud rate is less-than or equal-to 1200 baud.

3.
```
*RESEQUENCE

LEGAL VALUES ARE 1-100, DEFAULT IS 10
ENTER RESEQUENCING VALUE: 50
*
```
All lines are resequenced by 50. The user prompt is displayed because the baud rate is less than 1200.

4.
```
*R

LEGAL VALUES ARE 1-100, DEFAULT IS 10
ENTER RESEQUENCING VALUE: 100

LINE NUMBER ERROR
*
```
The line number register exceeds 65535. When this occurs, resequencing stops with the current line pointer at the line just prior to the overflow. The user can list the line to determine a smaller increment that will work (see LIST command). The user must resequence immediately and repeat until the error condition disappears.

Parameters: None

Syntax TABS

Purpose To allow the user to display and modify the current terminal tabs.

Notes

1) Upon execution of the TABS command, the first tab setting is displayed. A tab setting can be changed by entering the new tab setting after the current tab setting is displayed.

2) After a new tab setting is entered, terminating the entry with a comma (,) will cause the tab to be redisplayed. Entering a <CR> will terminate the command and return control to the editor. Entering a Next Entry character (<SP>,<LF>, cursor down) will display the next tab. Entry of a Previous Entry character (CNTL/K,Z or cursor up) will display the previous tab.

3) Tab settings must be in increasing value and less than or equal to decimal 70. Violation of either constraint will cause the tabs to be reset to their default value (8, 14, and 30).

Examples:

1.
```
*TABS                The tab settings are displayed.
TAB1 = 8 <SP>
TAB2 = 14<SP>
TAB3 = 30<SP>
*
```

2.
```
*TABS                The current tab setting is
TAB1 = 8 10,<CR>     changed from 8 to 10. Entering
TAB1 = 10<CR>        the comma after the 10 causes
*                    the changed tab to be displayed.
                     The command is terminated by a
                     <CR>.
```

Parameters: None

Syntax ZERO

Purpose To clear all text from memory and to initialize the text editor workspace pointers.

Notes

1) This command executes automatically prior to loading text from Ports 2 or 3 when the editor is executed.

2) This command is automatically executed when entereing the editor from Port 1 for the first time after power up, but not on a RESET.

3) A RESET does not affect the contents of the editor after the initial RESET. 4. The user may execute this command at any time to cause the editor to erase the contents of the current text file.

Examples:

```
*ZERO
ARE YOU SURE? (NO) Y

15000 FREE BYTES
*
```

When the command is executed, a prompt is issued which gives the user a chance not to initialize (thereby not losing the contents of the current text file). A YES input to the prompt erases the file and initializes the editor; any other entry leaves the editor unchanged. The number displayed under the prompt is the remaining useable RAM.

Parameters: None

Syntax [line#]<CR>

Purpose To delete a line of text in memory.

Note If the line does not exist, the command will abort. If the line exists, it is deleted from the text file. If the [Line#]<CR> is used to exit the autoincrement mode, the line will not be deleted if it exists.

Examples:

1.
```
*70<CR>
*
```
Line 70 deleted from text file.

2.
```
*180<CR>
LINE NUMBER ERROR
*
```
Attempt was made to delete a number that does not exist in the text file. The editor is left intact.

Parameters: None

Syntax <ESC>

Purpose To allow the user to abort any activity in the text editor when a character input is expected. The monitor equivalent to this command is also the Escape Key.

Note This special character is tested for in the input routine and causes a jump to the command handler of the text editor. See also DUPLICATE and EDIT commands.

Examples:

1.
```
*10L 9<ESC>
*
```
A LIST command is aborted, and a prompt is issued for next input.

2.
```
*60E
00060     ZALS A2
00060     ZALS A1<ESC>
*
```
An edit session is aborted after user changed A2 to A1, then decided to leave the line as it originally was.

5.4 TEXT EDITOR ERROR MESSAGES

The text editor issues three error messages. They are:

```
INPUT ERROR
RAM FULL
LINE NUMBER ERROR
```

An explanation of each error message is given in the following subsections.

5.4.1 INPUT FULL Error Message

This is the general input error message issued by the text editor when an illegal character is entered as a parameter in a command string.

5.4.2 RAM FULL Error Message

This error message is issued whenever the amount of available RAM for text storage (as displayed with the MEMORY command) drops below 64 bytes. The error message will continue to be issued while still allowing the user to add/edit lines to the text file until RAM is exhausted. If a file is loaded into the text editor, this error is issued when the RAM is filled, at which point, the load is aborted and control returned to the text editor command handler.

5.4.3 LINE NUMBER ERROR Message

This error is issued whenever an operation involving line numbers causes the register holding the line number to underflow below 000001, or overflow above 65535. The error triggering the issuance of this message under certain conditions will cause the following actions to occur:

1) During a downlink with the editor creating line numbers, this error will terminate the mode.

2) In autoincrement mode, this error will terminate the mode.

3) During execution of a RESEQUENCE line numbers command, this error will terminate the command. At that point, the line number holding the register will point to the line just before the overflow occurred (to allow the user to determine where the error occurred with respect to the end of the file).

4) This error will terminate the CHANGE and DUPLICATE command if the line to be changed or copied does not exist, or if the destination line number does not exist.

6. THE TMS2764 PROM UTILITY

6.1 INTRODUCTION

The TMS32010 EVM is equipped to program TMS2764 EPROMs. Four basic features are available:

- The ability to program an EPROM with the contents of the designated memory addresses. When programming is complete, the EPROM programmer will automatically execute a compare routine.
- A compare routine which matches the newly-programmed contents of an EPROM to the contents of the memory addresses to verify the copy.
- The ability to read the EPROM contents into memory.
- The ability to verify that the EPROM is erased (all locations = >FF).

6.1.1 On-Board Power Supply

The EVM is equipped to provide an on-board power supply of the +21 volts required for programming EPROMs. It is recommended that a period check of the voltage be made to ensure good programming.

> **CAUTION**
>
> The PROG LED will be lit any time +21 volts is applied to the EPROM socket. Be careful not to plug an EPROM into the socket if the PROG LED is on; the bits on the EPROM can be mistakenly programmed if the EPROM is inserted in the socket while the PROG LED is lit.

6.1.2 EPROM Programming Procedure

When an EPROM is placed in the socket, the following commands (see 6.2) are used to program an EPROM:

- VRFY Command to verify that the EPROM is erased. If an error is detected, it will be displayed in the following format:

```
ERROR   P:YYYY = yy     Where P:YYYY = yy is the PROM byte
                        address and data at that address.
```

- PROG command to program/reprogram the EPROM from a specified block of memory. As a time-saving measure, if a memory location contains >FF, the programming routine skips the corresponding location in the EPROM, assuming it to already be >FF (the erased state).
- COMP command to compare the contents of the EPROM to a specified block of memory
- READ command to read the contents of the EPROM to a specified block of memory

At the completion of execution of the programming command, a PROGRAMMING COMPLETE messge will display at the terminal. Immediately following appearance of the message, the EVM will execute a compare EPROM command, using the block limits specified in the programming command. When the compare is complete, a COMPARE COMPLETE message will display, if the EPROM programmed successfully. If it did not, and any errors are detected, the location in the EPROM and the data associated with that location will be printed, along with the corresponding location in memory and its associated data. The format for the display of a compare error is:

```
ERROR   M:XXX = xx      P:YYYY = yy
```

where M:XXX = xx is the RAM word address and data at that address, and P:YYYY = yy is the PROM byte address and data at that address. The cursor will stop at the end of the line. To examine subsequent error locations, press the <SP> bar and repeat as often as necessary to display all the errors. The message: COMPARE COMPLETE will display at completion. If a <CR> is entered, the compare process will be terminated.

6.1.3 Programming a Byte-Wide EPROM with Word-Wide RAM

Programming a byte-wide EPROM with word-wide RAM is accomplished by programming each word a byte at a time, with the most significant byte first. If subsequent byte locations do not program properly, the error display will show the byte address to be one apart, while the word address is the same for both bytes.

6.2 PROM UTILITY COMMAND DESCRIPTIONS

Table 6-1 lists the PROM utility commands.

TABLE 6-1 - PROM UTILITY COMMANDS

COMMAND	MNEMONIC
Compare EPROM to Memory	COMP
Display PROM Utility Menu	MENU
Program EPROM from Memory	PROG
Execute the PROM Utility	PROM
Quit PROM Utility	QUIT
Read EPROM to RAM	READ
Verify EPROM Erased	VRFY
EPROM Byte Offset	OFSET

A detailed description of each command is on the following pages. The commands are also summarized in Appendix B.

Parameters

1) EPROM start address (optional) default = 0
2) EPROM end address (optional) default = >1FFF
3) Memory start address (optional) default = 0

Syntax

COMP [EPROM st addr] [EPROM end addr] [memory st addr]

Purpose

To allow the user to compare the contents of the EPROM to the contents of memory.

Notes

1) The legal range of the EPROM addresses is from >0000 to >1FFF. The legal range of memory addresses is from >0000 to >0FFF.

2) When the locations do not compare successfully, both addresses and both values are displayed and the routine halts. At that point, a <CR> terminates the compare operation; pressing any other key allows the operation to continue.

Examples

```
1.   .COMP<CR>                    The contents of the EPROM
     COMPARE COMPLETE             are successfully compared
                                  to the contents of memory.


2.  .COMP 1000 AFFF 800<CR>
    ERROR   M:003 = CA   P:0006 = CD<CR>
```

In the above example, the operation was aborted after the first error was detected.

Parameters None

Syntax MENU

Purpose To display the PROM utility menu.

Example

```
.MENU<CR>                                  Menu is called.

** TMS320 EVM PROM UTILITY  **

"QUIT"   RETURN TO MONITOR
"MENU"   THIS DISPLAY                    > Display
"PROG"   PROGRAM PROM
"READ"   READ PROM
"COMP"   COMPARE PROM
"VRFY"   VERIFY PROM
"OFSET"   0-NONE/1-LSB/2-MSB
```

The EPROM programming utility ("PROM") contains an additional command to select or deselect an offset for programming byte-wide EPROMs as either high or low byte in a 16 bit wide system.

Command Name OFSET

Parameters 1) 0 - no offset
1 - odd offset
2 - even offset

Format CMD <OFFSET>

NO OFFSET: Each TMS320 16 bit word is programmed into two successive EPROM 8 bit locations, MSB first and LSB second.

ODD OFSET: The least significant bytes (LSB) of each TMS320 16 bit word are programmed into successive EPROM 8 bit locations.

EVEN OFFSET: The most significant bytes (MSB) of each TMS320 16 bit word are programmed into successive EPROM 8 bit locations.

Each time the EPROM utility is executed, the offset is reset to 0 (no offset). The offset must be initialized by the user each time it is used until the EPROM utility is exited. For odd and even offset applications, only the lower 4K bytes in the 8K byte EPROM are programmed. The current value of the offset can not be displayed and a parameter (0, 1, 2) must be entered with the "OFSET" command or PARAMETER ERROR is issued.

Parameters

1) EPROM start address (optional) default = 0
2) EPROM end address (optional) default = >1FFF
3) Memory start address (optional) default = 0

Syntax

PROG [EPROM st addr] [EPROM end addr] [memory st addr]

Purpose

To allow the user to program a TMS2764 EPROM.

Notes

1) The range of the EPROM addresses is from >0000 to >1FFF. The range of memory address is from >0000 to >0FFF.

2) The programming routine automatically enters the COMP command after the programming process is complete. When a mismatch occurs, the comparison can be aborted by pressing <CR>, or can be continued by pressing any other key.

3) Pressing the <ESC> key, or performing a RESET at any time during execution of this command, terminates the command and returns control to the PROM Utility top level. A subsequent RESET will cause a system reset.

Examples

```
1.  .PROG<CR>                    The entire 4K words of
    PROGRAMMING COMPLETE         TMS32010 RAM are pro-
    COMPARE COMPLETE             grammed into 8K bytes
                                 of EPROM.
    .

2.  .PROG 1000 1FFF 800<CR>
    PROGRAMMING COMPLETE
    ERROR   M:003 = CA   P:0006 = CD<SP>
    COMPARE COMPLETE

    .
```

In the above example, an attempt was made to program the top half of the RAM into the top half of the EPROM. One error occurred. Pressing the <SP> bar continues the comparison.

Parameters None

Syntax PROM

Purpose To invoke the PROM Utility.

Notes

1) The PROM Utility is executed from the monitor.

2) The period is the PROM Utility prompt (see example below).

3) The menu is automatically displayed for baud rates greater that 1200 baud, but can be requested at any time for any baud rate (see MENU command).

Examples

```
?PROM<CR>

** TMS320 EVM PROM UTILITY **

[ PROM utility menu displays if baud rate > 1200 '
```

Parameters None

Syntax QUIT

Purpose To return control to the EVM monitor program.

Examples

```
.QUIT<CR>                          User has quit the PROM
                                   utility.
[ Monitor menu is displayed
  if baud rate > 1200 '

?                                   Monitor prompt
```

Parameters

1) EPROM start address (optional) default = 0
2) EPROM end address (optional) default = >1FFF
3) RAM start address (optional) default = 0

Syntax READ [EPROM st addr] [EPROM end addr] [RAM st addr]

Purpose To allow the user to read the contents of a TMS2764 EPROM into TMS32010 memory.

Note The legal range of the EPROM addresses if from >0000 to >1FFF. The legal range of RAM addresses is from >0000 to >0FFF.

```
        +-+-Byte Address
        ¦ ¦
        ¦ ¦    +-Word Address
        ¦ ¦    ¦
        V V    V
```

Example

```
.READ 0 7FF 800<CR>
READ COMPLETE
```

The lower fourth of a TMS2764 EPROM is read into the upper half of the TMS32010 program RAM.

Parameters

1) EPROM start address (optional) default = 0
2) EPROM end address (optional) default = 0

Syntax

VRFY [EPROM st addr] [EPROM end addr]

Purpose

To allow the user to verify that the 2764 EPROM is clear (>FF in all locations).

Note

The legal range of EPROM addresses is from >0000 to >1FFF.

Examples

```
1.   .VRFY 0 1FFF<CR>          The EPROM is verified
     VERIFY COMPLETE           clear.


2.   .VRFY 0 1FFF<CR>          An error is found, and
     ERROR   P:0006 CD<CR>     the verify operation
                               is terminated.
```

6.3 SYSTEM ACCESS FROM PROM UTILITY COMMANDS

Three commands in the PROM utility accept the system access dollar sign prefix ($). These commands are: PROG, COMP, and READ. The VRFY command is not included, since all its parameters are EPROM addresses. As with the regular monitor system access commands, PROM utility access commands remove RAM address parameter restrictions, thereby allowing duplication of operating system EPROMs, etc. In the system access mode, the 0 to >FFF two-port TMS32010 RAM address range becomes >A000 to >BFFF byte addresses of the TMS9995 master processor of the EVM. default values for EPROM address parameters are the same as described in the above command descriptions. RAM address parameters still default to zero (zero in this case being the TMS9995 zero address). Error messages for $ commands contain four-digit RAM addresses. The following examples illustrate the $ PROM utility commands:

```
1.  .$PROG 0 FFF 2000          The second monitor  EPROM
    PROGRAMMING COMPLETE       is duplicated.
    COMPARE COMPLETE


2.  .$READ 0 FFF A000          The contents of the  EPROM
     READ COMPLETE             are read into the two-port
                               RAM.


3.  .$COMP 0 FFF A000          The EPROM is compared with
    COMPARE COMPLETE           the two-port RAM.
```

6.4 PROM UTILITY ERROR MESSAGES

Other than the programming and compare errors discussed in the command descriptions, four other error messages may be issued by the command interpreter. They are listed and described in Table 6-2.

TABLE 6-2 - PROM UTILITY ERROR MESSAGES

ERROR MESSAGE	DESCRIPTION
COMMAND ERROR	The command interpreter was not able to interpret the command.
PARAMETER ERROR	The parameter analyzer associated with the command received a character that was not valid for the type of data expected (Example: a Z for a hex input).
VALUE ERROR	A parameter was expected but not entered.
ADDRESS ERROR	The entered address is out of the legal range for the operation. Legal range is: 0 - >1FFF.

7. IN-CIRCUIT EMULATION

7.1 INTRODUCTION

The TMS32010 EVM is designed to emulate a user program in one of two modes: (1) using the internal clock and internal 4K-word memory (default mode), or (2) allowing for an external clock and/or external memory space (see INIT command). Emulation is achieved through the 40-pin emulation cable attached to the board. The emulation cable provides GND reference to the target system on pins 7, 10, and 30. No power connection is available through the ICE cable.

7.2 CONNECTING AN EXTERNAL CLOCK TO THE EVM

If the EVM is initialized to execute with an external clock, the signal must be applied to the emulation cable. Section 9 contains descriptions of cable switch settings and the oscillator connection pins.

7.3 LIMITATIONS

Since the monitor commands dealing with program memory are designed to interact with internal program memory only, they cannot be used with external program memory selected by the INIT monitor command. In addition, the reverse assembly provided during single-step and breakpoints is disabled when executing out of external program memory.

8. THE AUDIO TAPE SYSTEM

8.1 INTRODUCTION

This section discusses the features of the audio cassette tape system that may be connected to the EVM for use as a mass storage device. Also discussed are various features of the system.

The audio cassette tape system supports files by name and type. Filenames consist of one to five alphanumeric characters and are stored to tape with a file type descriptor associated with the command that created the file. Three types of files can be created and stored on tape. Table 8-1 lists them, along with their creation and reading commands.

TABLE 8-1 - TAPE SYSTEM FILE TYPES

FILE TYPE	CREATING COMMAND	READING COMMAND
SOURCE	Q 3 (Quit Editor)	EDIT 3 (Execute Editor) ASM 3 1 (Ex Assembler)
OBJECT	SPM 3 (Dump Object)	LPM 3 (Load Object)
STATE	SMS 3 (Dump State)	LMS 3 (Load State)

8.1.1 Using the <ESC> Key and RESET Switch

Whenever the audio tape system is active (READ/WRITE/DIRECTORY), pressing the <ESC> key or the RESET switch causes an abort of the activity. When this is done, the monitor prompt (?) is immediately displayed. This process allows the user to recover from a READ operation that cannot find the file in question without re-initializing the EVM. This feature and its use during the DIRECTORY operation is discussed in Section 8.4.

8.1.2 The TAPE ERROR Message

The TAPE ERROR message is triggered whenever computed and actual checksums differ during a READ operation. The message includes an approximate RAM address of the error. The operation then continues.

8.1.3 The Tape System LED

One LED reflects the tape operation (MOTOR). When lit, it indicates that the motor relay has engaged the tape cassette motor.

8.2 SAVING FILES TO TAPE

When a file is created by issuing a command with the output port specified as Port 3, the EVM responds with the following prompt:

```
FILENAME:
```

A filename of up to five characters maximum must be entered, followed by a <CR>. For input operations, the asterisk (*) is used to denote the wildcard (next appropriate) file. The asterisk can be used within any file name, but never as the first character. Once the filename is entered, the EVM will prompt with:

```
READY TO RECORD? (Y/N)
```

This is a reminder to put the cassette player on "Record". Once the correct response is entered, command execution continues and the recorder motor is turned on. Dumping of data to tape begins after a brief time delay (approximately 1.5 seconds) to allow the motors to get up to speed.

8.3 READING FILES

Figure 8-1 illustrates a typical audio taped data block.

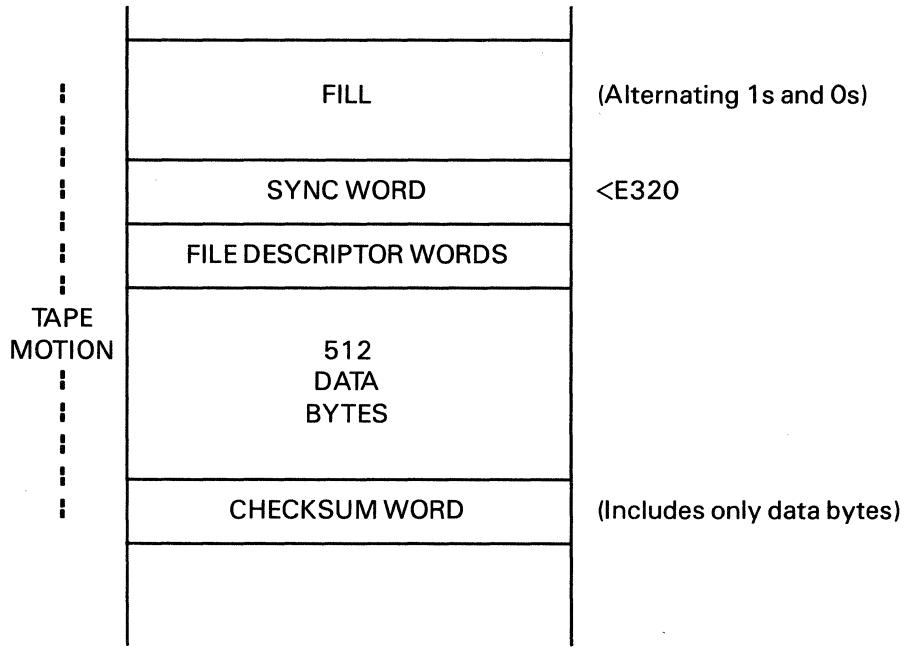


FIGURE 8-1 - TYPICAL AUDIO TAPE DATA BLOCK

When a file is loaded by issuing a command with the input port specified as Port 3, the EVM responds with the following prompt:

```
FILENAME:
```

A filename of up to five characters maximum may be entered, followed by a <CR>. After entry of the <CR>, the EVM begins searching the tape for a file with the given name that is of the type required by the command being executed (see Table 8-1). If the file is not found, an <ESC> or RESET will have to be performed to abort the execution of the command and return control to the monitor (the ? prompt will display).

As an option, the FILENAME prompt may be answered by entering an asterisk (*), the wildcard file name. This action will cause the first file encountered of the proper type to be loaded.

While loading is in progress, the number of input buffers (512 bytes) loaded will be counted as an indicator that the file has been located. Since the tape load routine starts to look for a file beginning at the current position of the tape, file names of the same name and type can be stored on the tape. Before starting the load operation, the tape can be positioned to miss an unwanted file either by using the DIRECTORY operation or

the same name for reference. The EDIT and ASM commands will look for a source file, the LPM command will look for an object file, and the LSM command for a state file.

8.4 THE AUDIO TAPE DIRECTORY

The EVM monitor tape DIRECTORY command (DIR <port>) allows the user to keep a record of files stored to tape. Performing a DIR or Port 2 allows the user with a printer to produce hard copy for storage with the tape.

During a DIRECTORY operation, the file name and type are printed out as they are first encountered. The number of blocks in the file is printed out last, when the last block is encountered. In this way, the DIRECTORY command can be used to position the tape at the logical end-of-tape for storage of a file. If a short file is written over part of a longer file, the DIRECTORY command will terminate the longer file entry with the current block count and initialize a new entry with the new file name. When the shorter file is passed, the remainder of the old, longer file will be displayed.

Since no motor control other than ON/OFF is available to the EVM, the monitor has no way of maintaining logical end-of-tape marks for purposes of terminating the DIRECTORY command. Therefore, the operation must be terminated either by pressing the <ESC> key or by toggling the RESET switch. When either action is performed from the DIRECTORY operation, control is passed directly to the monitor command handler, and the ? prompt is displayed.

The following is a sample of the DIRECTORY command output:

```
?DIR‹CR›

AUDIO TAPE DIRECTORY

16      TEST1.SOURCE
4       TEST1.OBJECT
3       TEST1.STATE
17      GAME.SOURCE
4       GAME.OBJECT

‹ESC› OR [ RESET ]

?
```

8.5 THE MOTOR CONTROL UTILITY

Since the tape cassette motor is under program control, it is only enabled when the EVM is attempting to create or load a file. To provide a way to enable the motor without disconnecting the motor control cable, the monitor command MO enables the motor until a subsequent MO command disables it, thus providing a direct way to rewind tapes for storage and/or to fast-forward a tape past the clear leader prior to storing a file at the beginning of the tape.

9. EVM HARDWARE FUNCTIONAL DESCRIPTION

9.1 INTRODUCTION

The operation of the TMS32010 EVM is controlled by a TMS9995 microprocessor that uses a system of addressable latches, multiplexers, and memory to send and receive control signals to the various segments of the EVM. Each of the latches and the multiplexers is mapped into the CRU space of the TMS9995. The details of this system are explained in this section.

9.2 MEMORY

Figure 9-1 illustrates the EVM memory configuration.

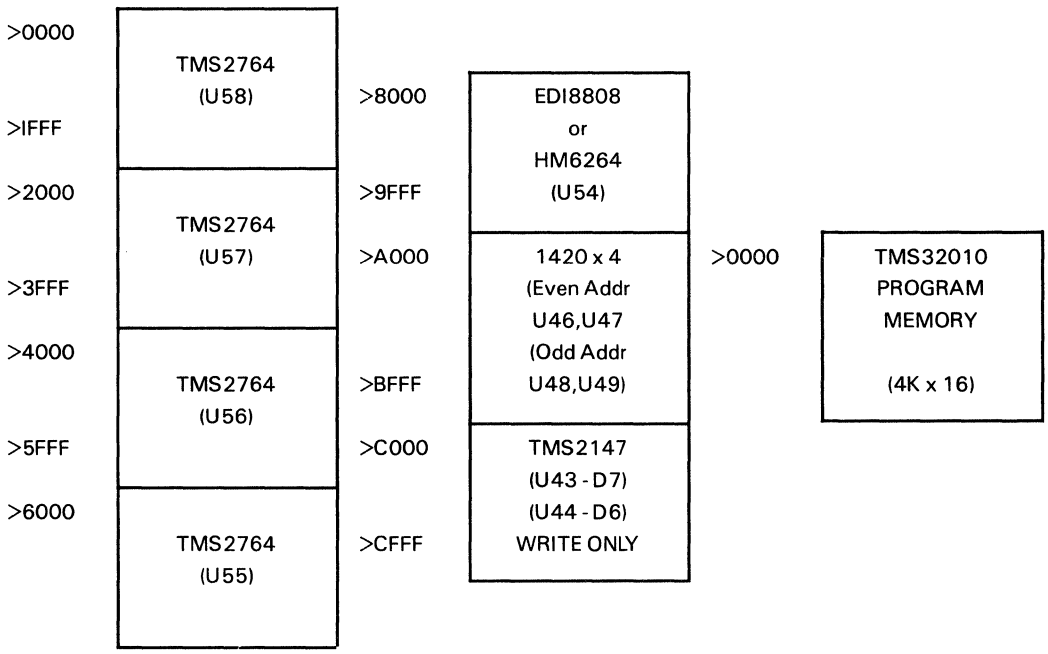


FIGURE 9-1 - EVM MEMORY CONFIGURATION

Figure 9-2 illustrates the EVM CRU memory map. The TMS9995 has access to 32K bytes of EPROM (four TMS2764s) that contain the monitor, assembler, text editor, etc., and 8K bytes of RAM (an HM6264 or an EDI8808) that are used by the resident software and the user for text editing and program loading. The TMS9995 can also access 8K bytes of fast RAM (four IMS1420s), called dual-port RAM, since the TMS32010 also has access to it. The dual-port RAM is used by the TMS32010 as its program memory to enable the TMS9995 to easily load a user program for execution. Additonal RAM exists for the purpose of program execution initiation and breakpoints (see section 9.3).

	FACILITY	BIT MAP							
>0000 >01FF	TMS9902 (Terminal)								
>0200 >02FF	J2 TMS9902 (Host)								
>0400 >0407	EPROM PROGMR (Data)	400 PD0	401 PD1	402 PD2	403 PD3	404 PD4	405 PD5	406 PD6	407 PD7
>0480 >0487	EPROM PROGMR ADDRESS LSB (Write Only)	480 PA0	481 PA1	482 PA2	483 PA3	484 PA4	485 PA5	486 PA6	487 PA7
>0480 >0487	AUDIO INPUT (Read Only)	480	481	482	483	484 Reserved	485	486	487 Audio In
>0500 >0507	EPROM PROGMR ADDRESS MSB & CONTROLS (Write Only)	500 PA8	501 PA9	502 PA10	503 PA11	504 PA12	505 PROG-	506 PSEL	507 5/21V
>580	CONTROL SIGNALS	580 Motor	581 Audio Out	582 ENRD-	583 ENWR-	584 ARM-	585 TGTIN	586 TGEN	587 HMACC

FIGURE 9-2 - EVM CRU ADDRESS MAP

9.3 PROGRAM EXECUTION AND BREAKPOINT LOGIC

The EVM is designed so that the TMS32010 is always executing code, either NOPs from two 74LS244 drivers, a user program on a target system (see Section 9.7), or a user program out of dual-port RAM on the EVM. What code will be executed is determined by whether there is a start point set, a stop point (breakpoint) set, or if target memory is enabled. If no start points are set, the 74LS244 output is enabled, thereby permanently placing a NOP on the TMS32010 data bus and thus causing the TMS32010 to continually execute NOPs.

A program is executed by first loading the program into the dual-port RAM, setting a start point, and, if desired, a breakpoint. A breakpoint is set by writing a 1 to the desired breakpoint address(es) in a TMS2147H-3 high-speed 4K x 1 RAM. The breakpoint RAM is mapped to >A0000 through >BFFE of the TMS9995 memory space and uses bit 7 of the data bus. (The correct breakpoint address is derived by multiplying the TMS32010 address by two and offsetting this by >A000.) A program start point is set in the same way except the start point RAM uses bit 6 of the TMS9995 data bus.

When the TMS32010 address bus matches a start point, the data-out bit of the start point RAM goes high, causing the start flip/flop (a 74ALS112) output to disable the NOPs and enable the dual-port RAM data onto the TMS32010 data bus. When the TMS32010 address bus matches a breakpoint address, the high data-out bit causes the breakpoint flip/flop to disable the dual-port RAM and enable the hardwired NOP opcode from the 74LS244s back onto the TMS32010 data bus. The output of this flip/flop also causes the address, BIO-and INT- to be latched into two 74LS374s. They exist in the TMS9995 CRU space as follows:

```
A0  - A7    at bits  >E000 - >E007
A8  - ALL   at bits  >C000 - C003
B10         at bit   >C004
INT         at bit   >C005
```

A special control bit exists at CRU bit >584 (ARM-) that when set low clears both the start and stop flip/flops and ensures execution of NOPs. This bit must return to a high state before the user program can run again.

9.4 COMMUNICATION PORTS

The EVM has two RS232 communication ports (J1 and J2) for communication with a terminal and a host system respectively. A TMS9902A asynchronous communications controller at each port provides the RS232 data format and signals. The connector at each port is configured as shown in Table 9-1.

The Clear To Send and Data Set Ready signals are pulled high on the EVM. The Release Data signal will be high when the EVM is ready to send or receive data. The Clear To Send and the Request To Send signals of the TMS99202A (both active low) wire-ANDed together and inverted to generate the Release Data signal.

TABLE 9-1 - EIA CONNECTOR DESCRIPTION

PIN	SIGNAL	I/O
1	Chassis Ground	IN
2	Transmit Data	IN
3	Receive Data	OUT
5	Clear to Send	IN
6	Data Set Ready	IN
7	Signal Ground	IN
8	Release Data	OUT
20	Data Terminal Ready	IN

9.5 EPROM PROGRAMMER

The EPROM programmer is controlled and addressed via the CRU interface of the TMS9995. The address is written to two 74LS259 addressable latches. The eight low-order bits, PA7-PA0, are located at CRU bits >487 - >480. The five high-order bits, PA12-PA8, are at CRU bits >504 - >500. The data bits, PD7-PD0, of the EPROM programming socket are located at CRU bits >407 - >400.

To write data to the programmer, the ENWR- signal (CRU bit >583) must first be set low, then the data is output on the CRU bus and latched into a 74LS259, then enabled to the socket through a 74LS244. Data is read by first setting ENRD- (CRU bit >582) low, then inputting the data through a 74LS251 multiplexer via the CRU bus.

The programming control signals, PROG-, PSEL, and 5/21V are located at CRU bits >505 - >507, respectively. The 5/21V bit, when set high, causes the TL497 switching voltage regulator to produce the 21 volts needed for programming a TMS2764 EPROM. The PSEL bit enables the select line for the EPROM programming socket when set high. Setting PROG- high then low (a negative-going edge), causes the data held at the socket to be programmed into the EPROM.

9.6 AUDIO CASSETTE INTERFACE

The audio cassette interface is mapped into the CRU space of the TMS9995. Jack J3 controls the motor of the cassette. An LED indicator is provided that lights when the motor is enabled. The audio output signal is output through Jack J5 and controlled by

CRU bit >581 by setting this bit high or low, according to the data to be transmitted. The audio signal is input through Jack J4, filtered by two op-amps, an RC4558, and discrete components, and is read through CRU bit >487. The data stored on the cassette is inverted and is re-inverted upon input.

9.7 TARGET CONTROL

The target connector is added to the EVM by plugging the host side of the connector P4 into Port P3 on the EVM. The 40-pin target port, P5, plugs directly into the TMS32010 socket on the user's target system. The TMS32010 address and data buses are unbuffered and go directly to the target connector. The target cable is limited to six inches to keep the propagation delay of the TMS32010 signals to a minimum.

9.8 CLOCK CONTROL

The clock of the TMS32010 will be the 20-MHz clock provided on the board, unless the user elects to use the target system clock. The target system clock and the target BIO- and INT- signals are selected by setting TGTIN (CRU bit >585) high; when this bit is low, the EVM signals and clock are used.

The target system clock can either emulate the internal oscillator of the TMS32010 or provide an external clock signal. The internal oscillator option is selected by closing switches 1 through 3 of the dip switch (SW2) on the target connector board and by connecting the desired crystal between pins 7 and 8 of the target port P5 (the TMS32010 X1 and X2/CLKIN pins). The external clock signal option is selected by opening switches 1 through 3 of SW2 and providing the clock signal at pin 8 of P5. Table 9-2 defines the SW2 switch settings.

TABLE 9-2 - CLOCK/OSC SWITCH SETTINGS

OPTION	SWITCHES			
	S1	S2	S3	S4
Internal Oscillator	OFF	OFF	OFF	X
External Clock	ON	ON	ON	X

Target system memory is enabled by setting CRU bits >586 (TGEN-) and >582 (ENRD-) high. The TGEN- signal enables the memory interface signal that is necessary to reach the target connector, and the ENRD- signal enables DEN- to the target connector.

A. COMPONENT ASSEMBLY AND SCHEMATICS

NOTES, UNLESS OTHERWISE SPECIFIED:

1. MAX. LEAD LENGTH TO BE .062 FROM CONDUCTOR SIDE OF BOARD
2. DO NOT SOLDER ON COMPONENT SIDE

[3] INSTALL AFTER PROCESS 1

[4] ZIF SOCKET (IT.69) MUST BE SOLDERED WITH THE CONTACTS IN THE "OPEN" POSITION

REVISIONS			
REV	DESCRIPTION	DATE	APPROVED
A	INFORMAL DESIGN CHG	6/83	
B	INFORMAL DESIGN CHG	8/83	
C	PRODUCTION RELEASE	9/8/83	
D	CN495307 9/28/83	9/29/83	
E	CN495356 Nelson 11/2/83		
UPDATED PICTORIAL AND LM PER REV E PWB			

CONVERSION CHART	
INCHES	mm
.010	0.25
.02	0.5
.062	1.57

REVISION LEVEL CONTROL BLOCK									
PWB	1605014	B	C	D	E				
DIAG	1605016	F	F	G	H				
ASSY	1605015	B	C	D	E				
S/W REL	1605022	1.2	1.2	1.2	1.2				

SEQ NO	IDENT	F-SPEC	NO	ADDITIONAL	NOTES
1	SLDR	124-02	00	FLOW SOLDER	

PROCESS — CLASSIFICATION

PROCESSES — FOR CORRELATION TO GOVT/IND SPECIFICATIONS, SEE TI DRAWING 729467

QTY	ITEM NO	PART OR IDENTIFYING NUMBER	NOMENCLATURE OR DESCRIPTION	PROCUREMENT SPECIFICATION	NOTES

PARTS LIST

NEXT ASSY	USED ON
1605029	8820 (A03)

APPLICATION

UNLESS OTHERWISE SPECIFIED
- DIMENSIONS ARE IN INCHES
- TOLERANCES: ANGLES ±1°, 3 PLACE DECIMALS ±.010, 2 PLACE DECIMALS ±.02
- INTERPRET DRAWING PER MIL-D-1000
- REMOVE ALL BURRS AND SHARP EDGES
- CONCENTRICITY MACHINED DIAMETERS .010 FIM
- DIMENSIONAL LIMITS APPLY BEFORE PROCESSES
- PARENTHETICAL INFO FOR REF ONLY

HOLE TOLERANCE

TEXAS INSTRUMENTS INCORPORATED Dallas, Texas

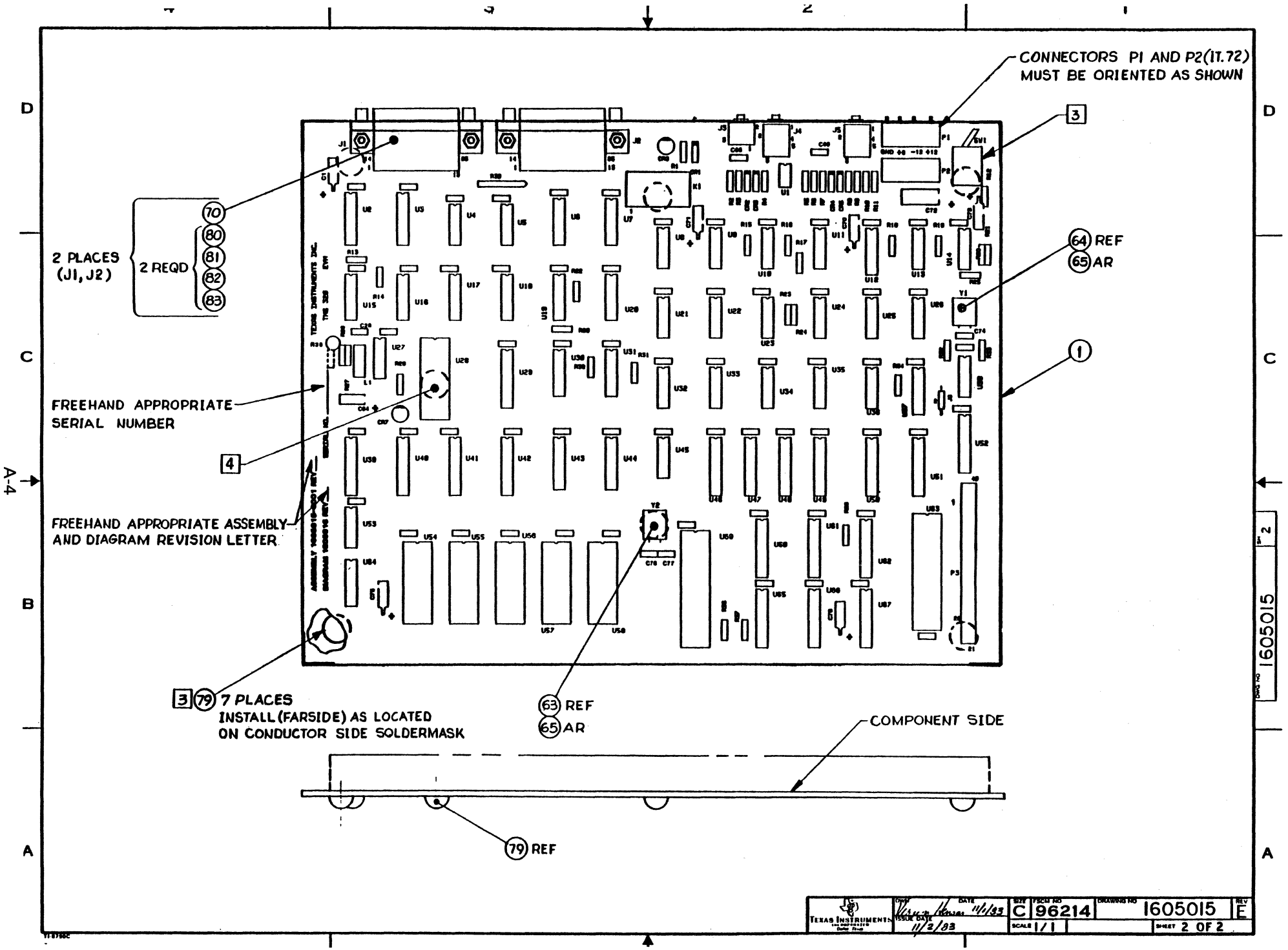
TMS320 EVM ASSEMBLY

SIZE C | FSCM NO 96214 | DRAWING NO 1605015

SCALE 1/1 | SHEET 1 OF 2

LM

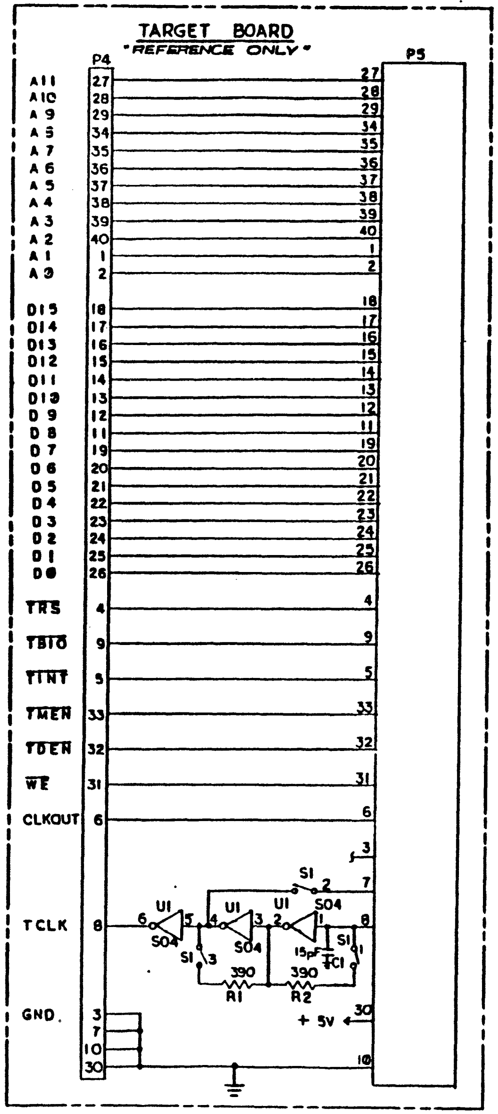
A-3



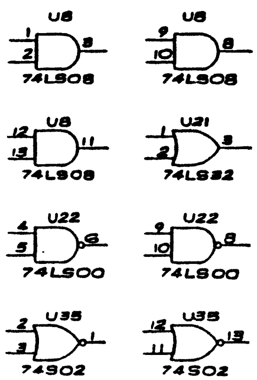
TEXAS INSTRUMENTS	DWN	DATE 11/1/93	SIZE C	FSCM NO 96214	DRAWING NO 1605015 REV E
	ISSUE DATE 11/2/93		SCALE 1/1		SHEET 2 OF 2

NOTES: UNLESS OTHERWISE SPECIFIED:

1. ALL RESISTANCE VALUES ARE IN OHMS
2. ~~RESISTOR VALUE SELECTED AT TEST~~
3. "NC" DENOTES NO CONNECTION
4. ALTERNATE VALUE FOR CAPACITOR "C72" IS 220μF
5. ALTERNATE VALUE FOR INDUCTOR "L1" IS 220μH
6. A SN74ALS244N DEVICE MAYBE SUBSTITUTED FOR THE SN74ALS241N DEVICE AT U62



SPARES (CONTINUED)

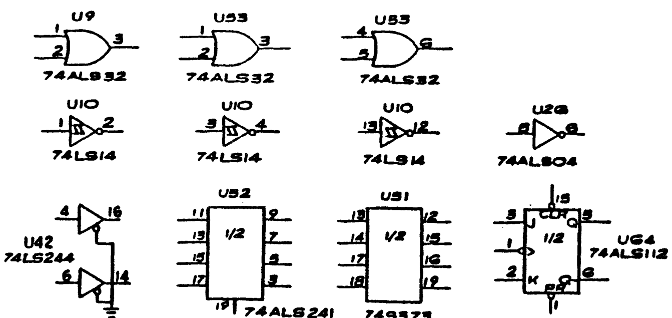


POWER AND GROUND LOCATOR CHART						
REF DES	DEVICE TYPE	POWER PINS				
		GND	+5V	+12V	-12V	V_{BB}
U59	TMS9995	31	10			
U63	TMS32010	10	30			
U2, 3	TMS9902A	9	18			
U55,56,57,58	TMS2764	14	28			
U54	HM6264	14	28			
U46,47,48,49	IMS1420-45	10	20			
U43,44	TMS2147-3	9	18			
U5	75188	7		14	1	
U4	75189	7	14			
U27	TL497	5	14			
U1	RC4558			8	4	
U26,32	74ALS04	7	14			
U11	7406	7	14			
U23, 8	74ALS08	7	14			
U10	74LS14	7	14			
U9, 21,53	74ALS32	7	14			
U13	74ALS74	7	14			
U12,64	74ALS112	8	16			
U6,19,31	74LS138	8	16			
U37	74LS157	8	16			
U29,41,42,62,67	74LS244	10	20			
U60,61,65,66	74ALS245	10	20			
U20,30	74LS251	8	16			
U7,16,17,18	74LS259	8	16			
U39,40	74LS374	10	20			
U28	EPROM SKT	14	28			1
U24,25,33,34	74S64	7	14			
U22	74ALS00	7	14			
U35,36	74S02	7	14			
U38	74S04	7	14			
U14	74F32	7	14			
U62	74ALS241 [6]	10	20			
U45	74S260	7	14			
U50,51	74S373	10	20			
U15	16PIN SR	8	16			

1605016 1

REVISIONS			
REV.	DESCRIPTION	DATE	APPROVED
A	Release to Production	12-27-82	[illegible]
B	AUDIO INPUT & OUTPUT JACK CONNECTIONS MODIFIED CAP. & RESISTOR SWITCH (R7 & C4) MADE. TWE MOVED FROM WE+ENB. 15pF CAP. ADDED TO PIN 2 OF TARGET CONN. 74LS04	3-27-83	[illegible]
C	MODIFIED TABLE WRITE CCT - REMOVED 7406'S & REPLACED WITH 74S260 & 3 STATE 74ALS08. ADDED 2 74ALS04 GATES TO NOP SIGNAL FOR DELAY TO GET RID OF BUS CONTENTION	4/22/83	[illegible]
D	CHANGED ALS to 74S04 FOR STABLE OSCILLATOR FREQUENCY. REMOVED 2 74ALS04 GATES FROM SCHEMATIC - NOT NEEDED	6/10/83	[illegible]
E	Replaced [illegible] circuit for external Program Memory on sheet 2. Removed [illegible] on 9995 side. Made provisions for ATE requirements.	7/14/83	[illegible]
F	Modified PROM enable ckt, Address Latch enable. Removed INIT signal.	7/22/83	[illegible]
G	INFORMAL CHANGE	9/22/83	[illegible]
H	CN495356	11/8/83	

SPARES



PARTS LIST

UNLESS OTHERWISE SPECIFIED

TEXAS INSTRUMENTS INCORPORATED

SI-METRIC

DIAGRAM, LOGIC
TMS320 EVM

1605015 | 8820 (A03)

NEXT ASSY | USED ON

APPLICATION

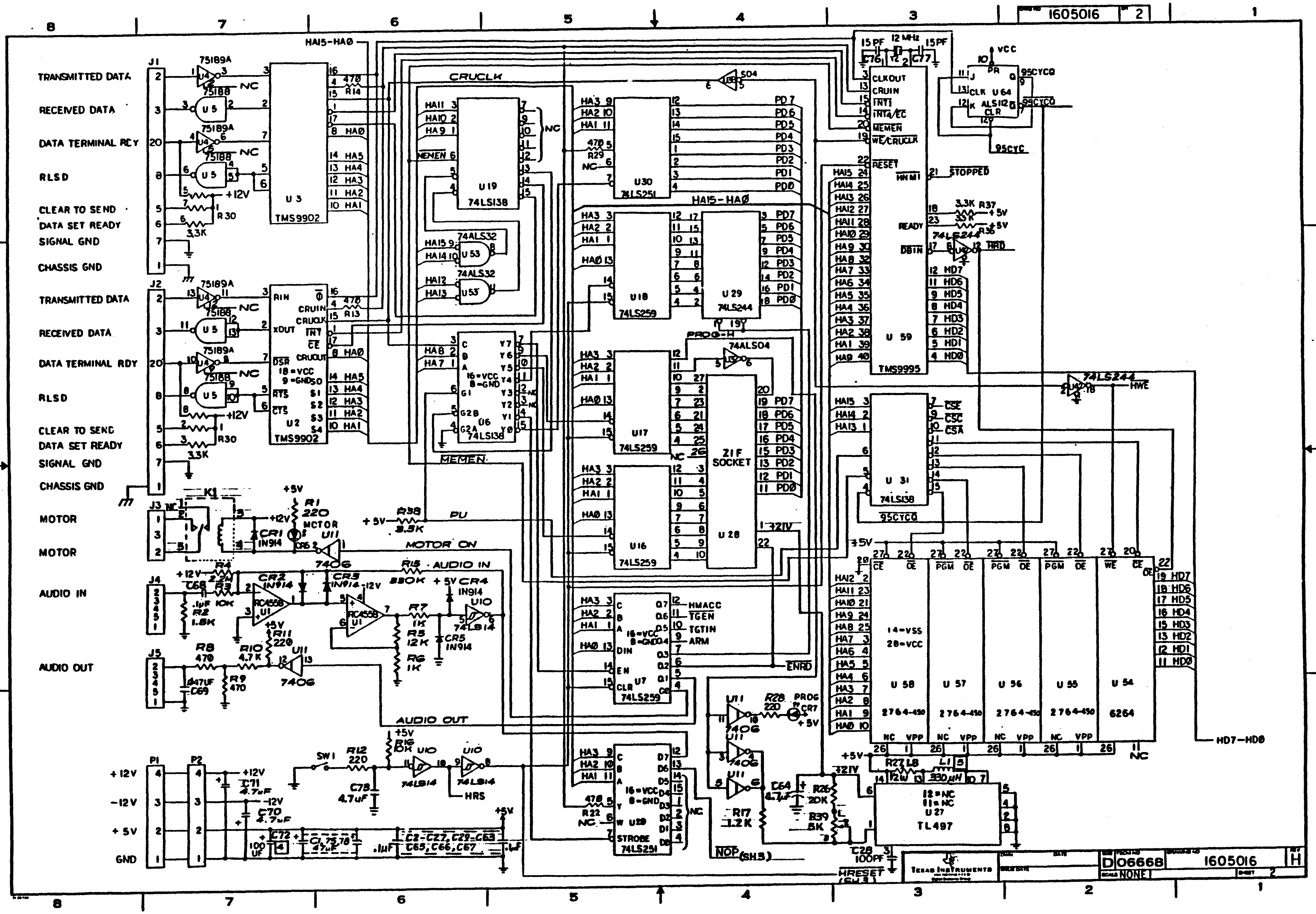
D 06668 1605016

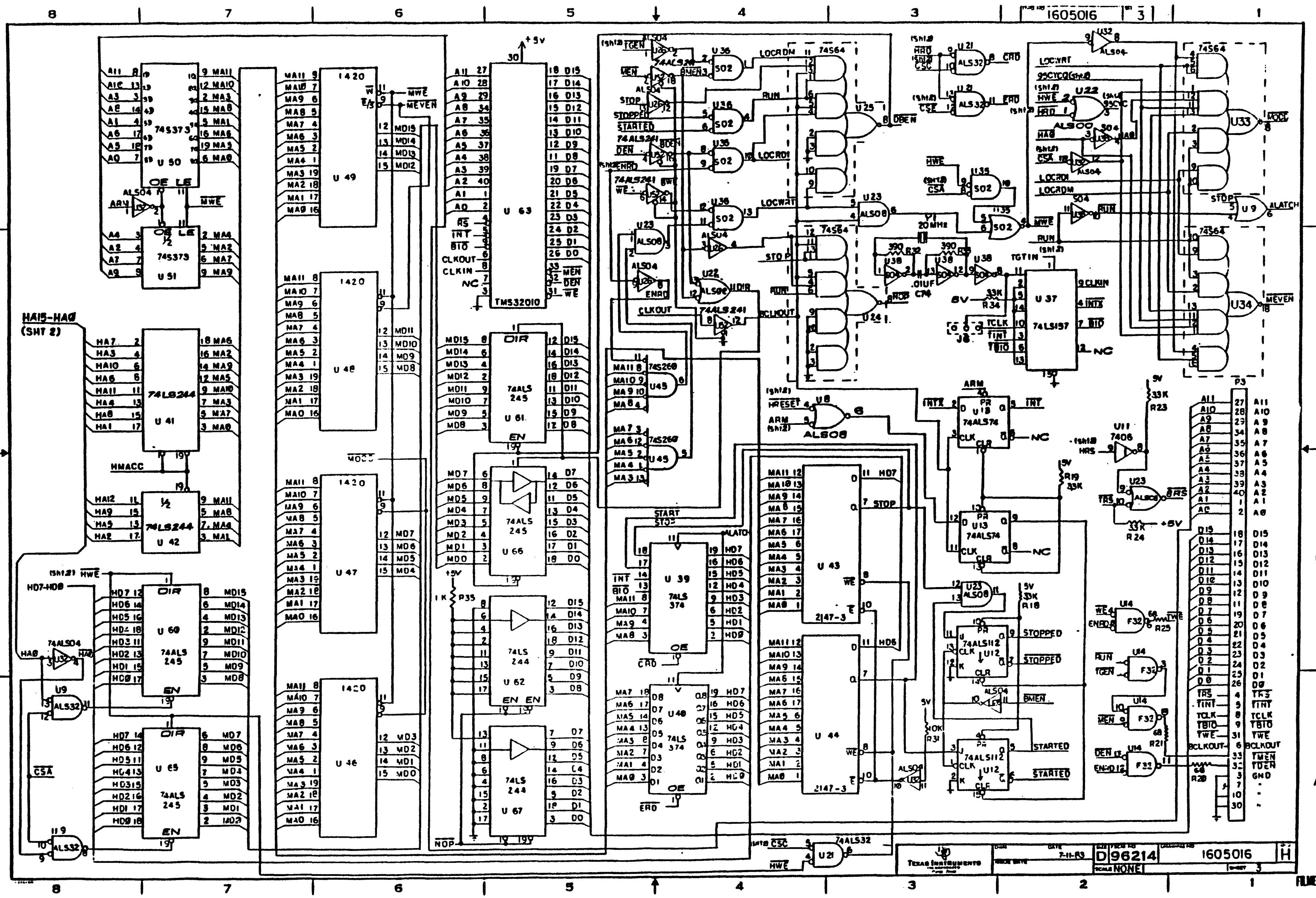
SCALE NONE | SHEET 1 OF 3

REV STATUS OF SHEETS: REV G G G; SH 1 2 3

PROCESS | CLASSIFICATION

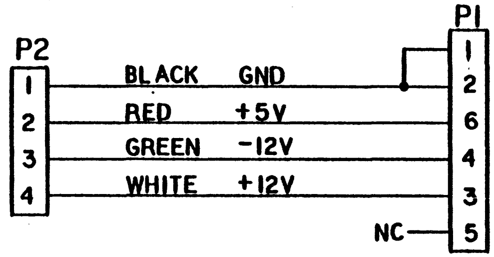
430


1605016
TRANSMITTED DATA
RECEIVED DATA
DATA TERMINAL RDY
RLSD
CLEAR TO SEND
DATA SET READY
SIGNAL GND
CHASSIS GND
MOTOR
AUDIO IN
AUDIO OUT
TMS9902
74LS138
74LS251
74LS259
74LS244
74ALS32
ZIF SOCKET
TMS9995
2764-450
6264
TL497
CRUCLK
MEMEN
MOTOR ON
AUDIO OUT
HA15-HAØ
HD7-HDØ
STOPPED
PROG-H
Texas Instruments
D 06668
1605016
H

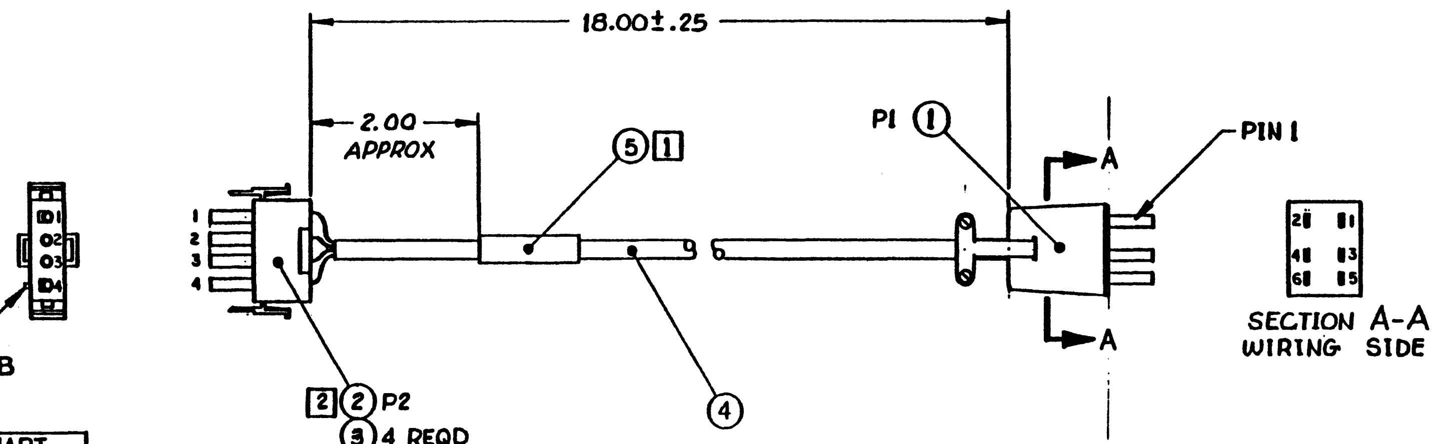

1605016
3
TMS32010
U 63
U 49
U 48
U 47
U 46
U 50
74S373
U 51
74S373
74LS244
U 41
74LS244
U 42
74ALS
245
U 60
U 65
U 61
U 66
74LS
244
U 62
U 67
U 39
74LS
374
U 40
U 43
2147-3
U 44
U 37
74LS157
74ALS74
U 13
74ALS112
U 12
74S64
U 25
U 24
U 33
U 34
U 9
ALATCH
MOCC
MEVEN
HA15-HAØ
(SHT 2)
HD7-HDØ
HMACC
20MHZ
P3
TEXAS INSTRUMENTS
D96214
1605016
H
NONE

NOTES, UNLESS OTHERWISE SPECIFIED:

1. MARK PER 2265070, LINE 4 TEXT "320 PWR"
2. DISREGARD VENDOR PIN NUMBER MARKING OF HOUSING (IT.2) AND BUILD AS SHOWN



WIRING DIAGRAM



CONVERSION CHART	
INCHES	mm
.010	0.25
.02	0.5
2.00	50.8
18.00±.25	457.2±6.4

SEQ NO	IDENT	F-SPEC	NO	ADDITIONAL	NOTES
1	SLDR	127-01	—	HAND SOLDER	
	PROCESS			CLASSIFICATION	

PROCESSES — FOR CORRELATION TO GOVT/IND SPECIFICATIONS, SEE TI DRAWING 729467

QTY	ITEM NO	PART OR IDENTIFYING NUMBER	NOMENCLATURE OR DESCRIPTION	PROCUREMENT SPECIFICATION	NOTES

PARTS LIST

NEXT ASSY	USED ON
1605029	8820(A03)

APPLICATION

UNLESS OTHERWISE SPECIFIED
- DIMENSIONS ARE IN INCHES
- TOLERANCES: ANGLES ±1°, 3 PLACE DECIMALS ±.010, 2 PLACE DECIMALS ±.02
- INTERPRET DRAWING PER MIL-D-1000
- REMOVE ALL BURRS AND SHARP EDGES
- CONCENTRICITY MACHINED DIAMETERS .010 FIM
- DIMENSIONAL LIMITS APPLY BEFORE PROCESSES
- PARENTHETICAL INFO FOR REF ONLY

HOLE TOLERANCE

DRN 9/15/83
CHK 9/28/83
ENGR 9/29/83
QA 9/29/83
CONTR 9/29/83
10-7-83

TEXAS INSTRUMENTS INCORPORATED Dallas, Texas

CABLE ASSEMBLY, POWER, TMS320 EVM

SIZE C FSCM NO 96214 DRAWING NO 1605020

SCALE 1/1 SHEET

REVISIONS

REV	DESCRIPTION	DATE	APPROVED

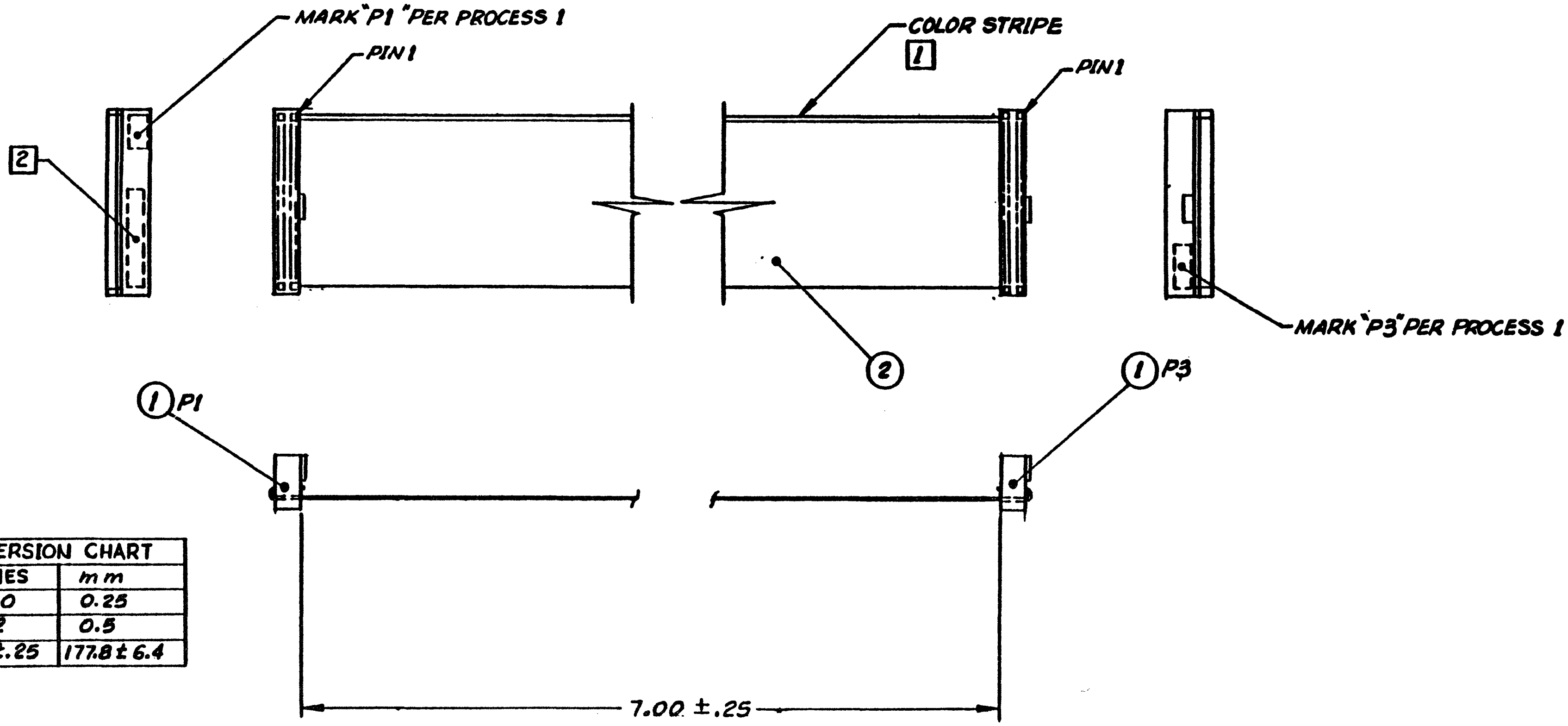
1605020

NOTES, UNLESS OTHERWISE SPECIFIED:

[1] ALIGN COLOR STRIPE ON CABLE (ITEM 2) WITH PIN 1 OF CONNECTORS P1, P3

[2] MARK APPROPRIATE PART NUMBER AND REV LETTER PER PROCESS 1

REV	DESCRIPTION	DATE	APPROVED



CONVERSION CHART	
INCHES	mm
.010	0.25
.02	0.5
7.00±.25	177.8±6.4

SEQ NO	IDENT	F-SPEC	NO	ADDITIONAL	NOTES
1	MARK	100-07	712	COLOR BLACK, TYPE 6	

PROCESS — CLASSIFICATION

PROCESSES — FOR CORRELATION TO GOVT/IND SPECIFICATIONS, SEE TI DRAWING 729467

-1 QTY	ITEM NO	PART OR IDENTIFYING NUMBER	NOMENCLATURE OR DESCRIPTION	PROCUREMENT SPECIFICATION	NOTES

PARTS LIST

NEXT ASSY	USED ON
1605029	8820(A03)

APPLICATION

UNLESS OTHERWISE SPECIFIED
- DIMENSIONS ARE IN INCHES
- TOLERANCES: ANGLES ±1°, 3 PLACE DECIMALS ±.010, 2 PLACE DECIMALS ±.03
- INTERPRET DRAWING PER MIL-D-1000
- REMOVE ALL BURRS AND SHARP EDGES
- CONCENTRICITY MACHINED DIAMETERS .010 FIM
- DIMENSIONAL LIMITS APPLY BEFORE PROCESSES
- PARENTHETICAL INFO FOR REF ONLY

HOLE TOLERANCE

DWN 9/14/83; 9/29/83; 9/29/83; QA 9/29/83; CONTR 9/29/83; RLSE 7-83

TEXAS INSTRUMENTS INCORPORATED Dallas, Texas

CABLE ASSY, 320 TARGET

SIZE C | FSCM NO 96214 | DRAWING NO 1605021

SCALE NONE | SHEET

1605021

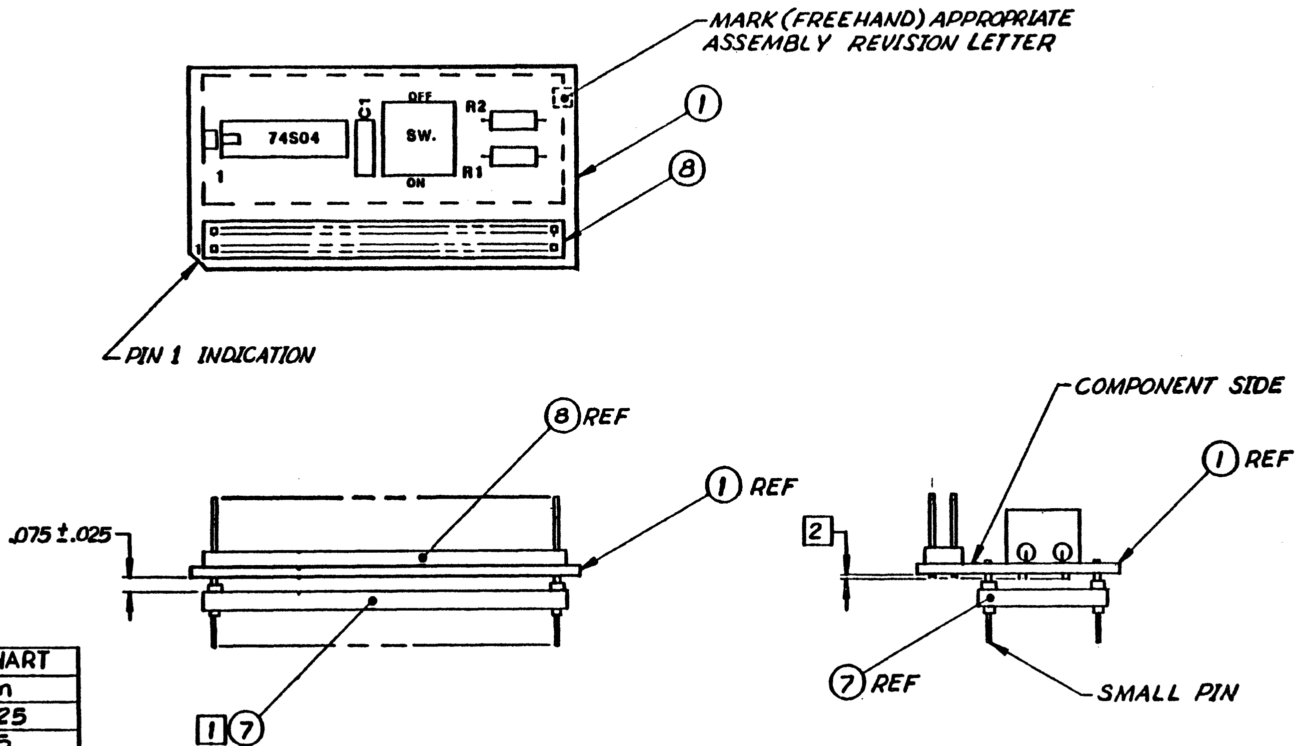
NOTES, UNLESS OTHERWISE SPECIFIED:

[1] INSTALL AFTER PROCESS 1

[2] TRIM LEADS OF ALL COMPONENTS MOUNTED ON COMPONENT SIDE TO .040 FROM CONDUCTOR SIDE PRIOR TO INSTALLING ADAPTER (IT. 7)

REVISION LEVEL CONTROL		
PWB 1605026	*	A
DIAG 1605028	*	*
ASSY 1605027	*	A

REVISIONS			
REV	DESCRIPTION	DATE	APPROVED
A	CN495021 Henson 10/26/83	10-26-83	



CONVERSION CHART	
INCHES	mm
.010	0.25
.02	0.5
.040	1.02
.075±.025	1.91±0.64

SEQ NO	IDENT	F-SPEC	NO	ADDITIONAL	NOTES
2	SLDR	124-02	–	HAND SOLDER	
1	SLDR	127-01	–	FLOW SOLDER	

PROCESS — CLASSIFICATION

PROCESSES — FOR CORRELATION TO GOVT/IND SPECIFICATIONS, SEE TI DRAWING 729467

-1 QTY	ITEM NO	PART OR IDENTIFYING NUMBER	NOMENCLATURE OR DESCRIPTION	PROCUREMENT SPECIFICATION	NOTES

PARTS LIST

UNLESS OTHERWISE SPECIFIED
- DIMENSIONS ARE IN INCHES
- TOLERANCES: ANGLES ±1°; 2 PLACE DECIMALS ±.010; 2 PLACE DECIMALS ±.02
- INTERPRET DRAWING PER MIL-D-1000
- REMOVE ALL BURRS AND SHARP EDGES
- CONCENTRICITY MACHINED DIAMETERS .010 FIM
- DIMENSIONAL LIMITS APPLY BEFORE PROCESSES
- PARENTHETICAL INFO FOR REF ONLY

HOLE TOLERANCE

NEXT ASSY	USED ON
1605029	8820(A03)

APPLICATION

TEXAS INSTRUMENTS INCORPORATED Dallas, Texas

TARGET BD. ASSY, 320EVM

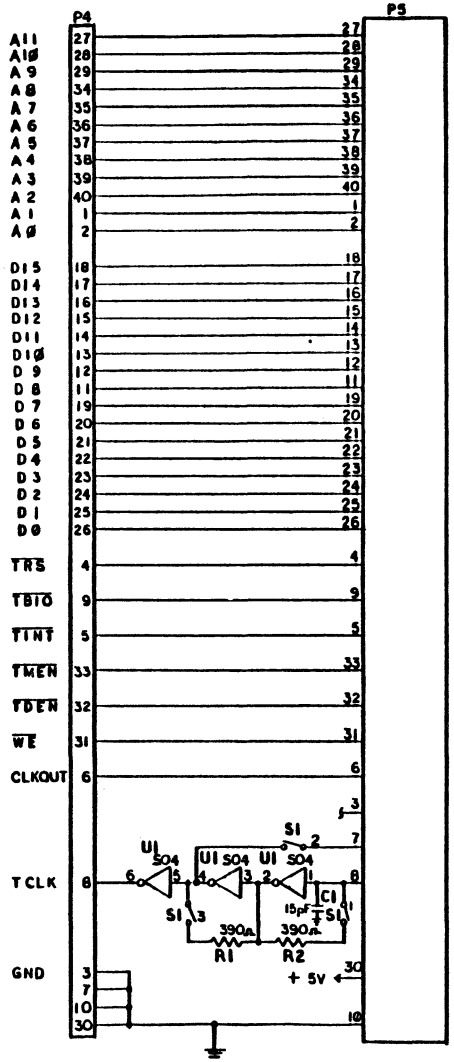
SIZE C | FSCM NO 96214 | DRAWING NO 1605027

SCALE 2/1 | SHEET

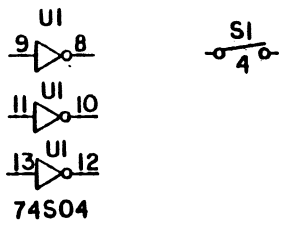
NOTES, UNLESS OTHERWISE SPECIFIED:

1. ALL RESISTORS ARE 1/4 W, 5%

REVISIONS			
REV	DESCRIPTION	DATE	APPROVED



SPARES



-1 QTY	ITEM NO	PART OR IDENTIFYING NUMBER	NOMENCLATURE OR DESCRIPTION	PROCUREMENT SPECIFICATION	NOTES
		PARTS LIST			

SEQ NO	IDENT	F-SPEC	NO	ADDITIONAL	NOTES
	PROCESS			CLASSIFICATION	

PROCESSES — FOR CORRELATION TO GOVT/IND SPECIFICATIONS, SEE TI DRAWING 729467

1605027	8820(A03)
NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED
- DIMENSIONS ARE IN INCHES
- TOLERANCES: ANGLES ±1°; 3 PLACE DECIMALS ± .010; 2 PLACE DECIMALS ± .02
- INTERPRET DRAWING PER MIL-D-1000
- REMOVE ALL BURRS AND SHARP EDGES
- CONCENTRICITY MACHINED DIAMETERS .010 FIM
- DIMENSIONAL LIMITS APPLY BEFORE PROCESSES
- PARENTHETICAL INFORMATION REF ONLY

HOLE TOLERANCE

DRN	DATE
DSGN	
CHKD	
APPD	
CONFIG	
RLSE	

TEXAS INSTRUMENTS INCORPORATED, Dallas, Texas

DIAGRAM, SCHEMATIC, TMS320 TARGET BOARD

SIZE	TECH NO	DRAWING NO
C	96214	1605028

SCALE —— SHEET

1605028

A-11

B. TMS32010 EVM COMMAND SUMMARY

B.1 INTRODUCTION

The tables in this appendix constitute a summary of all the commands presented in this manual. All are discussed in detail in the various chapters. Throughout the document, the following conventions have applied:

```
INPUT PORT:          IP
OUTPUT PORT:         OP
PORT NUMBERS:        1     EIA: Terminal
                     2     EIA: Uplink/Downlink/Printer
                     3     Audio Tape

KEYBOARD INTERRUPTS: May be  caused at  any time during any
                     display, dump, assembly listing, etc.:

                     <SP>  Stop/Start Display
                     <ESC> Abort Activity
                     <CR>  Reset
```

KEYBOARD ENTRY AIDS (Special Function Keys)

KEY/KEYS *	FUNCTION	
ESC	Delete Line/Create New Line	
RUB/DEL	Delete Last Character on the Line	
CNTL/N	Insert a Character	
CNTL/D	Delete a Character	
CNTL/F	Move Cursor to the Right	(———>)
CNTL/L	Move Cursor to the Right	
CNTL/P	Move Cursor to the Right	
CNTL/H	Move Cursor to the LEFT	(<———)
CNTL/	Redisplay Line	
CNTL/A	Cursor Home	
CNTL/	Cursor Home	
ESC/CNTL/R	Cursor Home	
CNTL/E	Move Cursor to End of Text	
CNTL/X	Delete to End of Line from Cursor	
CNTL/Y	Delete All Characters on Line	
CNTL/I	Tab Right (Tab)	
ESC/I	Tab Left (Back Tab)	

* In the case of multiple keys, press simultaneously.

TRANSPARENCY MODE COMMANDS

COMMAND	FUNCTION
COMM	Display/Modify Toggle Character
PASS	Transmit Toggle Character to Port 2

CONTROL CHARACTERS RESERVED FOR EVM USE

KEYS	FUNCTION
CNTL/[	Escape
CNTL/M	Carriage Return
CNTL/J	Cursor Down
CNTL/K	Cursor Up
CNTL/V	Cursor Down
CNTL/Z	Cursor Up

DISPLAY/MODIFY REGISTER SET COMMANDS

COMMAND	MNEMONIC
Display/Modify Accumulator	ACC
Display/Modify T Register	TREG
Display/Modify P Register	PREG
Display/Modify Auxiliary Register 0	AR0
Display/Modify Auxiliary Register 1	AR1
Display/Modify Program Counter	PC
Display/Modify Overflow Flag	OV
Display/Modify Overflow Mode	OVM
Display/Modify Data Page Pointer	DP
Display/Modify Auxiliary Register Pointer	ARP
Display/Modify Stack Locations	STACK
SUBCOMMAND	
This Menu	M
Redisplay Register	R
Redisplay Opposite Format	,x
Next Entry	<SP>, <LF>, CNTL/J, CNTL/V or Cursor Down
Previous Entry	CNTL/K,Z or Cursor Up

DISPLAY/MODIFY MONITOR COMMANDS

COMMAND	MNEMONIC
Display Monitor Commands Menu	HELP
Display Keyboard Entry Aids	KHELP
Display Monitor Error Messages	EHELP
Display Monitor Menu	MENU
Display/Modify Clock/Memory Source	INIT
Display/Modify Tabs	TABS
16-Bit Unsigned Hex-Decimal Conversion	UD16
32-Bit Unsigned Hex-Decimal Conversion	UD32
16-Bit Signed Hex-Decimal Conversion	SD16
32-Bit Signed Hex-Decimal Conversion	SD32
16-Bit Decimal-Hex Conversion	HX16
32-Bit Decimal-Hex conversion	HX32
Display Scaled 16-Bit Decimal Number	SCALE
Save/Show Machine State	SMS
Load Machine State	LMS
Save Program Memory	SPM
Load Program Memory	LPM
Display Program Memory	DPM
Display Data Memory	DDM
Display TMS32010 Register Set	STATE
Clear TMS32010 Register Set	CLEAR
Display ACC/T Reg/P Reg In Hex	HATP

TRACE COMMANDS

COMMAND	MNENOMIC
Set Single-Step Trace Line Locations	ST
Clear One Or All Trace Locations	CT
Display Trace Locations	DT
SUBCOMMAND	
This Menu	,M
Clear Field	,C
Redisplay Field	,R
Next Entry	<SP>,<LF>, CNTL/J, CNTL/V, or Cursor Down
Previous Entry	CNTL/K,Z or Cursor Up
Quit	<CR>

DISPLAY/MODIFY MEMORY COMMANDS

COMMAND	MNEMONIC
Display/Modify Data Memory	MDM
Find Word in Data Memory	FWDM
Find Byte in Data Memory	FBDM
Display/Modify Program Memory	MPM
Find Word in Program Memory	FWPM
Find Byte in Program Memory	FBPM
SUBCOMMAND	
This Menu	,M
Quit 'FIND' Byte/Word in Program/Data Memory	,Q
Change Display Format	,F
Redisplay Current Value	,R
Display Current Value in Hexadecimal	,H
Display Current Value in Signed Decimal	,S
Display Current Value in Unsigned Decimal	,D
Display Current Value in Binary	,B
Enter New 'MPM'/'MDM' Address	,A =
Next Entry	<SP>,<LF>,CNTL/J, CNTL/V or Cursor Down
Previous Entry	CNTL/K,Z or Cursor Up
Quit MXM/'FIND' Next Entry	<CR>

DISPLAY MENU COMMANDS

COMMAND	MNEMONIC
Display All the Menu Display Menus	/HELP
Display the Monitor Menu	/MON
Display the PROM Utility Menu	/PROM
Display the Text Editor Commands Menu	/EDIT
Display/Modify Memory Commands Menu	/MM
Display Register Commands Menu	/REGS
Display Set Breakpoint Subcommands Menu	/SB
Display Set Trace Subcommands Menu	/ST
Display Baud Rate Subcommands Menu	/BAUD
Display Single-Step Subcommands Menu	/SS

SINGLE-STEP COMMANDS

COMMAND	MNEMONIC
Fill Data Memory	FDM
Fill Program Memory	FPM
Fill Program Memory with NOPs	NOP
Move Program Memory	MOVE
Execute User Program with Breakpoints	EX
Execute User Program without Breakpoints	RUN
Single-Step User Program	SS
SUBCOMMAND	
This Menu	M
Change Display Type	Tx
Change Display Format	F
Enter Step Count	C
Display Program Counter	P
PC Display Range	R
List to Port 1	1
List to Port 2	2
Execute One Single-Step	<SP>
Return to Monitor	<CR>

BREAKPOINT COMMANDS

COMMAND	MNEMONIC
Set Breakpoints	SB
Clear One Or All Breakpoints	CB
Display Breakpoints	DB
Set Event Count For One Breakpoint	EC
SUBCOMMAND	
This Menu	,M
Clear Breakpoint	,C
Redisplay Breakpoint	,R
Display Event Counter	,E
Next Entry	<SP>,<LF>, CNTL/J, CMT/V or Cursor Down
Previous Entry	CNTL/K,Z or Cursor Up <CR>

DISPLAY/MODIFY BAUD RATE COMMANDS

COMMANDS	MNEMONIC
Display/Modify Port 1 Baud Rate	BAUD1
Display/Modify Port 2 Baud Rate	BAUD2
SUBCOMMANDS	
This Menu	,M
Redisplay Baud Rate	,R

NOTE: Allowable baud rates will be displayed at the bottom of the menu as follows:
Baud Rates: 110, 300, 600, 1200, 2400, 4800, 9600, 19200

MISCELLANEOUS MONITOR COMMANDS

COMMAND	MNEMONIC
Print Formfeed Character To EIA Port	FF
Display Files Stored On Audio Cassette	DIR
Enable Audio Cassette Motor	MO
Display Assembler Label Table	TABLE

ASSEMBLER DIRECTIVES

MNEMONIC	DEFINITION
AORG	Absolute Origin
BSS	Block Starting with Symbol
BES	Block Ending with Symbol
END	End Program
EQU	Define Assembly-Time Constant Directive
DATA	Initialize Word
TEXT	Initialize Text Editor
LIST	List Source
UNL	No Source List
SYMT	List Label Table
PAGE	Eject Page

PROM UTILITY COMMANDS

COMMAND	MNEMONIC
Compare EPROM to Memory	COMP
Display PROM Utility Menu	MENU
Program EPROM from Memory	PROG
Execute the PROM Utility	PROM
Quit PROM Utility	QUIT
Read EPROM to RAM	READ
Verify EPROM Erased	VRFY

EVM TEXT EDITOR COMMANDS

DESCRIPTION	MNEMONIC
Display Editor Commands Menu	HELP
Display Keyboard Entry Aids Menu	KHELP
Display/Modify Tabs	TABS
Clear Text and Initialize Editor	ZERO
Display Free RAM Available	MEMORY
Find String (8 characters maximum)	FIND
Quit Editor and Dump File	QUIT<output port>
Abort Current Activity	<ESC>
Autoincrement Line Number Mode	<line#>AUTO
Resequence Line Numbers to EOF	RESEQUENCE <INCREMENT>
List Line(s)	<line#>LIST<# of lines> Def = 1
Change Line Number	<line#>CHANGE<new line #>
Delete Line Number	<line#><CR>
Edit Line	<line#>EDIT
Save Line/Edit Next Line	CNTL/J, CNTL/V, line feed, or Cursor Down
Save Line/Edit Previous Line	CNTL/K, CNTL/Z, or Cursor UP
Enter Line	<line#><SP>
Duplicate Line	<line#>DUPLICATE<line#>

SYSTEM ACCESS COMMANDS

COMMAND	MNEMONIC	EQUIVALENT
Display/Modify Program Memory	$MPM	MPM
Display Program Memory	$DPM	DPM
Fill Program Memory	$FPM	FPM
Find Byte in Program Memory	$FBPM	FBPM
Find Word in Program Memory	$FWPM	FWPM
Move Memory	$MOVE	MOVE
UNIQUE COMMANDS		
Display/Modify Record Length	$DRL (Default = 182)	
Monitor Expansion Command	$USER	
Display Operating System Revision Level	$REV	
Perform a "Power Cycle" Reset	$BOOT	

March 1985
Revision A
1603479-9701
Printed in U.S.A.

TEXAS INSTRUMENTS
Creating useful products
and services for you.

SPRU005A