AMD Test Interface Port Board

User's Manual

Order #22505A

AMD

Test Interface Port Board User’s Manual

IF YOU HAVE QUESTIONS, WE'RE HERE TO HELP YOU.

The AMD customer service network includes U.S. offices, international offices, and a customer training center. Expert technical assistance is available from the AMD worldwide staff of field application engineers and factory support staff to answer E86™ family hardware and software development questions.

Frequently accessed numbers are listed below. Additional contact information is listed on the back of this manual. AMD's WWW site lists the latest phone numbers.

Technical Support

Answers to technical questions are available online, through e-mail, and by telephone.

Go to AMD's home page at **www.amd.com** and follow the Service link for the latest AMD technical support phone numbers, software, and Frequently Asked Questions.

For technical support questions on all E86 products, send e-mail to **epd.support@amd.com** (in the US and Canada) or **euro.tech@amd.com** (in Europe and the UK).

You can also call the AMD Corporate Applications Hotline at:

(800) 222-9323	Toll-free for U.S. and Canada
44-(0) 1276-803-299	U.K. and Europe hotline

WWW Support

For specific information on E86 products, access the AMD home page at **www.amd.com** and follow the Embedded Processors link. These pages provide information on upcoming product releases, overviews of existing products, information on product support and tools, and a list of technical documentation. Support tools include online benchmarking tools and CodeKit software—tested source code example applications. Many of the technical documents are available online in PDF form.

Questions, requests, and input concerning AMD's WWW pages can be sent via e-mail to **webmaster@amd.com**.

Documentation and Literature Support

Data books, user's manuals, data sheets, application notes, and product CDs are free with a simple phone call. Internationally, contact your local AMD sales office for product literature.

To order literature, call:

(800) 222-9323	Toll-free for U.S. and Canada
(512) 602-5651	Direct dial worldwide
(512) 602-7639	Fax

Third-Party Support

AMD FusionE86^{SM} partners provide an array of products designed to meet critical time-to-market needs. Products and solutions available include emulators, hardware and software debuggers, board-level products, and software development tools, among others. The WWW site and the *E86™ Family Products Development Tools CD*, order #21058, describe these solutions. In addition, mature development tools and applications for the x86 platform are widely available in the general marketplace.

Contents

About the Test Interface Port Board

Chapter 1

Getting Started

Chapter 2

System Features and Components

Appendix A

MACH® Device Equations

Index

List of Figures

List of Tables

About the Test Interface Port Board

The Test Interface Port (TIP) board was developed as a software debug and development board. For many reasons, it is often impractical to put extra peripherals on a target system just for development and software debugging. Serial ports are a good example. Often, adding extra peripherals makes development more difficult because the target peripherals are used by the application and cannot be used for software development and debugging. However, a target containing a small, 60-pin connector can host the TIP, a board rich in peripherals for development and software debugging.

The TIP board is a set of peripherals contained in one convenient location for software debugging, diagnostics, evaluation, and reference designs. The TIP board is designed to make it easy and convenient for the software to communicate the status with an engineer, tester, or even an end user. The TIP board is designed to support flexible, present-day applications and is used with a *target* board containing a microcontroller that can connect to the TIP board through the main interface connector (60-wire ribbon cable).

TIP Board Features

The TIP board provides the following features:

- Two 16550 RS-232 serial ports (9-pin, DCE).
- One PC-compatible parallel port.
- A 10BaseT Ethernet controller port that can be run in interrupt-driven mode.
- A 2 x 20 character ASCII-decoded display.
- An 8-segment (32-bit) hexadecimal display.
- Eight discrete LEDs (green).
- Three HP logic analyzer debug headers for easy address, data, and control signal debug access.

- Eight discrete TTL/CMOS outputs (can connect directly to a logic analyzer).
- Eight inputs connected to a DIP switch and header.
- A reset button that resets the target.
- An 8-bit DIP Flash memory, socketed to allow for upgrading.
- A jumper block (JP1) for selecting between internal (TIP board) or external (host board) Flash memory.
- A flexible interface for simple and inexpensive connection to a target board.
- An interrupt button.
- Five interrupt sources on the board: two serial ports, the parallel port, the Ethernet controller port, and the user-interrupt button. All these interrupts go to the main interface connector separately. In addition, all these interrupts are logic ORed into a single signal that also goes to the connector.
- A macro array CMOS high-density/high-performance (MACH®) device that provides individual chip selects for each on-board peripheral device (except the Ethernet controller port).
- Support for 16-bit addressing or 8-bit addressing, selected with the SW3 switch.
- Five programmable registers for controlling the TIP board's operation.

Documentation

The *AMD Test Interface Port Board User's Manual*, order #22505, provides information on the system and board features, functionality, and interfaces. Additional information can be found in the documentation listed on page xiii.

About This Manual

Chapter 1, "Getting Started" describes the power supply requirements and the main interface connector configuration for the TIP board. This chapter also explains the proper procedure for powering up the TIP board.

Chapter 2, "System Features and Components" describes in detail the peripherals, memory, main interface connector signals and pin out, and interrupts of the TIP board, including the jumper settings and programmable registers used to control the operation of the TIP board.

Appendix A, "MACH® Device Equations" contains a current listing of the MACH device program code.

Suggested Reference Material

The following AMD documentation may be of interest to the TIP board user.

- *AMD Am79C961 Ethernet Controller* specifications, order #18183

For current application notes and technical bulletins, see our World Wide Web page at **www.amd.com**.

The following non-AMD documentation may also be of interest to the TIP board user.

- Hitachi's *HD44780U (LCD-II) Dot Matrix Liquid Crystal Display Controller/ Driver* data sheet. For more information, refer to the Hitachi specification by accessing the web site **http://www.hitachi.com** and searching on HD44780U.
- Texas Instruments' *TL16C552 Dual Asynchronous Communications Element With FIFO* data sheet. For more information, refer to the Texas Instruments TL16C552 specification by accessing the web site **http://www.ti.com** and searching on TL16C552.

Documentation Conventions

The *Test Interface Port Board User's Manual* uses the notational conventions shown in Table 0-1 (unless otherwise noted).

Table 0-1. Notational Conventions

Symbol	Usage
Boldface	Indicates that characters must be entered exactly as shown, except that the alphabetic case is only significant when indicated.
Italic	Indicates a descriptive term to be replaced with a user-specified term.
`Typewriter face`	Indicates computer text input or output in an example or listing.
`[]`	Encloses an optional parameter. To include the information described within the brackets, type only the parameter, not the brackets themselves.
$\overline{\texttt{SIGNAL}}$	An overbar over a signal name indicates that it is active low.
`SIGNAL#`	A pound sign after a signal name is used to indicate an active low in schematics.

Chapter 1

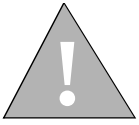
Getting Started

The Test Interface Port (TIP) board comes prepared for immediate use. Connect the supplied flexible ribbon cable to the TIP board and to the host's interface connector, then follow the correct power-up procedure (see "Board Power" on page 1-1), and the TIP board is ready to use. You can program the on-board Flash memory with board diagnostics, a monitor program, or application software.

NOTE: To use any application that may be programmed on the TIP board's Flash memory, set the Flash Select jumper to the INT position.

Board Power

The TIP board requires an external 5-V DC power supply with a 2.0-A current rating. Although the external power supply provides the 5-V power for the board, the board still does not power up (even with the external power supply connected) unless a host board is also properly connected. This condition is caused by the two power sources being routed through two MOSFETs that are managed by a comparator. The comparator gets its power from the host VCC. This MOSFET/comparator routing prevents any power to the board without the presence of both the external and host power sources.



CAUTION: Do not provide power to the TIP board through a host board without first connecting an external power source to the TIP board. Failure to correctly follow this procedure causes back powering of the FETs which could damage the TIP board.

Use the following procedure to power up the TIP board:

1. Be sure the host board is *not* powered up.

2. Attach the interface ribbon cable connector between the TIP board and the host board. On the TIP board, the ribbon cable connector attaches to connector P1, shown in Figure 1-1
3. Connect the external 5-V DC power supply to the TIP board's barrel connector (P2), shown in Figure 1-1.

CAUTION: Failure to follow this procedure can result in irregular results or damage to the TIP board.

4. Perform the normal power-up procedures on the host board.

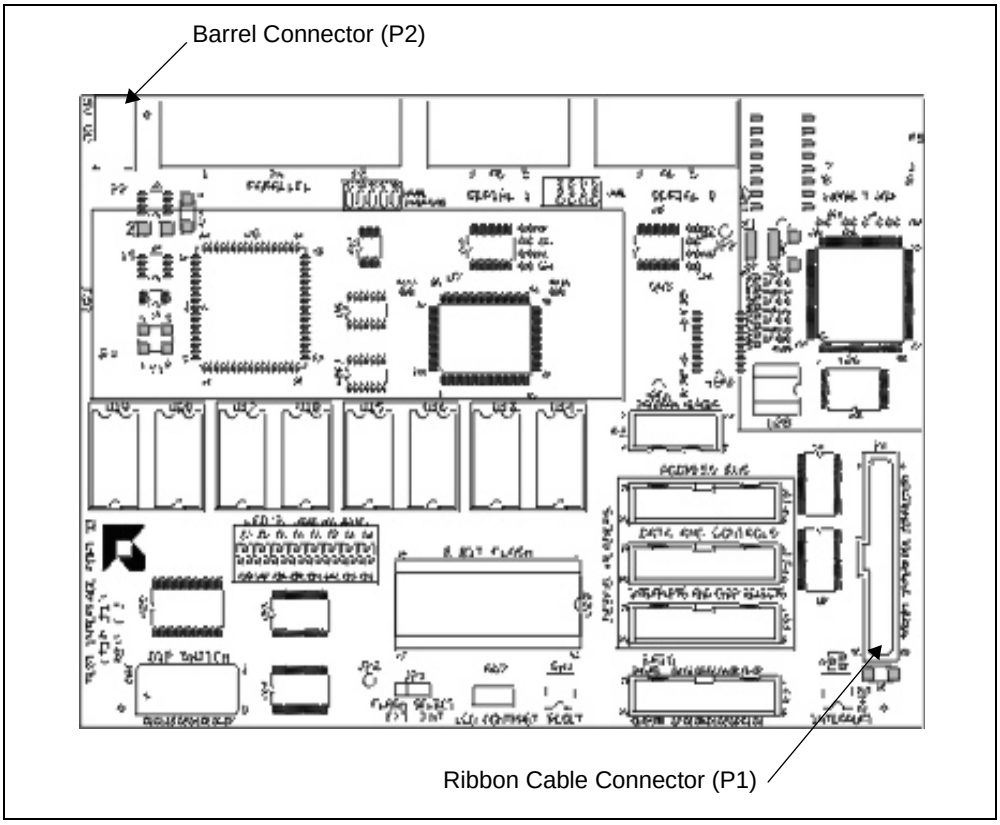


Figure 1-1. Barrel (P2) and Ribbon Cable (P1) Connectors Locations

Interface Cable

The TIP board requires a 60-wire ribbon cable assembly to connect to the host interface. Figure 1-2 illustrates the interface cable assembly. Figure 1-3 on page 1-4 illustrates the orientation of the host interface connector and the connector on the interface cable assembly that connects to the host. For a description of the interface signals, see “Main Interface Connector” on page 1-5.

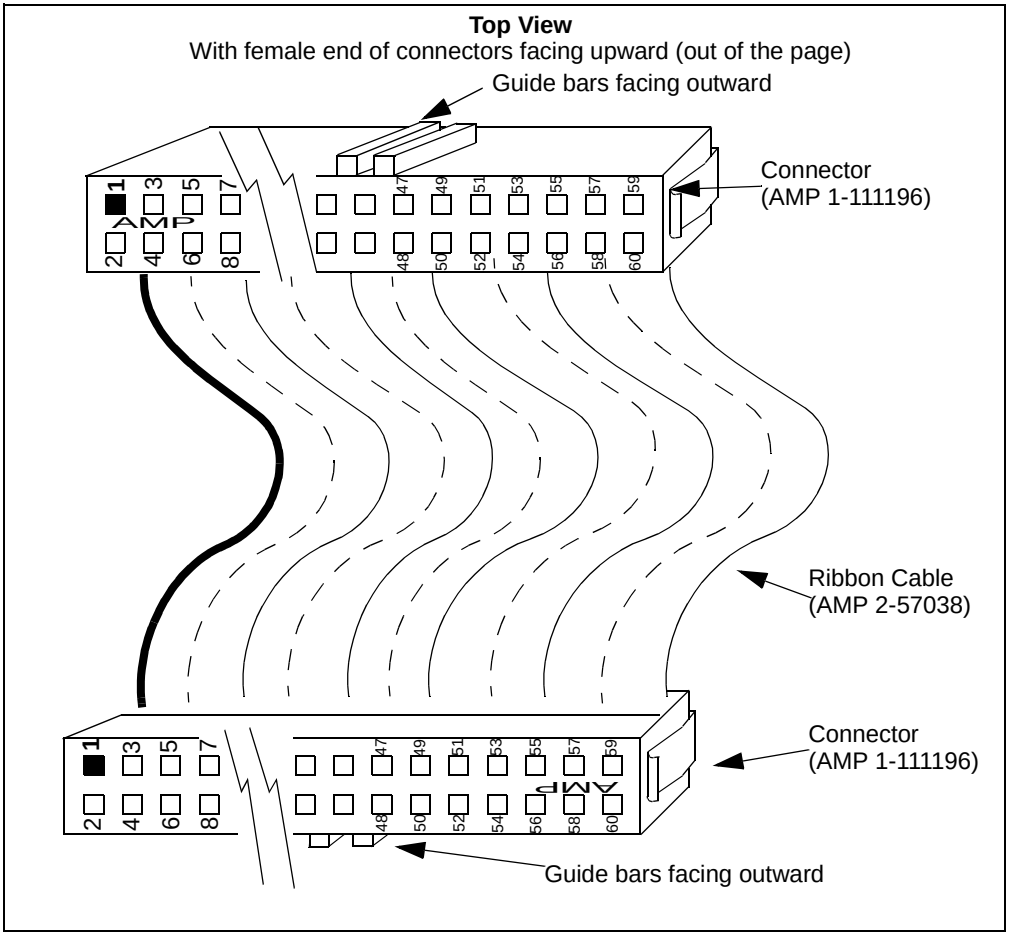


Figure 1-2. TIP Board Ribbon Cable

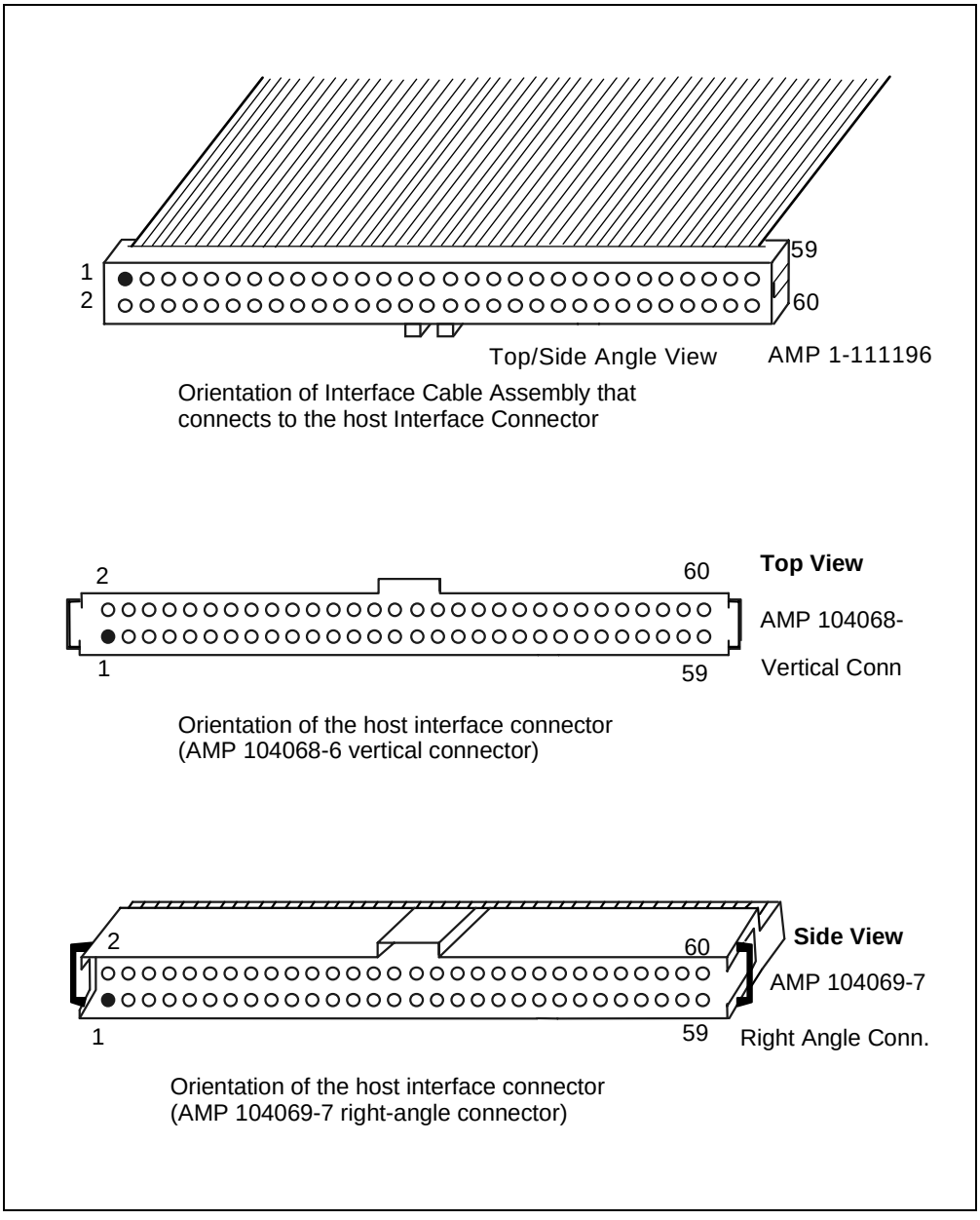


Figure 1-3. Ribbon Cable Connector Orientation

Main Interface Connector

The 60-wire ribbon cable connects to the TIP board on the main interface connector (P1). Table 1-1 describes each of the 60 signals on the connector. Figure 1-4 on page 1-9 illustrates the pinout of P1.

Table 1-1. Interface Connector Signal Descriptions

Pin No.	Signal Name	Input/ Output	Description
1–10, 13–22	TA0–TA19	I	**Address Lines:** Receive the physical memory latched address for the Flash memory or Ethernet controller and the physical I/O latched address for all other TIP board peripherals through the MACH device.
11, 12, 23, 24	GND	—	**Ground Pins**
25–32	TD0–TD7	I/O	**Data Bus:** Input and output data during respective read or write cycle.
33	–	–	Pin is blank.
34	–	–	Pin is blank.
35	$\overline{\text{TIPSEL}}$	O	**TIP Select #:** The hardware method of identifying that the TIP board is physically connected to the host. The TIP board has a 10-KΩ pullup resistor on $\overline{\text{TIPSEL}}$. If the host board wants to use $\overline{\text{TIPSEL}}$ to indicate that the TIP board is connected, then the host board should have a weak pulldown resistor (100 KΩ) on the signal connected to $\overline{\text{TIPSEL}}$. Then, when the TIP board is connected to the host board, $\overline{\text{TIPSEL}}$ signal goes Low, indicating that the TIP board is properly connected to the host.
36	–	–	Pin is blank.

Table 1-1. Interface Connector Signal Descriptions (Continued)

Pin No.	Signal Name	Input/ Output	Description
37	TIPSEL	O	**TIP Select:** The hardware method of indentifying that the TIP board is physically connected to the host. The TIP board has a 1-KΩ pulldown resistor on TIPSEL. If the host board wants to use TIPSEL to indicate that the TIP board is connected, then the host board should have a weak pullup resistor (10 KΩ) on the signal connected to TIPSEL. When the TIP board is connected to the host board, TIPSEL signal goes High, indicating that the TIP board is properly connected to the host.
38	TAEN	I	**Address Enable:** For ISA systems: When asserted High, TAEN enables DMACs on the buses and prevents I/O devices from responding. For the Am186™CC CDP: a chip select that, when asserted Low, allows for communication with the Ethernet controller.
39	–	–	Pin is blank.
40	$\overline{\text{TRD}}$	I	**Read Strobe:** Indicates to the system that the host microcontroller is performing a memory or I/O read cycle.
41	–	–	Pin is blank.
42	$\overline{\text{TWR}}$	I	**Write Strobe:** Indicates to the system that the host microcontroller is performing a memory or I/O write cycle.
43	ENETIRQ	O	**Ethernet Interrupt Request:** Indicates that one of several status flags is set (consult Ethernet controller specification for details).

Table 1-1. Interface Connector Signal Descriptions (Continued)

Pin No.	Signal Name	Input/ Output	Description
44	$\overline{TS2}$	I	**Bus Cycle Status:** Used by Am186 family of microcontrollers as a logical memory or I/O indicator.
45	PARINT	O	**Printer Port Interrupt:** Dedicated parallel port interrupt signal.
46	–	–	Pin is blank.
47	SERINT1	O	**Serial Port 1 Interrupt:** Dedicated serial port 1 interrupt signal.
48	MAIN_IRQ	O	**Main Interrupt Line:** A shared interrupt line. MAIN_IRQ is asserted if any of the five interrupts on the TIP board are asserted.
49	SERINT0	O	**Serial Port 0 Interrupt:** Dedicated serial port 0 interrupt signal.
50	$\overline{HRESET}$	O	**Host Reset:** Toggled by the TIP board reset button, which then indicates to the host microcontroller to perform a system-wide hardware reset.
51	IOCHRDY	O	**I/O Channel Ready:** Indication by the Ethernet controller that valid data exists on the data bus for reads and that data has been latched for writes.
52	TRESET	I	**TIP Reset:** Generated by the host microcontroller when performing a hardware reset. TRESET affects the TIP board's Ethernet controller, and parallel and serial ports.
53	–	–	Pin is blank.
54	$\overline{FLASHRD}$	I	**Flash Read Enable:** Output enable.

Table 1-1. Interface Connector Signal Descriptions (Continued)

Pin No.	Signal Name	Input/ Output	Description
55	SEL186	I	**Select Am186:** Identifies that a Am186 microcontroller board is connected and is hosting the TIP board. This helps to determine when the Ethernet controller should respond to read and write cycles. There is a 1-KΩ pulldown resistor connected to this pin for when a Am186 board is not connected. SEL186 should be driven High by any Am186 host boards connecting to the TIP board.
56	$\overline{\text{FLASHWR}}$	I	**Flash Write Enable:** Write enable.
57, 59	VCC	I	**Host Power:** +5 V DC provides power to the comparator, which in turn provides power to the MOSFET driver, which drives the FETs, allowing the external power supply to provide +5 V DC to the board.
58	$\overline{\text{FLASHCS}}$	—	**Flash Chip Select:** Flash memory chip enable.
60	$\overline{\text{EXTFLHCS}}$	I	**External Flash Chip Select:** An optional external Flash memory chip select jumper. The host boards can take advantage of $\overline{\text{EXTFLHCS}}$ by making the TIP board's Flash memory select jumper the primary Flash memory select component of the two boards (theTIP board and the host board). This minimizes components on the host board.

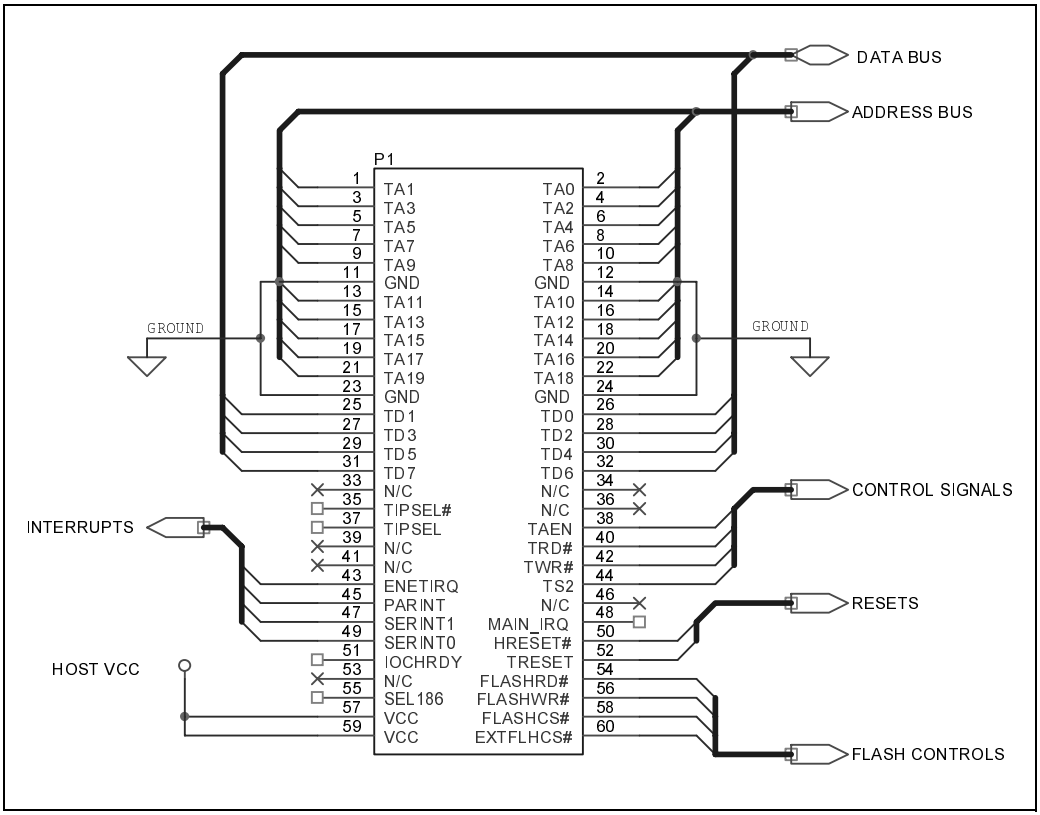


Figure 1-4. Main Interface Connector Pinout

Chapter 2

System Features and Components

The Test Interface Port (TIP) board is a set of peripherals at one convenient location for software debugging, diagnostics, evaluation, and reference designs. The TIP board is designed to make it easy and convenient for the software to indicate to an engineer, tester, or an end user the tasks that are being performed. The TIP board is designed to support flexible, present-day applications, and is intended to be used with any host board containing a microcontroller that can connect to the TIP board through the small, 60-pin, main interface connector. The TIP board supports either I/O-mapped or memory-mapped operation and can be used in either mode.

The TIP board requires minimal software initialization for basic operation. This chapter provides detailed information about the features and components of the TIP board. The following sections explain the operation of the board in detail, including jumper settings, switch settings, and programmable registers:

Layout and Placement

The TIP board is laid out for convenient connection to the host board and to the various external devices. It has connectors for DC power, parallel port, RJ-45, and the serial ports along one side of the board. On a side adjacent to the port connector side are the main interface connector, the general-purpose input/output header, the debug headers, and the MACH device programming header. The LCD, hex displays, and LEDs are arranged close together in the middle of the board. Refer to Figure 2-1 on page 2-3 for layout and component placement.

Figure 2-2 on page 2-4 is a block diagram of the TIP board showing the connections between the various peripherals on the board.

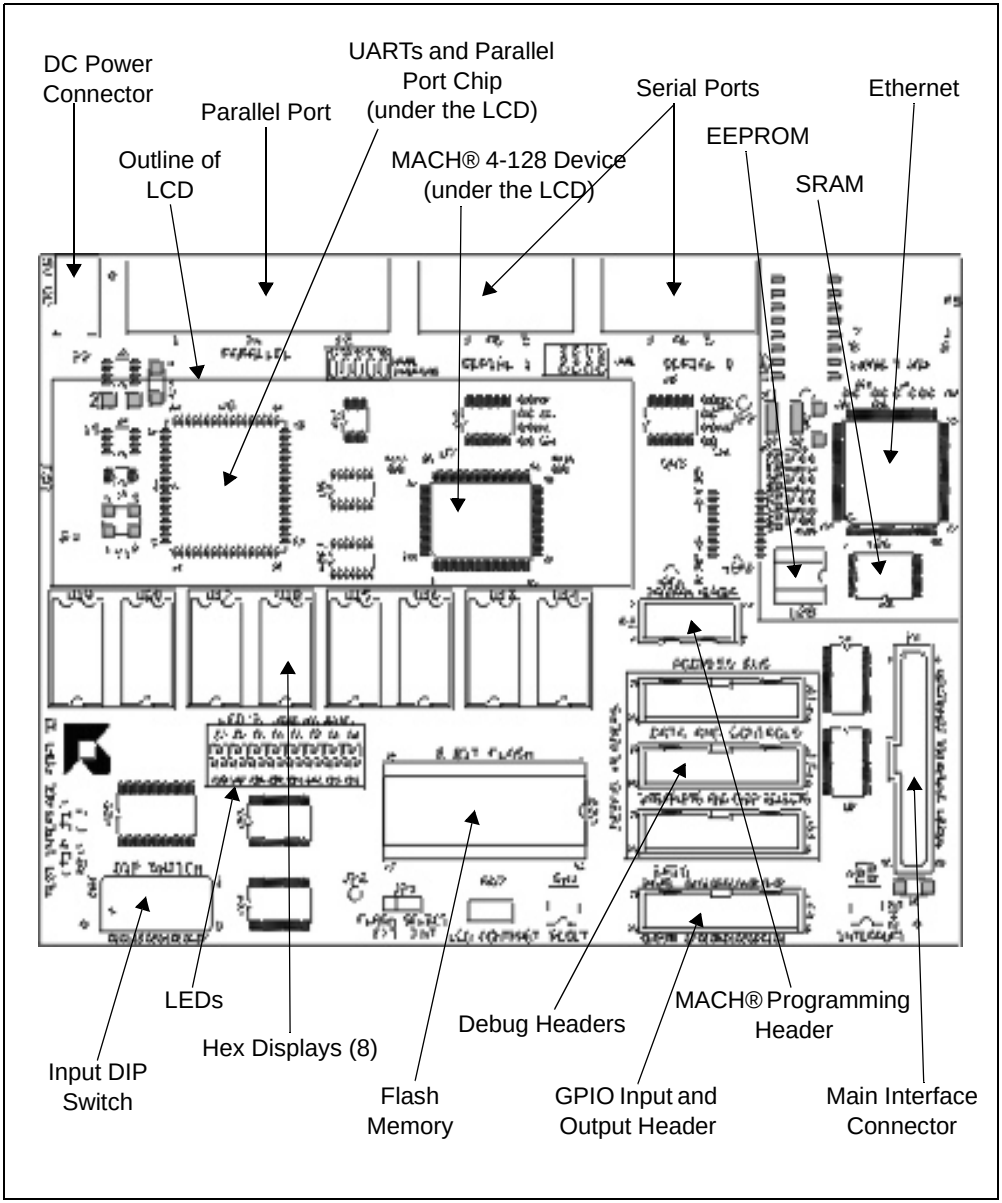


Figure 2-1. TIP Circuit Board Layout

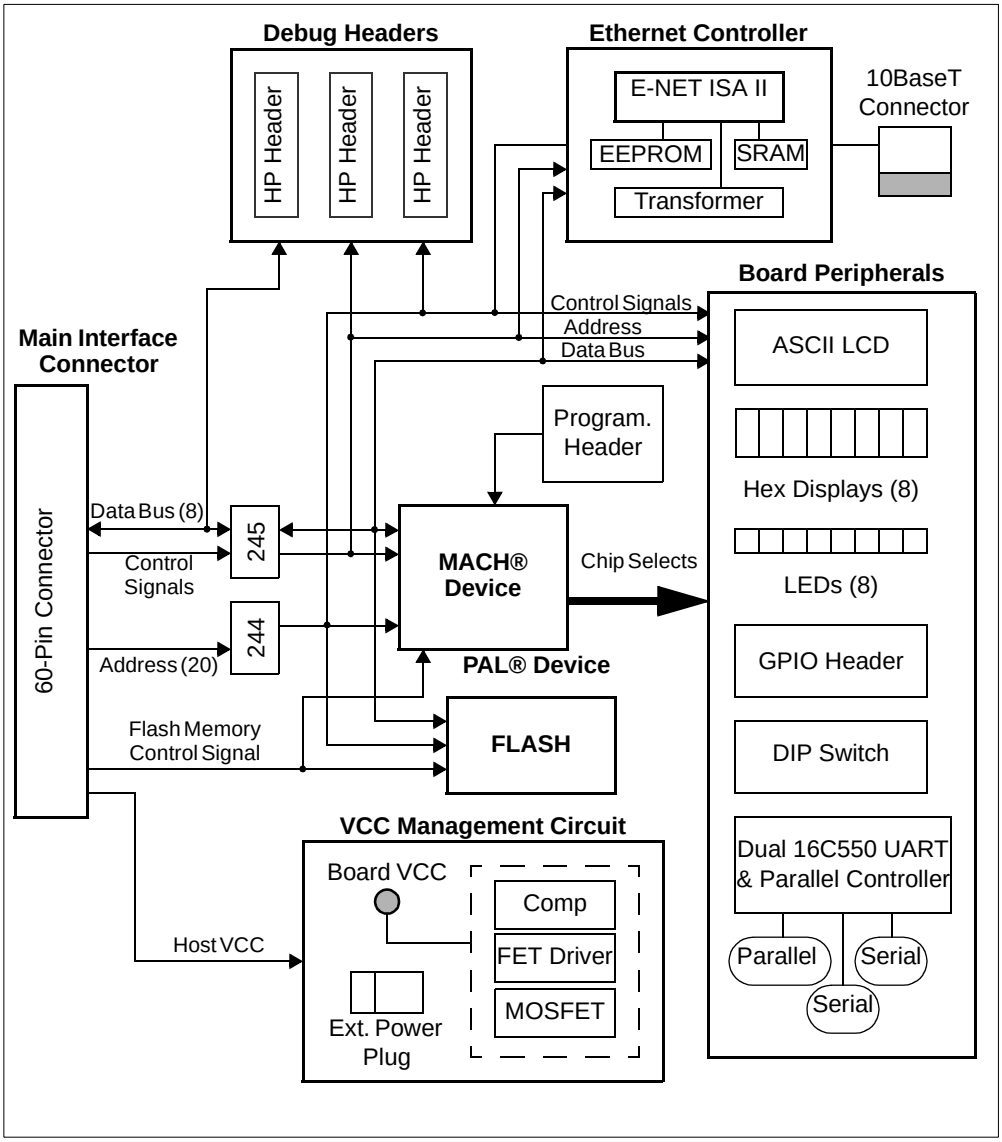


Figure 2-2. TIP Board Block Diagram

Serial Ports

NOTE: For more information about initializing or programming the serial ports, refer to the Texas Instruments TL16C552 data sheet.

The TIP board provides two 16550 RS-232 serial ports (9-pin, DCE). The TIP board contains a Texas Instruments TL16C552 chip that controls two 16550 Universal Asynchronous Receiver Transmitters (UARTs) and a parallel port. For more information about the parallel port, refer to page 2-10. These serial ports run up to 115200 baud and can be used with or without interrupts. The interrupt line from each serial port connects to a dedicated pin on the interface connector or through a shared interrupt circuit, which has one dedicated pin on the interface connector. Figure 2-3 illustrates the serial-port connector pinout.

Serial port 0 typically serves as a system debugging console. The TIP board typically sends debugging messages, such as asserts, fault messages, and trace statements, to this port. This interface can also support a command line or other method of accepting user input. For information about the I/O address locations, refer to Table 2-1 and Table 2-2.

Serial port 1 is typically dedicated for use by a software debugger. For example, the CAD-UL debugger can communicate with the board-level monitor using this port.

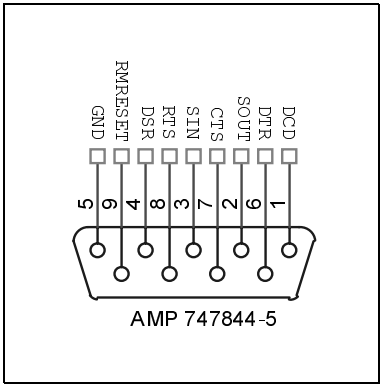


Figure 2-3. Serial-Port Connector Pinout

Table 2-1. I/O Map for 8-Bit Addressing for Serial Port

Address	Read/Write	Description
310h–317h	R/W	Serial port 0
318h–31Fh	R/W	Serial port 1

Table 2-2. I/O Map for 16-Bit Addressing for Serial Port

Address	Read/Write	Description
320h–32Eh	R/W	Serial port 0
330h–33Eh	R/W	Serial port 1

Programming the Serial Ports

This section contains general information about the programming operations and functions of the serial ports including operating mode, UART operation, interrupts, and registers.

Operating Mode

The 16550-compatible UART mode (FIFO mode) contains two 16-byte FIFOs for transmitting and receiving to off-load the CPU from repetitive service routines. The CPU can write 16 bytes to the transmit FIFO and use the THRE interrupt or poll the THRE bit to trigger another 16 bytes. The receive FIFO has a programmable trigger level that can interrupt the CPU at 1, 4, 8, or 16 bytes present. Writing a byte to a full transmit FIFO results in the last byte being lost. If the receive FIFO is full, receiving one more character generates an overrun error. The last character received is lost. The remaining 16 bytes in the FIFO are unchanged.

UART Operation

The UART converts serial data received on the serial input line (SIN) into parallel data that can be processed by the microcontroller. The UART also converts parallel data into serial data for transmission off the chip on the serial output line (SOUT). Data can be transmitted and received at the same time.

To generate the baud rate of the transfer, the UART clock is divided by a divisor value chosen by the programmer. The UART baud-rate generator automatically calculates the baud rate from the divisor value that is programmed into the two baud rate divisor registers (divisor latch LSB and divisor latch MSB). These registers are read at initialization to set the baud rate for the transfer.

Each byte of data is transferred using a format called a frame. The transmitter and receiver must agree on the frame format, in addition to the baud rate, or the transmission is not successful. The frame format is determined by the value written into the Line Control register. A frame consists of a start bit, five to eight data bits, an optional parity bit, and either 1, 1.5, or 2 stop bits. Transmission of a frame is initiated when software writes a byte to the Transmit Holding register. Reception of a frame is initiated when a start bit is received (the SIN input is driven Low for one baud-rate clock period). This start bit allows the receiver to synchronize its clock with the sender’s clock. Errors are reported in the Line Status register.

Interrupts

The serial port supports the standard UART interrupts as follows:

- Received Data Available
- Transmit Holding Register Empty
- Modem Status
- Receiver Line Status

If two interrupt sources are pending simultaneously, only the highest priority interrupt is indicated by the ID2–ID0 field of the Interrupt ID Register. When the interrupt source is cleared, a subsequent read from this port will return the next highest priority interrupt source.

Registers

The registers store three types of information: control, status, and data. The divisor latch access bit (DLAB) in the Line Control register (bit 7) is used with the address, read, and write inputs to select the register that is written to or read from. The Transmit Holding register and Receive Buffer register are data registers that hold from five to eight bits of data. If less than eight bits of data are transmitted, data is right justified to the least significant bit. Bit 0 of a data word is always the first serial data bit received and transmitted. The data registers are double-buffered so that read and write operations can be performed when the serial port is performing the parallel-to-serial or serial-to-parallel conversion.

The following registers are available on the serial port. The bits for these registers are described in the TI TL16C552 Specification.

- **Line Control Register:** This register is used to configure the format of the UART frame for data transfer, including character length, stop bits, and parity.
- **Divisor Latch LSB:** This register holds the least significant byte of a 16-bit baud rate clock divisor that is used to generate the 16x baud clock (when DLAB is 1).
- **Divisor Latch MSB:** This register holds the most significant byte of the clock divisor (when DLAB is 0).
- **Transmitter Holding Register:** The byte to be transmitted is written to this write-only register (when DLAB is 0).
- **Receive Buffer Register:** The received byte is read from this read-only register (when DLAB is 0). This register shares an address with the Transmit Holding Register.

- **Interrupt Enable Register:** This register enables the following serial port interrupts: modem status, receiver line status, transmitter holding empty, received data available, and time-out interrupts (when DLAB is 0).
- **Interrupt Identification Register:** This is a read-only register used to identify UART interrupts.
- **FIFO Control Register:** This is a write-only register used to enable and control the FIFO in 16650-compatible mode.
- **Line Status Register:** This register shows the status of the data transfer, including parity and framing errors, in addition to break and empty indicators.
- **Modem Control Register:** This register is used to enable interrupts and loopback diagnostic mode, and to assert $\overline{\text{RTS}}$ and $\overline{\text{DTR}}$.
- **Modem Status Register:** This register contains both real-time and latched status bits for $\overline{\text{DCD}}$, $\overline{\text{RIN}}$, $\overline{\text{DSR}}$, and $\overline{\text{CTS}}$.
- **Scratch Pad Register:** This is a general purpose I/O location used to hold temporary data and is not required for serial data transfer.

Configuring the Serial Port for DTE

Because the TIP board provides information to a terminal, it is considered data carrier equipment (DCE), and the serial ports are configured accordingly in its default design. You can also configure the TIP board to serve as data terminal equipment (DTE) if necessary.

To reconfigure serial port 0 to operate as DTE, use the following procedure:

1. Depopulate resistors R24 through R31.
2. Populate resistors R44 through R50, located on the back of the board.
3. A gender changer is now required for serial port 0, either on the TIP board end or the connecting cable end.

To reconfigure serial port 1 to operate as DTE, use the following procedure:

1. Depopulate resistors R34 through R41.
2. Populate resistors R51 through 57, located on the back of the board.
3. A gender changer is now required for serial port 1, either on the TIP board end or the connecting cable end.

Parallel Port

NOTE: For more information about initializing the parallel port, refer to the Texas Instruments TL16C552 specification.

The TIP board contains a Texas Instruments TL165552 chip that controls the parallel port and two serial ports (for more information about the serial ports, refer to "Serial Ports" on page 2-5). The parallel port can be used for very fast downloads or for connecting to a printer for logging purposes. The interrupt line from this port is connected to a dedicated pin on the interface connector, or connected to a shared interrupt circuit with one signal feeding back to the interrupt connector. Figure 2-4 illustrates the parallel port connector pinout. For information about the I/O address locations, refer to Table 2-3 and Table 2-4.

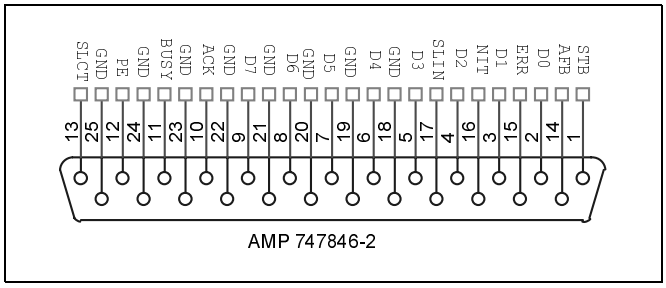


Figure 2-4. Parallel-Port Connector Pinout

Table 2-3. I/O Map for 8-Bit Addressing for Parallel Port

Address	Read/Write	Description
320h 321h 322h	R/W	Parallel port

Table 2-4. I/O Map for 16-Bit Addressing for Parallel Port

Address	Read/Write	Description
340h 342h 344h	R/W	Parallel port

Programming the Parallel Port

The TI TL16C552 parallel port interface is controlled primarily by software and provides all the status inputs, control outputs, and the control signals necessary for the external parallel port data buffers. Communication between the host and the peripheral is asynchronous. The parallel port data path is external to the microcontroller. The parallel port can be physically mapped to one of two different I/O locations or can be completely disabled. Only edge-triggered interrupts are supported.

The parallel port can connect to a Centronics-style printer interface. The parallel port is selected when chip select 2 (CS2) is low. The state of the read (IOR) and write (IOW) terminal controls the read or write function of the register. The Read Data register controls when the microprocessor can read information on the parallel bus.

The parallel port interface is mapped to 320h–322h (8-bit) or 340h–344h (16-bit). The following direct-mapped registers are available.

- **Read Data Register:** This register enables the microprocessor to read the information on the parallel bus.
- **Read Status Register:** This register enables the microprocessor to read the status of the printer in the six most significant bits. The status bits are: printer busy ($\overline{\text{BSY}}$); acknowledge ($\overline{\text{ACK}}$), a handshake function; paper empty (PE); printer selected ($\overline{\text{SLCT}}$); error ($\overline{\text{ERR}}$); and printer interrupt ($\overline{\text{PRINT}}$).
- **Read Control Register:** This register enables the state of the control lines to be read.
- **Write Data Register:** This register enables the microprocessor to write a byte to the parallel bus.
- **Write Control Register:** This register sets the state of the control lines. These states are: direction ($\overline{\text{DIR}}$); interrupt enable (IN2 EN); select in ($\overline{\text{SLIN}}$); initialize the printer ($\overline{\text{INIT}}$); autofeed the paper ($\overline{\text{AFD}}$); and strobe ($\overline{\text{STR}}$), which informs the printer of the presence of a valid byte on the parallel bus.

Ethernet Controller Port

NOTE: For information about initializing the 10BaseT Ethernet controller port and for correct setup and configuration of the Ethernet, refer to the *AMD Am79C961A PCnet-ISA+ Single Chip, Plug & Play Full Duplex Ethernet Controller for ISA* specification, order #18183.

The following are the 8-bit and 16-bit base addresses for the Ethernet:

Base addresses: 8-bit = 220h, 16-bit = 220h

The TIP board provides an Ethernet controller port that can be run in interrupt-driven mode. The Ethernet controller port supports software downloading and debugging over a network. For example, the CAD-UL debugger uses this capability.

Many devices already have an Ethernet controller port for use by the device application. However, it is usually not possible to use the device port for running the debugger. For example, if the system is stopped at a break point, and in the context of the debug monitor, then the driver and TCP/IP stack cannot run. However, the TIP board Ethernet controller port can be driven in a polled manner and can function independently from the application drivers and protocol stacks.

All of the interrupts from the Ethernet controller are gathered into a single interrupt signal that is connected to a dedicated pin on the interface connector or through a shared interrupt circuit. The Ethernet controller port can run in interrupt-driven mode.

LCD

NOTE: For more information about the LCD driver, refer to the Hitachi HD44780U-II LCD Driver data sheet.

Overview

The LCD, illustrated in Figure 2-5, is a 2 x 20 character, ASCII-decoded display for displaying text. The display can contain any type of text. Being ASCII-decoded, the LCD enables the user to represent alphabetical characters in the display. For example, writing 41h to a display location causes an A to appear on the LCD. Software can only write to this display; it cannot read the data back. For information about the I/O address locations, refer to Table 2-5 and Table 2-6.

The R67 potentiometer adjusts the contrast on the LCD. With the front of the board facing you (the main interface connector on the right), turn the thumb wheel on the R67 counterclockwise to dark the contrast on the LCD, making the characters more visible. When the LCD appears to be off and the board is powered up, try adjusting the potentiometer.

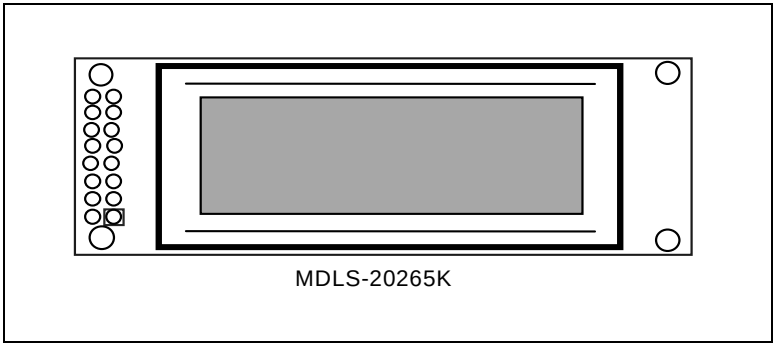


Figure 2-5. LCD Display

Table 2-5. I/O Map for 8-Bit Addressing for ASCII Display

Address	Read/Write	Description
30Ch	W	ASCII display—write control instructions
30Dh		ASCII display—busy flag / address read
30Eh		ASCII display—write data to LCD
30Fh		ASCII display—read data

Table 2-6. I/O Map for 16-Bit Addressing for ASCII Display

Address	Read/Write	Description
308h	W	ASCII display—write control instructions
30Ah		ASCII display—busy flag / address read
30Ch		ASCII display—write data to LCD
30Eh		ASCII display—read data

Configuring the LCD

You must set up the LCD for 8-bit operation and 8-digit x 2-line display. The power-up procedure detailed in "Board Power" on page 1-1 should initialize the LCD and leave it ready for configuration. For a quick configuration of the LCD, use the following procedure. If you encounter any problems or need more detailed information, refer to the Hitachi spec, HD44780U (LCD-II) (Dot Matrix Liquid Crystal Controller/Driver).

1. To set the LCD to 8-bit operation, 2-line display, and 5x8 dot character font, write 38h to the ASCII Display–Write Control Instructions address listed in the tables in "I/O Maps" on page 2-32. This write should clear the LCD screen.
2. To turn on the display and the cursor, write OEh to the ASCII Display–Write Control Instructions address. After this write, the cursor should appear on the LCD screen.

3. To set the LCD mode to increment the address by one and to shift the cursor to the right at the time of writing, write 06h to the ASCII Display–Write Control Instructions address.

The LCD is now ready to display the character for the respective 8-bit ASCII code. For example, writing 41h, 4Dh, and 44h in sequence to the ASCII Display–Write Data address should display “AMD” on the LCD screen.

Hexadecimal Display

The eight-segment hexadecimal display, illustrated in Figure 2-6, behaves much like the ASCII-decoded display, but displays binary information in hexadecimal format instead of ASCII-decoded data. This display contains eight hexadecimal digits for displaying values up to 32 bits wide. For example, during debugging, the value of a CPU or peripheral register can be written to the display for viewing without software decoding. Software can only write to this display; it cannot read back the value.

To display a hexadecimal value on the eight-segment display, write the bytes of the value to the appropriate I/O addresses as shown in Table 2-7 and Table 2-8.

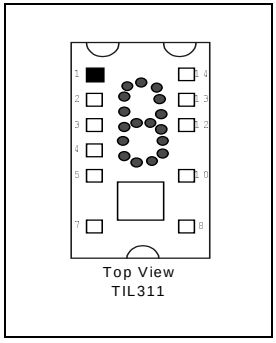


Figure 2-6. Eight-Segment Display

Table 2-7. I/O Map for 8-Bit Addressing for Hexadecimal Display

Address	Read/Write	Description
306h	W	Hex Display—Byte0: Hex digits 1 and 0
307h		Hex Display—Byte 1: Hex digits 3 and 2
308h		Hex Display—Byte 2: Hex digits 5 and 4
309h		Hex Display—Byte 3: Hex digits 7 and 6

Table 2-8. I/O Map for 16-Bit Addressing for Hexadecimal Display

Address	Read/Write	Description
310h	W	Hex display—Byte 0: Hex digits 1 and 0
312h		Hex display—Byte 1: Hex digits 3 and 2
314h		Hex display—Byte 2: Hex digits 5 and 4
316h		Hex display—Byte 3: Hex digits 7 and 6

General-Purpose LEDs

NOTE: For information about programming the information, refer to “Discrete LED Outputs Register” on page 2-19.

Eight LEDs

The eight general-purpose LEDs indicate status events or errors. Software can write to and read from the LED buffer. This capability supports a read-modify-write style of operation. For the TIP board, the green cathode (light) is used. Figure 2-7 illustrates the pinout of one of the LEDs.

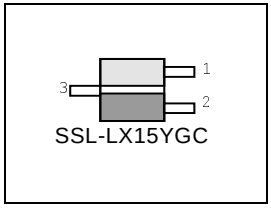


Figure 2-7. LED Pinout

Discrete LED Outputs Register

The following are the 8-bit and 16-bit addresses for the Discrete LED Outputs register:

Address: 8-bit = 305h, 16-bit = 300h

The Discrete LED Outputs register controls the eight individual LEDs on the TIP board, as shown in Table 2-9. This is a read/write register and can be used in a read-modify-write manner.

- When an LED bit is set to 1, the corresponding LED is turned on.
- When an LED bit is cleared to 0, the LED is turned off.

When writing to this register, the new value is latched at the LED drivers and the LEDs immediately reflect the state of the newly written bits.

Table 2-9. Discrete LED Outputs Register Bit Definitions

Bit	LED	Description
0	D0	The least significant LED (right-most LED)
1	D1	The next most significant LED
2	D2	The next most significant LED
3	D3	The next most significant LED
4	D4	The next most significant LED
5	D5	The next most significant LED
6	D6	The next most significant LED
7	D7	The most significant LED (left-most LED)

Debug Headers

The TIP board provides three HP logic analyzer debug headers for easy address, data, and control signal debug access. These headers (2 x 10-pin shrouded, low-profile header connector) allow connection between the logic analyzer and the test points using the HP flex cable header. The debug signals are on the following header as shown in Figure 2-8:

- Address bus
- Data bus and control signals
- Port chip selects and interrupts, and Ethernet interrupt

For pinout information, refer to the TIP board schematics included in your kit.

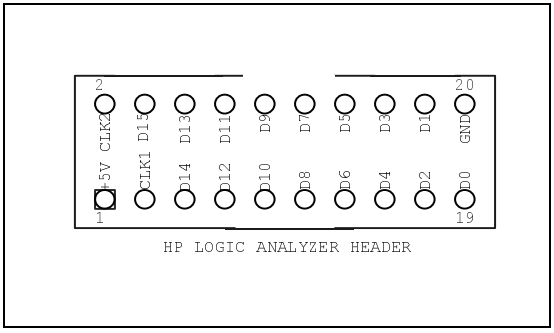


Figure 2-8. HP Logic Analyzer Header

General-Purpose Inputs and Outputs

The TIP board provides eight discrete inputs and eight discrete outputs through a standard 20-pin header, and eight discrete inputs through a DIP switch (illustrated in Figure 2-9). These are TTL/CMOS signals.

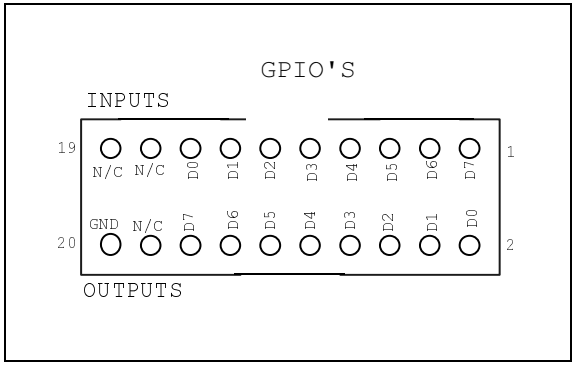


Figure 2-9. General-Purpose Input/Output Header

The inputs are general-purpose and allow the software to only read the input buffer. Therefore, these inputs do not support a read-modify-write style of operation. For information about reading the input buffer, see “Discrete Inputs Register” on page 2-22.

The outputs can be directly connected to a logic analyzer by a straight-through ribbon cable. These outputs support the tracing of software operation, *not* hardware. For example, to measure the interrupt latency of an interrupt and the interrupt handler execution time, simply toggle a bit on entry to and exit from the interrupt service routine (ISR). You can measure this with an in-circuit emulator (ICE), but it can be difficult to set up.

However, using the TIP board, you can measure the above operation by adding simple instructions to the ISR and using a logic analyzer. Software can write to and read from the output buffer thereby supporting a read-modify-write operation. For information about reading or writing the output buffer, see “Discrete Outputs Register” on page 2-23.

Discrete Inputs Register

The following are the 8-bit and 16-bit addresses for the Discrete Inputs register:

Address: 8-bit = 303h, 16-bit = 306h

The Discrete Inputs register returns the state of the eight general-purpose inputs on the TIP board, as shown in Table 2-10. These inputs appear on the INPUT side of connector P7. The value of each bit indicates the current state of its associated input.

- When a bit reads 1, the input is currently active (turned on/set to 5 V at the output).
- When a bit reads 0, the input is currently inactive (turned off/set to 0 V at the output).

Writing to this register has no effect on the state or operation of the TIP board.

Table 2-10. Discrete Inputs Register Bit Definitions

Bit	Pin	Description
0	15	The most significant (left most) output
1	13	The next most significant output
2	11	The next most significant output
3	9	The next most significant output
4	7	The next most significant output
5	5	The next most significant output
6	3	The next most significant output
7	1	The least significant (right most) output

Discrete Outputs Register

The following are the 8-bit and 16-bit addresses for the Discrete Outputs register:

Address: 8-bit = 302h, 16-bit = 304h

The Discrete Outputs register controls the eight individual general-purpose outputs on the TIP board, as shown in Table 2-11. These outputs appear on the OUTPUT side of connector P7. This read/write register can be used in a read-modify-write manner. When this register is read, the value of each bit indicates the current state of its associated output.

- When an output bit is set to 1, the output is currently active (turned on/set to 5 V at the output).
- When an output bit is cleared to 0, the output is currently inactive (turned off/set to 0 V at the output)

When writing to this register, the new value is latched to the output drivers and the output pins immediately reflect the state of the newly written bits.

Table 2-11. Discrete Outputs Register Bit Definitions

Bit	Pin	Description
0	2	The least significant (right most) output
1	4	The next most significant output
2	6	The next most significant output
3	8	The next most significant output
4	10	The next most significant output
5	12	The next most significant output
6	14	The next most significant output
7	16	The most significant (left most) output

DIP Switch

NOTE: For information about reading the state of the DIP switch, refer to “DIP Switch Inputs Register” on page 2-25.

Eight-Position DIP Switch

The TIP board provides an eight-position DIP switch, illustrated in Figure 2-10, that provides configuration information and mode control of the host software. For example, a DIP switch could enable special debugging features or change the system mode of operation. These switch inputs do not cause interrupts.

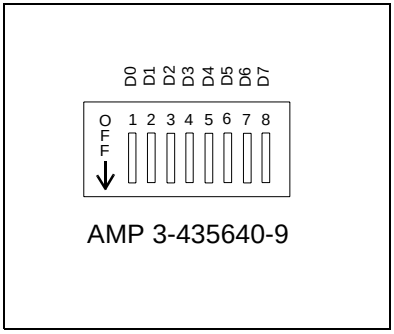


Figure 2-10. DIP Switch

DIP Switch Inputs Register

The following are the 8-bit and 16-bit addresses for the DIP Switch Inputs register:

Address: 8-bit = 304h, 16-bit = 302h

The DIP Switch Inputs register (read-only) returns the state of the eight-position DIP switch (SW4) on the TIP board, as shown in Table 2-12. When this register is read, the value of each bit indicates the current state of its associated DIP switch.

- When a bit reads 1, the input is currently active (turned on).
- When a bit reads 0, the input is currently inactive (turned off).

Writing to this register has no effect on the state or operation of the TIP board.

Table 2-12. DIP Switch Inputs Register Bit Definitions

Bit	SW	Description
0	0	The first (least significant/left-most) switch
1	1	The second switch
2	2	The third switch
3	3	The fourth switch
4	4	The fifth switch
5	5	The sixth switch
6	6	The seventh switch
7	7	The eighth (most significant/right-most) switch

Reset Button

The following are the 8-bit and 16-bit addresses for the Interrupt Reset register:

Address: 8-bit = 34Ah, 16-bit = 34Ah

The TIP board provides a reset button, illustrated in Figure 2-11. The reset button toggles a signal to the host board which then performs a system-wide hardware reset. The signal connected to the RESET pin on the interface connector (pin 50) should be an input to the host board and should connect to the main microcontroller reset. For example, on the Élan™ SC400 microcontroller, this signal is connected to the $\overline{\text{RESET}}$ input, which is corresponds to the POWERGOOD pin in the AT system architecture.

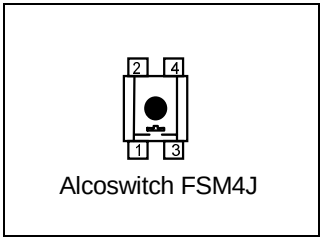


Figure 2-11. Reset Button

This same reset can also be performed remotely through pin 9 of the serial ports. To use this feature, you must remove resistor R32 (for serial port 1) and R42 (for serial port 0) from the TIP board, then populate resistors R33 and R43 for serial ports 0 and 1 respectively.

The TIP board discrete registers should not respond to the hardware reset signal. However, this condition is not certain, even under normal conditions. For example, a hardware reset should not change the state of the LCD, hex display, discrete LEDs, discrete inputs, or discrete outputs. A hardware reset does reset the serial ports and the parallel port.

Flash Memory

NOTE: For more information about the flash memory, refer to the Am29F040 data sheet.

The TIP board is equipped with Flash memory. The socket in which the Flash memory is mounted supports a variety of Flash memory options (any 8-bit device) that can be installed in place of the existing unit. Figure 2-12 illustrates the Flash memory for identification purposes.

Figure 2-12. 8-Bit Flash Memory Identification

Using the Flash Memory

You can program the Flash memory with diagnostics, a monitor, or a small application. However, a primary design function of the Flash memory is to provide the host board with an alternate boot location. For example, if the host board has a soldered-down Flash memory device and that Flash memory becomes corrupted, you can reprogram the Flash memory without removing it from the board by using the functions of the TIP board. You can accomplish this task by programming the TIP board Flash memory with a utility program, like the AMD E86MON™ software, which has a reprogram function to allow copying the contents of the TIP board Flash memory to the host board Flash memory.

More specifically, when the TIP board is hosted by an Am186™CC/CH/CU device microcontroller that is connected to a monitor, the TIP board can reprogram the Customer Development Platform (CDP) on-board Flash memory. With the Flash memory on the TIP board being programmed with E86MON software, the CDP can be booted through the TIP board. Now you can begin the reprogramming sequence by entering **z** on the keyboard. Follow a few simple instructions, and the Flash memory on the Am186CC/CH/CU CDP device is reprogrammed. This is a simpler process than removing the soldered down Flash memory on the CDP board, programming it on a programmer, then re-soldering the Flash memory back on to the CDP device.

Selecting the Flash Memory

Jumper block 1 (JP1) selects which Flash memory is used, either the internal (TIP board) or the external (host board) Flash memory. To select the use of the internal 8-bit DIP Flash, jumper pins 2 and 3 on JP1 and drive a logic low to the $\overline{\text{FLASHCS}}$ signal (pin 58 of P1 of the main interface connector). To select the use of the host board Flash, jumper pins 1 and 2 on JP1. Figure 2-13 indicates the pin locations on JP1.

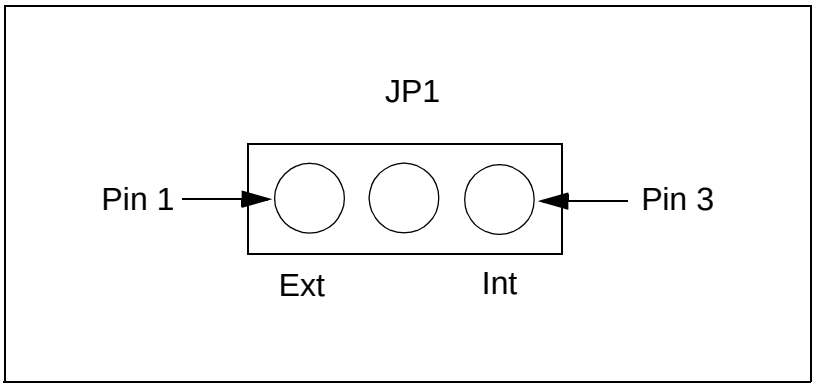


Figure 2-13. JP1 Pin Locations

Interrupts

WARNING: The peripheral interrupt signals on the TIP board are not terminated. As a result, the software must ensure that each peripheral interrupt is enabled so the interrupt input to the OR gate that is driving the MAIN_IRQ signal is not floating. Refer to the peripheral data sheets for information about how this is done. Alternately, the target board may use pull-down resistors on the peripheral interrupt signals.

The TIP board supports interrupts with the following features:

- An interrupt push button (SW2)
- Five interrupt sources on the board: two serial ports, the parallel port, the Ethernet controller port, and the user-interrupt button

For designs that can accommodate all the individual signals and want to have dedicated interrupts, the serial ports, parallel port, and Ethernet controller port have individual interrupt signals going to the main interface connector (P1). For designs that have limited connector space, these same signals are also logically ORed together on the MAIN_IRQ signal, which is also routed to the main connector.

The interrupt generated by the interrupt push button is routed to the main connector only through the MAIN_IRQ signal. When the interrupt button is pressed, the SW_IRQ signal (generated from the MACH), goes high causing the MAIN_IRQ signal to also go high. The SW_IRQ signal will remain high until the Interrupt Reset register (address 34Ah) in the MACH is written to. For the address of this register, refer to “I/O Maps” on page 2-32.

MACH® Device

The peripherals on the TIP board are interfaced to the host system through a macro array CMOS high-density/high-performance (MACH®) device that provides individual chip selects for each on-board peripheral device (except the Ethernet controller port). This interface gives the TIP board a degree of independence from the host system. For a complete listing of the current MACH® device code, see Appendix A, “MACH® Device Equations”.

I/O Address Mode

The TIP board supports 8-bit addressing or 16-bit addressing. The 8-bit or 16-bit addressing is determined by the SW3 switch.

Selecting the optional 8-bit cycle mode changes the MACH device I/O map address. Note that the peripheral access addresses change. For the I/O map addresses in the different addressing modes, see “I/O Maps” on page 2-32. If you need to run the TIP board in 8-bit mode on a 186 family board, you must select the 8-bit mode.

For example, to select the 8-bit I/O addressing mode on the Am186ED processor, write 01b to the Auxiliary Configuration register at physical location FFF2h. By default, this register is set to 00b for 16-bit I/O addressing. Consult the host microcontroller documentation for the correct 8- or 16-bit I/O addressing configuration.

I/O Maps

Table 2-13 lists the 8-bit I/O address locations for the respective peripherals, and Table 2-14 lists the 16-bit I/O address locations.

Table 2-13. I/O Map for 8-Bit Addressing

Address	Read/Write	Description
302h	R/W	Discrete Outputs register
303h	R	Discrete Inputs register
304h	R	DIP Switch Inputs register
305h	R/W	Discrete LED Outputs register
306h 307h 308h 309h	W	Hex Display–Byte 0: Hex digits 1 and 0 Hex Display–Byte 1: Hex digits 3 and 2 Hex Display–Byte 2: Hex digits 5 and 4 Hex Display–Byte 3: Hex digits 7 and 6
30Ch 30Dh 30Eh 30Fh	W	ASCII display–write control instructions ASCII display–busy flag / address read ASCII display–write data to LCD ASCII display–read data
310h–317h	R/W	Serial port 0
318h–31Fh	R/W	Serial port 1
320h 321h 322h	R/W	Parallel port
34Ah	W	Interrupt reset (write to this address to reset the push button interrupt signal)
348h	R	Version register
220h	R/W	Ethernet base address

Table 2-14. I/O Map for 16-Bit Addressing

Address	Read/Write	Description
300h	R/W	Discrete LED Outputs register
302h	R	DIP Switch Inputs register
304h	R/W	Discete Outputs register
306h	R	Discrete Inputs register
308h 30Ah 30Ch 30Eh	W	ASCII display–write control instructions ASCII display–busy flag / address read ASCII display–write data to LCD ASCII display–read data
310h 312h 314h 316h	W	Hex display–Byte 0: Hex digits 1 and 0 Hex display–Byte 1: Hex digits 3 and 2 Hex display–Byte 2: Hex digits 5 and 4 Hex display–Byte 3: Hex digits 7 and 6
320h–32Eh	R/W	Serial port 0
330h–33Eh	R/W	Serial port 1
340h 342h 344h	R/W	Parallel port
34Ah	W	Interrupt reset (write to this address to reset the push button interrupt signal)
348h*	R	Version register
220h	R/W	Ethernet base address

* Currently, the Version register is only accessible by the Am186 microcontrollers.

Timing

Certain 186 microcontroller boards require added wait states to accommodate the timing specifications of the 16C550 UARTs. For example, on the SD186ED demonstration board, the $\overline{\text{PCS}}$ and $\overline{\text{MCS}}$ Auxiliary (MPCS) register needs to be set to a defined state by writing 8038h to physical location FFA8h. Then, insert three wait states to the Peripheral Chip Select (PCS) register by writing 0073h to physical location FFA4h. Refer to the 16C550 specification and the microcontroller manual of the host board to determine the correct configuration and the appropriate method of adding wait states.

Version Register

NOTE: Currently, the Version register is only accessible by the Am186 microcontrollers.

The following are the 8-bit and 16-bit addresses for the Version register:

Address: 8-bit = 348h, 16-bit = 348h

The Version register contains the version of the code that is running in the MACH® programmable device and in the revision of the TIP board. This is a read-only register. Table 2-15 lists the bit mappings for this register.

Table 2-15. Version Register Bit Definitions

Bits	Read/Write	Description
0–3	R	**MACH Device Code Version** These read-only bits reflect the version of the code in the MACH programmable part, which controls the TIP board. This value starts at one and is incremented every time a new version of the MACH device code is released from AMD. This value enables software to detect which version of the MACH device is resident on the TIP board and to modify its behavior accordingly. This value also helps you determine when to upgrade the MACH device code. Writing to these bits has no effect on the state of these bits, the MACH device, or any other devices on the TIP board.
4–7	R	**TIP Board Revision** These read-only bits reflect the revision of the physical TIP board. These bits are not necessarily incremented with every spin of the TIP board; incrementing depends on the significance of any changes made to the board. Writing to these bits has no effect on the state of these bits, the MACH device, or any other devices on the TIP board.

Appendix A

MACH® Device Equations

This appendix contains the version of the MACH® code for the TIP board that is current as of the printing of this manual. Access the AMD website at **www.amd.com**, and follow the Embedded Processors link to find the latest version of this code.

NO SUPPORT OBLIGATION: AMD is not obligated to furnish, support, or make any further information, software, technical information, know-how, or show-how available to you.

NO WARRANTIES, LIMITATIONS OF LIABILITY: AMD is providing these materials to you "as is, with all faults." AMD makes no warranty whatsoever, express, implied, statutory, contractual or otherwise with respect to the materials, and expressly disclaims any implied warranty of merchantability, fitness for a particular purpose, title or non-infringement and any warranties arising by virtue of custom of trade or course of dealing. AMD also advises you that the materials specify components not manufactured or sold by AMD in the normal course of its business.

In no event shall AMD be liable for any indirect, punitive, special, incidental or consequential damages in connection with or arising out of your use of or inability to use the materials, including but not limited to loss of profits, use, data or other economic advantage. If there shall, notwithstanding the above provisions, at any time be or arise any liability on the part of AMD by virtue of this agreement and/or the materials furnished by AMD to you, you agree that in no event will the total aggregate liability of AMD for any claims, losses, or damages exceed $5,000. The foregoing limitation of liability is complete and exclusive, shall apply even if AMD has been advised of the possibility of claims, losses, or damages exceeding such limit, and shall apply regardless of the success or effectiveness of any other remedies possessed by you or third parties. This limitation of liability reflects an allocation of risk between AMD and you in view of the fact that AMD has not charged you for the materials or their use.

```
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
"   MACH Code Version 2.0 for Test Interface Port (TIP) board                             "
"                                                                                         "
"   AMD's MACH 4-128/64 provides the I/O mapping for the TIP board.                       "
"   With address lines 0-19, AEN, Read, and Write signals as primary inputs to the MACH  "
"   the MACH code generates all the chip select for the TIP peripherals with the          "
"   exception of the Flash and the Ethernet.                                              "
"                                                                                         "
"   Written:    September 1998                                                            "
"     For  :    Advanced Micro Devices Austin / EPD                                       "
"     By   :    Gino Davis - System Engineering                                           "
"                                                                                         "
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""



"\\\\\\\\\\\\ DECLARATIONS
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
  INPUT   a19..a0 ;            " Address bus, bits a0 - a19
  INPUT   d0..d7 ;             " Data bus, bits d0 - d7
  INPUT   /rd ;               " Read strobe - indicates the microcontroller is performing
                               "   a read cycle
  INPUT   /wr ;              " Write strobe - indicates the microcontroller is performing
                               "   a write cycle
  INPUT   /flashrd ;          " Flash read strobe - for controller with a different read
                               "   signal for flash access
  INPUT   /flashwr ;         " Flash write strobe - for controller with a different write
                               "   signal for flash access
  INPUT   /s2 ;              " Bus cycle indicator for 186 microcontroller - High for
                               "   logical memory or low for I/O
  INPUT   /aen ;             " DMA Addess Enable - When low allows for I/O address
                               "   decoding
  INPUT   /tipflhcs ;          " TIP Flash Chip Select
  INPUT   /swintrp ;          " Switch interrupt - goes low when Interrupt button is
                               "   pressed
  INPUT   sel186 ;            " Select 186 - signal identifing that 186 microcontroller
                          "  board is hosting the TI  INPUT   selbit ;
                              " 8 or 16 Bit I/O adressing select signal - H: 16 bit or
                               "   L: 8 bit
  INPUT   reset ;             " TRESET signal used for qualifing interrupt signal in MACH


  OUTPUT    /db2g ;          " Data buffer enable, enable buffer for reading or write data
  OUTPUT    db2dir ;           " Data bus direction, steers data bus for read or write
                               " Data direction db2dir H: A to B  or  L: B to A
  OUTPUT    /swirq ;          " Latched interrupt switch, when swintrp goes low, swirq is
                               "   latched high
  OUTPUT   swirqid ;
  OUTPUT    hdrltclk;           " General purpose output (GPO) header chip select for
                               "  writing
  OUTPUT    /hdrltoe ;          " General purpose output (GPO) header chip select for
                               "  reading
  OUTPUT    /hdrbfoe ;        " General purpose input (GPI) header chip select, read only
  OUTPUT    /dipswoe ;         " Dip switch chip select, read only
  OUTPUT    /ledoe ;           " LED chip select for reading
  OUTPUT    ledclk ;           " LED chip select for writing
  OUTPUT    lcden ;            " LCD enable chip select
  OUTPUT    lcdrs ;           " LCD register select chip select, H: data input  L:
                               "  instruction input
  OUTPUT    lcdrw ;            " LCD read/write chip select, H: read  L: write
  OUTPUT    /hexcs0 ;          " Hex Display chip select for byte 0
  OUTPUT    /hexcs1 ;          " Hex Display chip select for byte 1
  OUTPUT    /hexcs2 ;          " Hex Display chip select for byte 2
```

```
  OUTPUT    /hexcs3 ;           " Hex Display chip select for byte 3
  OUTPUT    /sercs0 ;           " Serial port 0 (COM 0) chip select
  OUTPUT    /sercs1 ;           " Serial port 1 (COM 1) chip select
  OUTPUT    /parcs ;            " Parallel port chip select


  MACRO addr [a19,a18,a17,a16,a15,a14,a13,a12,a11,a10,a9,a8,a7,a6,a5,a4,a3,a2,a1,a0];



             " TIP VERSION REGISTER  (Read Only-Input 348h)
             " {Bits 0:3 - code version, Bits 4:7 - board revision}
             " NOTE: data lines read INVERTED (example /do0 reads as a high)

             " THIS IS MACH CODE VERSION 3 FOR BOARD REVISION 1

  OUTPUT  /do0 ENABLED_BY [s2 * (addr = 348h) * rd ] ;
  OUTPUT  /do1 ENABLED_BY [s2 * (addr = 348h) * rd ] ;
  OUTPUT  do2 ENABLED_BY [s2 * (addr = 348h) * rd ] ;
  OUTPUT  do3 ENABLED_BY [s2 * (addr = 348h) * rd ] ;
  OUTPUT  /do4 ENABLED_BY [s2 * (addr = 348h) * rd ] ;
  OUTPUT  do5 ENABLED_BY [s2 * (addr = 348h) * rd ] ;
  OUTPUT  do6 ENABLED_BY [s2 * (addr = 348h) * rd ] ;
  OUTPUT  do7 ENABLED_BY [s2 * (addr = 348h) * rd ] ;
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""



"\\\\\\\\\\\\ BEGIN PROGRAM
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""


                "PUSH BUTTON SWITCH INTERRUPT
swirq = [swintrp * /reset];

swirqid = swirq
        + [swirqid * /[(addr = 34Ah) * wr ] * /reset] ;

     " {Once the interrupt switch is pushed, a low signal is latched until the address 34Ah
     " has been written to. Writing any value to this address resets swirqid to a high
     "  state.}



IF (selbit = 1 ) THEN                " {IF SW3 IS IN 16 BIT POSITION, ACCESS THE FOLLOWING
                                     "   16 BIT EQUATIONS}

    IF (sel186 = 1)  THEN                 " {IF CONNECTED TO AN 186 MICROCONTROLLER HOST
                                          "   BOARD}


"\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\  186 MICROCONTROLLER 16 BIT I/O ADDRESS EQUATIONS  ///////
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

                  " LED  (Output)
        ledclk = [(addr = 300h) * wr * s2 ]
               + [ledclk * wr] ;

                  " LED (Input)
```

```
     ledoe = [(addr = 300h) * rd * s2 ]
           + [ledoe * rd] ;

               " DIP SW (Input)
     dipswoe = [(addr = 302h) * rd * s2 ]
             + [dipswoe * rd] ;

               " Discrete GPO (Output)
     hdrltclk = [(addr = 304h) * wr * s2 ]
              + [hdrltclk * wr] ;

               " Discrete GPO (Input)
     hdrltoe = [(addr = 304h) * rd * s2 ]
             + [hdrltoe * rd] ;

               " Discrete GPI (Input)
     hdrbfoe = [(addr = 306h) * rd * s2 ]
            +  [hdrbfoe * rd] ;



                           " >>>>>>>>>>  LCD  <<<<<<<<<<

               "  ENABLE  (308h-30Eh)             (00000000 0011 0000 1)
    lcden = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/a5*/
      a4 *a3 * rd * s2 ]
          + [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/a5*/
              a4 *a3 * wr * s2 ]
           + [lcden * [wr + rd] ] ;

               " SELECTS REGISTERS
    lcdrs = a2 * [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
      a5*/a4 *a3] ;
                  "{If a2 is a zero(0) you are communicating to the Instruction
                     Register}
                   "{If a2 is a one(1) you are communicating to the Data Register}

               " SELECTS DATA READ OR WRITE
    lcdrw = a1 * [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
      a5*/a4 *a3] ;
                    "{If a1 is zero(0) it's a write cycle}
                    "{If a1 is one(1) it's a read cycle}



             " >>>>>>>>>>>> HEX DISPLAYS  <<<<<<<<<<<<

               " Byte 0 (Output)
     hexcs0 = [(addr = 310h) * wr * s2 ]
            + [hexcs0 * wr] ;

                " Byte 1 (Output)
     hexcs1 = [(addr = 312h) * wr * s2 ]
            + [hexcs1 * wr] ;

              " Byte 2 (Output)
hexcs2 = [(addr = 314h) * wr * s2 ]
            + [hexcs2 * wr] ;

               " Byte 3 (Output)
     hexcs3 = [(addr = 316h) * wr * s2 ]
            + [hexcs3 * wr] ;
```

```
                  " >>>>>>>>>>>>> PORTS <<<<<<<<<<<<<
"THE BELOW CHIP SELECTS ARE LATCHED WITH THEIR ADDRESS.

                  " COM Port 0 (320h - 32Eh)            (00000000 0011 0010 ...0)
      sercs0 = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*a5*/
         a4 */a0 * s2 ]
             + [sercs0 * /a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/
                  a6*a5*/a4 */a0];


                 " COM Port 1 (330h - 33Eh)             (00000000 0011 0011 ...0)
       sercs1 = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*a5*a4
         */a0 * s2 ]
            + [sercs1 * /a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/
                  a6*a5*a4 */a0];

                " Parallel Port  (340h - 344h)         (00000000 0011 0100 0..0)
        parcs =  [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*a6*/a5*/
          a4 */a3*/a0 * s2 ]
              + [parcs * /a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /
                  a7*a6*/a5*/a4  */a3*/a0];



    ELSE                " {IF CONNECTED TO AN ISA SYSTEM }

"\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\  ISA SYSTEM 16 BIT I/O ADDRESS EQUATIONS  ////////////////
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

                  " LED  (Output)
        ledclk = [(addr = 300h) * wr * aen] ;

                  " LED (Input)
        ledoe = [(addr = 300h) * rd * aen] ;

                  " DIP SW (Input)
        dipswoe = [(addr = 302h) * rd  * aen] ;

                   " Discrete GPO (Output)
        hdrltclk = [(addr = 304h) * wr * aen] ;

                   " Discrete GPO (Input)
        hdrltoe = [(addr = 304h) * rd * aen] ;

                  " Discrete GPI (Input)
        hdrbfoe = [(addr = 306h) * rd * aen] ;



                      " >>>>>>>>>>  LCD  <<<<<<<<<<

                   "  ENABLE  (308h-30Eh)            (00000000 0011 0000 1)
       lcden = /a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/a5*/a4
         *a3 * rd * aen
            + /a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/a5*/a4
                 *a3 * wr * aen;

                  " SELECTS REGISTERS
       lcdrs = a2 * [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
```

```
        a5*/a4 *a3] ;
                     "{If a2 is a zero(0) you are communicating to the Instruction
                        Register}
                     "{If a2 is a one(1) you are communicating to the Data Register}

                " SELECTS DATA READ OR WRITE
      lcdrw = a1 * [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
        a5*/a4 *a3] ;
                      "{If a1 is zero(0) it's a write cycle}
                      "{If a1 is one(1) it's a read cycle}



                    " >>>>>>>>>>>> HEX DISPLAYS  <<<<<<<<<<<<

                " Byte 0 (Output)
       hexcs0 = [(addr = 310h) * wr * aen] ;

                " Byte 1 (Output)
       hexcs1 = [(addr = 312h) * wr * aen] ;

                " Byte 2 (Output)
       hexcs2 = [(addr = 314h) * wr * aen] ;

                " Byte 3 (Output)
       hexcs3 = [(addr = 316h) * wr * aen] ;



                    " >>>>>>>>>>>> PORTS <<<<<<<<<<<<

                " COM Port 0 (320h - 32Eh)              (00000000 0011 0010 ...0)
       sercs0 = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*a5*/
        a4 */a0 * aen] ;

                " COM Port 1 (330h - 33Eh)              (00000000 0011 0011 ...0)
      sercs1 = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*a5*a4
        */a0 * aen] ;

                " Parallel Port  (340h - 344h)          (00000000 0011 0100 0..0)
      parcs = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*a6*/a5*/a4
        */a3*/a0 * aen] ;


    END IF;


ELSE                        " {IF SW3 IS IN 8 BIT POSITION, ACCESS THE FOLLOWING 8 BIT
                               EQUATIONS}



  IF (sel186 = 1)  THEN

"\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ 186 MICROCONTROLLER 8 BIT I/O ADDRESS EQUATIONS ////////
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""


              " DISCRETE GPO  (Output)
       hdrltclk = [(addr = 302h) * wr * s2]
```

```
         + [hdrltclk * wr] ;

           " DISCRETE GPO  (Input)
hdrltoe = [(addr = 302h) * rd * s2]
        + [hdrltoe * rd] ;

           " DISCRETE GPI  (Input)
hdrbfoe = [(addr = 303h) * rd * s2]
        + [hdrbfoe * rd] ;

           " DIP SW  (Inputs)
dipswoe = [(addr = 304h) * rd * s2]
        + [dipswoe * rd] ;

           " LED  (Output)
ledclk = [(addr = 305h) * wr * s2]
        + [ledclk * wr] ;

           " LED  (Input)
ledoe = [(addr = 305h) * rd * s2]
      + [ledoe * rd] ;



            " >>>>>>>>>>>> HEX DISPLAYS <<<<<<<<<<<<

           " Byte 0 (Output)
hexcs0 = [(addr = 306h) * wr * s2]
       + [hexcs0 * wr] ;

         " Byte 1 (Output)
hexcs1 = [(addr = 307h) * wr * s2]
       + [hexcs1 * wr] ;

           " Byte 2 (Output)
hexcs2 = [(addr = 308h) * wr * s2]
       + [hexcs2 * wr] ;

           " Byte 3 (Output)
hexcs3 = [(addr = 309h) * wr * s2]
       + [hexcs3 * wr] ;



                  " >>>>>>>>>>  LCD  <<<<<<<<<<

           " ENABLE  (30Ch-30Fh)               (00000000 0011 0000 11)
lcden = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/a5*/
  a4 * a3*a2 * rd * s2]
      + [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/a5*/
           a4 *a3*a2 * wr * s2]
       + [lcden * [rd+ wr]] ;

           " SELECTS REGISTERS
lcdrs = a1 * [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
  a5*/a4 * a3*a2] ;
               "{If a1 is a zero (0) you are communicating to the Instruction
                  Register}
                "{If a1 is an one (1) you are communicating to the Data Register}

           " SELECTS DATA READ OR WRITE
lcdrw = a0 * [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
  a5*/a4 * a3*a2] ;
```

```
                         "{If a0 is zero (0) it is a write cycle}
                         "{If a0 is one (1) it is a read cycle}



                       ">>>>>>>>>>>>  PORTS <<<<<<<<<<<<
"THE BELOW CHIP SELECTS ARE LATCHED WITH THEIR ADDRESS.

                  " COM Port 0 (310h - 317h)              (00000000 0011 0001 0)
        sercs0 = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
         a5*a4 */a3 * s2 ]
             + [sercs0 * /a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/
                  a6*/a5*a4*/a3] ;


                  " COM Port 1 (318h - 31Fh)              (00000000 0011 0001 1)
        sercs1 = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
         a5*a4 *a3 * s2 ]
             + [sercs1 * /a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/
                  a6*/a5*a4*a3] ;


                  " Parallel Port  (320h - 322h)          (00000000 0011 0010 00)
        parcs =  [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*a5*/
         a4 */a3*/a2 * s2 ]
              + [parcs * /a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/
                  a6*a5*/a4*/a3*/a2] ;




  ELSE

"\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ ISA SYSTEM 8 BIT I/O ADDRESS EQUATIONS /////////////////
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""



                  " DISCRETE GPO  (Output)
        hdrltclk = [(addr = 302h) * wr * aen] ;

                  " DISCRETE GPO  (Input  302)
        hdrltoe = [(addr = 302h) * rd * aen] ;

                  " DISCRETE GPI  (Input)
        hdrbfoe = [(addr = 303h) * rd * aen] ;

                " DIP SW  (Inputs)
        dipswoe = [(addr = 304h) * rd * aen] ;

                " LED  (Output)
        ledclk = [(addr = 305h) * wr * aen] ;

                 " LED  (Input)
        ledoe = [(addr = 305h) * rd * aen] ;


                      " >>>>>>>>>>>> HEX DISPLAYS <<<<<<<<<<<<

                  " Byte 0 (Output)
        hexcs0 = [(addr = 306h) * wr * aen] ;
```

```
                " Byte 1 (Output)
        hexcs1 = [(addr = 307h) * wr * aen] ;

                " Byte 2 (Output)
        hexcs2 = [(addr = 308h) * wr * aen] ;

                " Byte 3 (Output)
        hexcs3 = [(addr = 309h) * wr * aen] ;



                          " >>>>>>>>>>  LCD  <<<<<<<<<<

                " ENABLE  (30Ch-30Fh)               (00000000 0011 0000 11)
      lcden = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/a5*/
          a4 * a3*a2 * rd * aen]
               + [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/a5*/
                    a4 *a3*a2 * wr * aen] ;

                " SELECTS REGISTERS
        lcdrs = a1 * [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
          a5*/a4 * a3*a2] ;
                        "{If a1 is a zero (0) you are communicating to the Instruction
                           Register}
                         "{If a1 is an one (1) you are communicating to the Data Register}

                " SELECTS DATA READ OR WRITE
       lcdrw = a0 * [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
          a5*/a4 * a3*a2] ;
                              "{If a0 is zero (0) it is a write cycle}
                              "{If a0 is one (1) it is a read cycle}



                          ">>>>>>>>>>>>  PORTS <<<<<<<<<<<<

                " COM Port 0 (310h - 317h)          (00000000 0011 0001 0)
        sercs0 = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
          a5*a4 */a3 * aen ] ;


                " COM Port 1 (318h - 31Fh)          (00000000 0011 0001 1)
        sercs1 = [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*/
          a5*a4 *a3 * aen ] ;



                " Parallel Port  (320h - 322h)        (00000000 0011 0010 00)
        parcs =  [/a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7*/a6*a5*/
          a4 * /a3*/a2 * aen ] ;


  END IF;


END IF;



             " DATA BUS/TRANSCEIVER ENABLE
db2g = /a19*/a18*/a17*/a16*/a15*/a14*/a13*/a12 * /a11*/a10*a9*a8 * /a7
      + tipflhcs
```

```
 + [sel186 * aen * [rd + wr]]      " Enable data buffer for 186's AEN
+ [db2g * sel186 * aen] ;        " For Enet, data had required hold time after rising
                                     edge of read,
                                " therefore had to latch the buffer enable with 186's
                                     AEN.


        " DATA/TRANSCEIVER DIRECTION   (DATA DIRECTION DEFAULTS TOWARDS THE TIP)
            db2dir = /rd + [/flashrd * tipflhcs] ;

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
```

Index

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

O

P

R

S

T

U

V

W