Model 2010
8031 Single Board Computer
Copyright 1986, 1988, GTEK, Inc.

Document Number 2010M.PUB

First Revision, March 23, 1988

Please read installation section before applying power.

Chapter 1

Introduction

1.1 Hardware Overview

The GTEK Model 2010 single board computer is intended to be the best single board computer to use for control applications ever made.

The 2010 has 40 programmable I/O lines. Each I/O line may be programmed for either input or output without regard to its position on the connector. Eight of those 40 are PORT 1 of the processor. The other 32 are on 2 Z80 PIO chips.

In addition to those 40 I/O lines, are lines to be used to expand the functions of the 2010 board. AD0-AD7, ALE, RD, WR, PSEN, A8-A15, ROM, CMM, VDD, P1.5, P1.6, P1.7, Vcc, Vdd, DTR and GROUND are brought to a 34 pin expansion bus (see appendix A). This makes expansion to more memory or boards, like a D/A A/D converter or more I/O, easily attached. With proper design, the board could just piggy back on top of the 2010.

The 2010 has a built in 5 volt regulated power supply. It needs to be connected to a single ended power supply capable of at least 9 volts at 500 milliamps to be adequately powered. If you attach expansion boards to it, you must use an adequate power supply. The RS-232 uses a MAX-232 chip to obtain the +- 12 volts for the RS-232 supply.

With the proper power supply, you can supply about 250 to 500 milliamps to peripherals. You must provide for adequate cooling of the regulator, however. The regulator is capable of delivering up to 1 amp provided proper heat sinking is provided.

1.2 Hardware Specifications

Physical Size:
3.55 x 6.90 x .6 inches
90.17 x 175.26 x 15.24 millimeters

Weight:
6 ounces (170 grams)

Power Requirements:
9 Volts at 500 milliamps

Direct I/O Lines Programmable As Input Or Output:
40

EXpansion I/O Addresses Through Expansion Bus:
C000-FFFFH, and 0000-7FFFH if the EEprom is not used.

PROCESSORS: 8031 at 11 MHz (standard)
(optional) 8751, 8751H at 11 MHz
8032 at 11 MHz
8052AHBASIC
(Also see your processor specification sheet)

1.3 Software Overview

Much of the software for the 2010 is strictly communications software. That is, you could use other programs to communicate with the 2010, but our software will recognize upload, download and other commands and handle them accordingly.

Most of the commands are handled by the 2010 directly, rather than on your computer. This allows it to run in the same way on virtually any computer or terminal.

The software used to handle communications is called CPX.COM. A program called PINSTALL is provided to install it for the Baud Rate or COM port you are using. Use CPX to upload and download programs in Intel Hex format and to communicate with the 2010.

1.4 SOFTWARE SPECIFICATIONS

CPX Communicates at from 300 Baud through 57,600 Baud. It has commands to read and save Intel Hex files, specify load addresses, and load/execute a program.

1.5 FIRMWARE OVERVIEW

Much of the versatility of the model 2010 board comes from being able to refresh ram through an interrupt service routine every 2 milliseconds. The built in monitor firmware takes care of this in conjunction with the PAL. You are not aware of this going on except that you cannot make exclusive use of the timer taking care of this function.

Much of the firmware monitor is tied into the PAL functionality. This means that if you write code to go into the Eprom or EEprom, then you must also include the monitor code. If you don't you may lose function of the ram refresh and of course all of the built–in commands and RS-232 communications. The MONITOR is necessary to handle Ram Refresh, RS-232 communications, etc.

1.6 Firmware Specifications

The firmware included with the 2010 chip takes care of the Ram Refresh and and any new commands added to the Monitor. Refer to the memory map for specific memory locations used.

00000-0FFFF Ram Bank 0

10000-1FFFF Ram Bank 1 (on 256k Models)

20000-2FFFF Ram Bank 2 (on 256k Models)

30000-3FFFF Ram Bank 3 (on 256k Models)

40000-4FFFF Rom Code Memory (read Only)

50000-7FFFF Reserved

80000-8007F Internal Data Memory (800FF 8032)

C0080-C00FF Special Function Register Region

See Code Memory Map In Appendix D

—Notes—

Chapter 2

Installation and Setup

2.1 Unpacking

When unpacking the model 2010 board, be sure to watch for items such as jumpers, disks, cables, and instructions and/or errata sheets while you are unpacking. Many phone calls have been made in the past, because in the haste of unpacking and getting to the main board, material that was thought to be packing material was protecting a disk or instruction manuals. These materials should be plainly marked as "instructions" or "disk", but some people don't take the time to read it and discard it. If you think you are missing anything from your order, please be sure to go back through the packing material to make sure that it was not accidentally discarded.

2.2 Installation

The 2010 board (depending on the options ordered) generally comes set up to plug in and run. Most will be set up with the machine language monitor on a 2764 installed with jumpers set for an 8k Eprom. No other option jumpers will be put on the board.

2.2.1 Quick Start

To begin communicating immediately with the board follow these instructions:

1—Plug in RS-232 cable to computer and 2010 board. (or make cable from instructions in appendix D).

2—Plug wall transformer into board.

3—Plug wall transformer into 120 Volt outlet.

4—Run CPX communications program.

5—Issue commands.

Remember that if you don't have the CPX communications program that you will have to set the communications parameters first on your computer to communicate with the board. The 2010 is set up to be a DCE device. This means that on an IBM PC or AT type computer (that

has a DTE port) the cable will run straight through. (see appendix D for hookup)

2.2.2 Normal Installation

After unpacking the 2010, set it up for your use. Normally, it will already be installed to use with the machine language monitor. However, you may install the board in this manner:

1—If you didn't get a cable from us, make an RS-232 cable like this:

a) The cable required is a straight through cable. Pin 2 on the computer hooks to pin 2 on the 2010. Hook up pins 1, 2, 3, 4, 5, 6, 7, and 20. 8052AH BASIC does not require hardware handshaking, so you could just use pins 2, 3 and 7 for a cable.

b) The computer end of the cable will require a female connector, while the 2010 end requires a male connector.

2—Hook the RS-232 cable to the 2010 and the computer.

3—Check the jumpers for operation. A normal first time operation will al ready have the jumpers in the correct location, however check the jumpers as follows:

JB5

—1 is /EA of the processor pulled high by a 2K RP (for internal program memory).

—2 is Ground (for external program memory access)

Default: pin 1 grounded.

JB6

—1 is Vcc. (used for 2764/27128 /pgm pin.)

—2 is pin 27 of the program memory socket (common)

—3 is A14 of the Address Bus

—4 is /WR from the processor

Defaults:

1—2 for 2764,27128 (Default for our monitor 2764A)

4—2 for 2864, 28256 to allow /WR to program EEprom.

3—2 for 27256, 27512 to allow A14 on pin 27 (A14).

JB7

—1 is Vcc. (used for 2764/27128/27256 Vpp pin.)

—2 is pin 1 of the program memory socket (common)

—3 is A15 of the Address Bus

—4 is A14 of the Address Bus

Defaults:

1—2 for Vcc to Vpp of 2764,27128,27256 (Normal for our 2764A)

4—2 for 28256 to allow A14 onto pin 1 (A14)

3—2 for 27512 to allow A15 onto pin 1 (A15—requires special PAL)

Memory (Usable)	JB5	JB6	JB7	Type:
2764/A 8K Eprom	1-2	2-1	2-1	Machine Language
2864 8K EEprom	1-2	2-4	2-1	Machine Language
27128/A 16K Eprom	1-2	2-1	2-1	Machine Language
27256 32K Eprom	1-2	2-3	2-1	Machine Language
28256 32K EEprom	1-2	2-4	2-4	Machine Language
27512 32K Eprom	1-2	2-3	2-3	ML (32K max only)

JB5 controls external access of program code fetches. If it is jumpered /EA is grounded, forcing the processor to fetch code externally from the Eprom. If you have a 8751 or 87C51 you should leave pin 1 and 2 of JB5 open (none) to fetch code internally from the Eprom.

JB6 controls where pin 27 of the program memory socket (Eprom or EEprom) connects. Pin 27 connects to pin 2 of the jumper block. If pin 2 is jumpered to pin 1 (Vcc) then that pin is held at Vcc for a 2764. To pin 3 will connect line A14 to pin 27 for a 27256. To pin 4 will connect /WR to pin 27 for for /WE for 8K EEproms, Xicor X2864A for example.

JB7 controls where pin 1 of the program memory socket (Eprom or EEprom) connects. Pin 1 of the program memory socket connects to pin 2 of the jumper block. If pin 2 is jumpered to pin 1 (Vcc) then that pin is held at Vcc for a 2764. To pin 3 will connect line A15 to pin 1 for 27512. To pin 4 will connect line A14 to pin 1 for a Xicor X28256 for example.

4—If you are using the Wall Transformer, plug it into the miniature phone jack just to the bottom of the DB-25 connector. If you are using

another external power supply, the be sure that you don't exceed 9 Volts input. If you do, make sure that the extra power dissipation from the 7805 is taken care of. A 12 volt power supply may require additional heat sinking or forced air cooling of the heat sink of the 7805 regulator.

On the Miniature phone jack connector the tip is + and the ring is —. There is a diode in series with the line going to the regulator to prevent reverse current flow, so if the board does not operate, check for power reversal.

If you are supplying a regulated +5 volts from an external power supply, unsolder and remove the 7805 regulator from the board. Hook the +5 volts to either the expansion connector or the hole where the output pin of the 7805 was, and the ground to either the expansion plug or the middle leg hole of the 7805.

There is a place for a reset button located at SW1 on the PC board. Use a SPST or SPDT momentary contact PC mount switch, if you need reset.

—NOTES—

Chapter 3

Using The Monitor

3.1 Commands

Commands on the 2010 running under the monitor begin with one letter. An sssss means that this is up to a 24 bit start address for the command. If there is also an eeeee that means it is a 24 bit ending address to be used with the command. A list of commands and an explanation of their function follows. There are some examples.

A "cr" means carriage return, and usually means to hit the enter key, especially if it is in "greater than" and "less than" brackets, like <cr> (0Dh). A "lf" means line feed (0Ah). A <sp> or <space> means the space bar (20h).

3.1. 1 ' ' (SPACE command)

The SPACE command causes a cr/lf to occur before executing the command that follows on the line. It has no apparent effect on any operation otherwise.

Example:

<monitor> **<sp>U0,2<cr>**

(extra cr/lf here)

000000 0000 00 ...

<monitor>_

3.1. 2 'cr' (Carriage Return command)

The CARRIAGE RETURN command has the effect of reissuing the command prompter if there are no other commands present.

Example:

<monitor> **<cr>**

<monitor> **U0,3<cr>**

000000 0000 0000 ....

<monitor>_

3.1. 3 'lf' (Line Feed command)

The LINE FEED command has the effect of issuing a line feed on screen. The <monitor> prompt is not reissued.

Example:

<monitor> **<lf>**

<lf>

<lf>

U0,4<cr>

000000 0000 0000 00 .....

3.1. 4 ! (Ram Test command)

The ! RAM TEST command allows you to test your ram in operation. Error messages indicate the nature of the fault. A period is sent to the console for every page of ram that is checked. The Ram Test runs until someone presses any key on the console.

Example:

<monitor> **!**

Ram test - press any key to terminate

.....**<sp>**

<monitor>

3.1. 5 ':' (Input Intel Hex)

The INPUT INTEL HEX command allows you to input Intel Hex code to the console until an End Record is reached or a Syntax error (*SN err @aaaaa) or Data error (*DT err @aaaaa) is reached. No keystrokes after (:<cr>) are echoed to the console. This command is primarily used to obtain executable programs into the ram.

Example:

<monitor> **:<cr>**

<monitor>

Note that no characters were echoed to the console even though you could have loaded a 32K machine language file into ram.

3.1. 6 Asssss (ASSEMBLE command)

The ASSEMBLE command causes 8031 mnemonics to be converted to 8031 op codes in memory at the indicated addresses. You must be specific with jmps and calls, for instance you must use LJMP and LCALL or AJMP and ACALL.

Example: Assemble at 10000 then abort:

<monitor> **a10000<cr>**

010000 **< cr>**

<monitor>_

Example: Assemble at 0 then abort:

<monitor> **a0<cr>**

```
000000    LJMP   E803<cr>
000003    AJMP   000E<cr>
000005    LJMP   FB00<cr>
000008    MOV    R2,A<cr>
000009    MOV    R7,A<cr>
00000A    MOV    R7,A<cr>
00000B    LJMP   FB09<cr>
00000E    JMP    FB03<cr>
000011    MOV    R7,A<cr>
000012    MOV    R7,A<cr>
000013    LJMP   FB06<cr>
000016    MOV    R7,A<cr>
000017    MOV    R7,A<cr>
000018    MOV    R7,A<cr>
000019    MOV    R7,A<cr>
00001A    MOV    R7,A<ESC>  \
```

3.1. 7 Esssss<cr> (ENTER command)

The ENTER command allows you to enter bytes into the ram in a formatted fashion at the specified starting address (sssss). Any location from 0000H to FFFFH may be ENTERed. Remember that you could overwrite any resident programs you have with this command, so be careful, especially above 0E000H. If you strike a carriage return <cr> at the prompt, <monitor> will issue a cr/lf and display the next address. If you enter data <ff>, 2 spaces are issued and the data at the next address is displayed. The display is done in this manner until you hit a <cr>, when a new line is given with the address and data displayed. Any time an ESC is hit, you are taken back to <monitor>. Data is not altered until you either strike return or space on a single valid ascii hex character, or enter 2 valid ascii hex characters. Characters you type are in bold.

Example:

<monitor> **E0<cr>**

00000 02–**ff** E8–**ff** AA–**aa** BB–**< sp>** CC–**< sp>** DD–**ff** EE–**< cr>**

00007 E8–**< cr>**

00008 FB–**< cr>**

00009 EC–**9< sp>** AA–**< cr>**

0000A EC–**< ESC>** \

<monitor>_

3.1. 8 Gsssss<cr> (GO command)

The GO command will begin execution at the specified start address (sssss). If no start address (sssss) is given, then execution is begun at the address specified by the source address within the Intel Hex file just loaded, or 0.

Example:

<monitor> **G<cr>**
(program executes)

If you had just loaded a hex file that specified the starting address in the end record, then you will begin executing at that specified start address. Otherwise, it will begin where you specify with Gssssss.

Example:

<monitor> **G1234<cr>**

The above will begin execution at 1234H. Make sure you have some code to execute there.

3.1. 9 I<cr> (Issue command)

The Issue command will cause the prompter to be issued.

3.1.10 Fsssss,eeeee,dd

The FILL command will fill Ram with the specified 8 bit data (dd) between and including the starting address (sssss) through and including the ending address (eeeee).

Example:

<monitor> **F20,2E,FF<cr>**

<monitor> **L20,2F<cr>**

00000 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FF00 .....

<monitor>_

3.1.11 Lsssss,eeeee<cr> (LIST command)

The LIST command will cause the contents of sssss-eeeee inclusive to be displayed on the console in a formatted ascii-hex dump. If the form Lsssss<cr> or L<cr> is used, then only 128 bytes are LISTed to the screen.

Example:

<monitor> **l<cr>**

```
01010 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
01020 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
01030 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
01040 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
01050 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
01060 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
01070 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
01080 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
```

128 bytes listed to the screen in this example. The IP was pointing to something other than 0 when the LIST command was issued.

<monitor> **l2095<cr>**

```
02095                FF 0000  FFFF 0000 FFFF 0000     ..........
020A0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
020B0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
020C0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
020D0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
020E0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
020F0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
02100 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
02110 000 0  FFFF  00                                    .....
```

128 bytes are LISTed to the screen in this example. The IP was set to 2095 by giving the start address (sssss).

<monitor> **l,21ff<cr>**

(NOTE: L, will give current pointer address)

02115 00 FFFF 0000 FFFF 0000 FFFF .....

02120 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

02130 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

02140 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

02150 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

02160 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

02170 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

02180 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

02190 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

021A0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

021B0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

021C0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

021D0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

021E0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

021F0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

More than 128 bytes were given in this example because the end address (eeeee) was given, and the IP address (2115 at the time) was used for the start address (sssss).

<monitor> **L21F0,21FF<cr>**

021F0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

<monitor>

3.1.12 Msssss,eeeee,ttttt (Move command)

The MOVE command moves a block of data from the starting address (sssss) through the ending address to the target address (ttttt).

Example:

<monitor> **L8000,801F<cr>**

08000 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

08010 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF .....

<monitor> **M0,1f,8000<cr>**

<monitor> **L0,1f<cr>**

08000 AA55 AA55 AA55 AA55 AA55 AA55 AA55 AA55 .....

08010 0102 0304 0506 0708 090A 0B0C 0D0E 0F10 .....

<monitor> **L8000,801F<cr>**

08000 AA55 AA55 AA55 AA55 AA55 AA55 AA55 AA55 .....

08010 0102 0304 0506 0708 090A 0B0C 0D0E 0F10 .....

3.1.13 OIsssss,eeeee (Output Intel hex file command)

The Output Intel command will cause an Intel Hex file to be listed to the console. This command is usually used in conjunction with CPX or a modem program to obtain the object code from Ram.

Example:

<monitor> **OI0,f<cr>**

:1000000000FF00FF00FF00FF00FF00FF00FF00FF0D

:0000000000

<monitor>_

3.1.14 Psssss (PROGRAM command)

The PROGRAM command will cause the data beginning at start address (sssss) to be altered by the ascii hex characters following the delimiter after the start address. A valid delimiter is a space, cr, lf, comma or dash. A PERIOD character will cause the command to terminate. A P<delim> will cause replacement to begin at 0.

Any time the command line buffer is overrun (entering more than 78 characters) or a *SN error or *DT error occurs, (aborting you back to the <monitor> prompter), you must enter your data again from the <cr/lf> for that data to be entered into the ram.

Example:

<monitor> **P,aa55aa55aa55<cr>**

aa55aa55aa55aa55<cr>

aa55aa55aa55aa55<cr>

aa55 <more than 78 chars on this line> **aa55-beep**

(Here, where you entered more than 78 characters, you should begin again from where you left off before the beep. If you're not sure, then List until you find where you left off at the error)

<monitor>_

3.1.15 Rsssss,eeeee (READ command)

The READ command will cause the data beginning at the start address (sssss) up to and including the end address (eeeee) to be output to the console in a stream of ascii-hex characters. If R<cr> is done, then 0-7Fh is output. There are no other characters other than 0-9 and A-F output with this command. Warning- Remember that the source pointer may not be set to 0 when you issue R<cr> and the output gives you no indication of where this code is ac tually being read from, so R0<cr> is a safer command to use if you want to read starting at 0!!

Example:

<monitor> **R<cr>**

0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000
FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF
0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000
FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF
0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF
<monitor>_

<monitor> **R0,F<cr>**

01AA5A010E02FB00FAFFFF02FB0902FB

<monitor> **R40000,4000F<cr>**

01AA5A010E02FB00FAFFFF02FB0902FB

<monitor>_

3.1.16 Usssss,eeeee (UNASSEMBLE command)

The UNASSEMBLE command will "unassemble" code starting at the start address (if specified sssss) up to and including the end address (if specified eeeee). Otherwise a U<cr> will unassemble up to 26 bytes of code.

Example: Unassemble

```
<monitor>  u0<cr>
000000     LJMP   E803
000003     AJMP   000E
000005     LJMP   FB00
000008     MOV    R2,A
000009     MOV    R7,A
00000A     MOV    R7,A
00000B     LJMP   FB09
00000E     LJMP   FB03
000011     MOV    R7,A
000012     MOV    R7,A
000013     LJMP   FB06
000016     MOV    R7,A
000017     MOV    R7,A
000018     MOV    R7,A
000019     MOV    R7,A
00001A     MOV    R7,A
<monitor>  u0,A<cr>
000000     LJMP   E803
000003     AJMP   000E
000005     LJMP   FB00
000008     MOV    R2,A
000009     MOV    R7,A
00000A     MOV    R7,A
<monitor>_
```

3.2 Using The U And A Commands

The Unassemble and the Assemble commands are used to assemble and unassemble code in the Ram on a 2010. By using this feature, it may eliminate the need for you to assemble off board and then download for every little change you might make in your code, like patches for instance.

See appendix A for special function register names that can be used with the assembler/disassembler.

Chapter 4

System Function Calls

Function calls may be made to the operating system of the 2010. In general a character is put into the accumulator and then address 5 is called with an LCALL.

0 to 07FH Direct Console Output of the Accumulator

A function call to 5 with the accumulator loaded with the character to be sent will cause the character in the accumulator to be output to the console serial line.

0E3H Return address of start, end, target

Call 0E3h will return the address of a 9 byte location that holds the 24 bit source, 24 bit end, and 24 bit target addresses. This command is usually used for commands like Move sssss, eeeee, ttttt. The accumulator will be pointing at the source address, and the end address will be at source + 3 and the target address will be at source + 6.

0E4H Display Error Message

Call 0E4h displays an error message and returns to the monitor. If B=0 then "SNERR" is displayed, B=1 is "DTERR", B=2 is "CSERR", B=3 is "STERR" and B>3 is "UNKNOWN ERROR".

0E5H Get 24 bit number

Call 0E5h will get a 24 bit number from ascii-hex numbers in the system command line buffer and puts them at the address pointed to by R0 in the internal ram. Returns CY set if it was terminated with a <cr>; if it was terminated with something other than a valid delimter (space, cr, lf, comma, dash), it terminates by issuing *SN err and returns to the monitor (not to you).

0E6H Delete command from monitor

Call 0E6h will delete a command from the monitor. B is the command name. ACC is equal to 0 if successful and 0FFh if it failed. DPTR

returns the address that the command occupied, so you can actually "rename" any command of the system monitor by immediately making a function call number E7. See following example.

0E7H Add command to monitor

Call 0E7h will add a command to the system monitor. On entry, B is the legal command name. DPTR is the execution address. Control-Q (17), Control-S (19), 0, 255 and rom based command names are illegal.

Example:

```
MOV     B,#'L'      ;let's rename "L" command to be "D"
MOV     A,#0E6H     ;function call to delete command
LCALL   5           ;this returns address in DPTR
                    ;otherwise you would load the address
                    ;of your routine in DPTR
MOV     B,#'D'      ;rename to be 'D'
MOV     A,#0E7H     ;function call to add command
LCALL   5           ;now 'L' is 'D'
```

0E8H get binary from ascii in input queue

Call 0E8h will get a binary byte from 2 ascii hex characters from the console input queue. The two characters must be consecutive, but may be preceded by any number of valid delimiters. Returns carry set if ok, and clear if there is a data error. This procedure destroys R2 and R6 and the byte is returned in the accumulator.

0E9H binary to ascii output to console

Call 0E9h will output the accumulator to the console as 2 ascii hex characters.

0EAH get command line

Call 0EAh will get a 78 character command line into the system buffer. No input parameters are needed. This function works like function 0FAh, but needs no input or output parameters.

0EBH reset command line pointer

Call 0EBh will reset system command line pointer to beginning of the current line (used with 0EAh and 0ECh).

0ECH Get character from system buffer

Call 0ECh is to get a character from the system command line buffer or return a 0 if none is available. This function destroys DPTR, and the character is returned in the accumulator. This function also increments the pointer to the next character on the command line. You can use EB to point back at the beginning.

0EDh RESERVED.

0EEH Return number of bytes in queue

Call 0EEh will return the number of bytes in the output queue including any currently being transmitted. A 0 returned in the accumulator means there is nothing in the queue and nothing is being transmitted.

0EFH Return last character received

Call 0EFh will return the last character received at the console input or 0 if there is none available. This function call is similar to FF except you can get the same character over and over again.

0F0H Terminate and return

Call 0F0h is terminate program and return to the monitor. No parameters are necessary.

0F1H wait console input with echo

Call 0F1h is wait for console input with echo. Character is echoed to the console and is contained in the accumulator upon return.

0F2H output B character to console

Call 0F2h is output character in the B register to the console. The character could be 0 - 0FFh.

0F3h thru 0F8h RESERVED.

0F9H output external memory string

Call 0F9h is output string from external memory to the console, pointed to by DPTR. The string must terminate with a dollar sign ($). If your string is in the range of 0-7FFFH and you want the program to be in Eprom then you must use your own routine to MOVC from code

memory. Addresses from 8000- FFFFH are considered to always be code memory.

0FAH get command line

Call 0FAh is get command line. On entry, DPTR points to the user defined buffer. The first location of the buffer must be loaded by the user to indicate the maximum number of characters the buffer may hold. On return, the second character indicates the number of characters that were entered (including the return). The third and all following locations up to the end of the buffer are the characters entered. You must strike a carriage return (cr) for the function call to terminate. Line editing is limited to a backspace. An ESC will cause a backslash to be printed immediately and the function call is terminated and returned to the calling routine with only a cr in the buffer.

0FBH console status check

Call 0FBh is console status check. Returns 0 in the accumulator if nothing is available. (same as 0FEh)

0FCH get command line

Call 0FCh is get command line. On entry, DPTR points to the user defined buffer. The first location of the buffer must be loaded by the user to indicate the maximum number of characters the buffer may hold. On return, the second character indicates the number of characters that were entered (including return if entered). The third and all following locations up to the end of the buffer are the characters entered. The function call is automatically terminated when the end of the buffer is reached, or if carriage return (cr) is struck. Similar to function call 0FAh except reaching the end of the buffer automatically terminates the command. Line editing is limited to a backspace. An ESC will cause a backslash to be printed immediately and the function call is terminated and returned to the calling routine with only a cr in the buffer.

0FDH direct console input

Call 0FDh is direct console input. This function call waits for a character to come from the console and the character is returned in the accumulator. It does not echo the character to the console.

0FEH direct console status check

Call 0FEh is direct console status check. This routine returns 0 in the accumulator if no character is available.

0FFH direct console input

Call 0FFh is for direct console input. This function call returns an ascii character in the accumulator or 0 if none is available. It does not echo to the console and does not wait for a character to be available

—Notes—

—Notes—

Chapter 5

Communications Software

5.1CPXINSTALLATION

On your disk you have 2 programs, CPX.COM and PINSTALL.COM. To install CPX to run on your computer, first run PINSTALL. PINSTALL will show you a list of choices to make for Baud Rate and COM line. Choose the correct selec tion and exit PINSTALL. A typical choice would be for 9600 Baud on COM1: or COM2:. CPX will then be ready to use.

5.2USINGCPX

To use CPX, run it and you will see a short preamble telling you how CPX is installed. After that it tries to communicate with your board. Error messages might tell you about framing errors or break the first time you run it during the day, but it will always log on if you have your cable connected and the board running already.

Most of the time, CPX is simply a communications program. However, by pressing CONTROL–F during communication, you can enter a filename to send a file to the 2010. A CONTROL–C or an ESC will get you back to your system. Syntax is as follows (after CONTROL–F):

<monitor> **^F**

Enter Command —>**filename [option,option,option< cr>**

no extension (.HEX) is necessary or

Enter Command —>**[option,option,option<cr>**

Options are:

R—means read an intel hex file to disk from memory

that you are specifying the starting and ending address from ram to send to the Intel Hex file on disk. SSSS means start address (in hex) and EEEE means end address (in hex.)

EXAMPLE:

Enter Command —>**test<cr>**

The above will look for a file called test.hex on the disk, and when found, send it to the 2010. The 2010 will load the file to the memory locations specified in the hex file in Ram.

Enter Command —>**test [r,@200-7FF< cr>**

The above will try to make a file called test.hex on disk (and issue an error message if one currently exists) and then copies the contents of Ram from 0200 to 07FF in Intel Hex format to the file. The only extension that will be created is .HEX.

5.3 Automatic Production Of An Eprom With Autoexecute

Using Version 1.15 or later, you may make an eprom that will automatically execute your program contained in Eprom (from 2000–7FFFh). Assuming that you have a GTEK Eprom programmer, include the file called M2010R7 in the Eprom. This can be done automatically with a Batch file. Create it with the examples below. The name of the batch file will be BURNME.BAT and the name of your file to burn into the Eprom will be MYFILE.HEX or whatever your filename is + .HEX.

Remember that your code must be originated at 2000h and the first byte of that code MUST be A5. Your executable code should begin at 2001h. Code can be from 2000h–7FFFh

Example Batch File called burnme.bat:

```
PGMX   M2010R7  [M1,V,TN
PGMX    %1   [V,TN]
rem Done
```

and to run it:

C:\> **BURNME MYFILE**

—OR—

```
PGMX   M2010R7
PGMX   %1
```

and to run it:

C:\> **BURNME MYFILE**

You can modify the default baud rate in your monitor so that it will boot to a certain baud rate on power–up by modifying the byte at location 0008h in the eprom. This location is used the same way in ram at location 0008h.

Location 8H Baud Rate Divisor:

Baud	Data	Baud	Data
57,600	0FFh	28,800	0FEh
19,200	0FDh	9,600	0FAh
4,800	0F4h	2,400	0E8h
1,200	0D0h	600	0A0h

—Notes—

Chapter 6

2010M RS–232 Interface

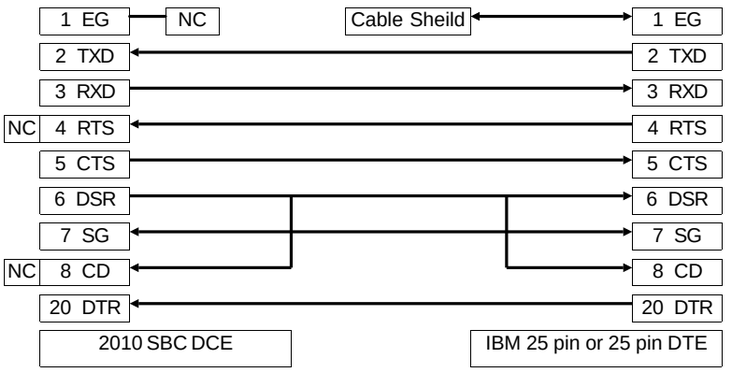
The model 2010 has a DB25S connector configured as Data Communications Equipment (DCE).

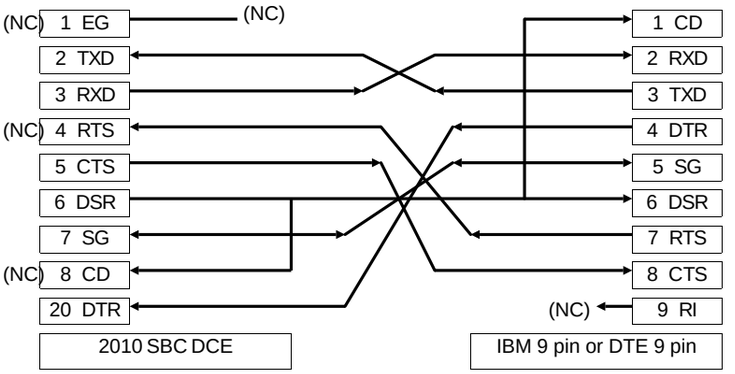
The meanings of the pins used on the 2010 DB25S connector is as follows:

Pin#	Direction	Function
1—(EG)	<—>	Equipment Ground. Not hooked up.
2—(TXD)	<—	Transmit Data. Data input to processor.
3—(RXD)	—>	Receive Data. Data output from processor.
4—(RTS)	<—	Request To Send. Not hooked up.
5—(CTS)	—>	Clear To Send. Output handshake. Normally at +12 volts. Goes to —12 volts when 2010 wants you to stop sending.
6—(DSR)	—>	Data Set Ready. Always at +12 volts when power is applied to the 2010.
7—(SG)	<—>	Signal Ground.
8—(CD)	—>	Carrier Detect. Not hooked up.
20–(DTR)	<—	Data Terminal Ready. Input handshake. +12 volts to allow 2010 to send, —12 volts to stop 2010 from sending.

2010 (Male DCE) to IBM PC/XT/AT DB25 (female DTE)
Gtek part number RSFDCE

2010 ..... Pin #	Pin # ..... IBM
EG (nc) 1	1 Cable Sheild . EG
TXD (i) 2	2 ..... (o) TXD
RXD (o) 3	3 ..... (i) RXD
RTS (nc) 4	4 ..... (o) RTS
CTS (o) 5	5 ..... (i) CTS
DSR/CD (nc) 6/8	6/8 ..... (i) DSR/CD
SG (i/o) 7	7 ..... (i/o) SG
DTR (i) 20	20 ..... (o) DTR





AT DB9 (male) to 2010 DB25 (female)

2010 (DCE) . . . Pin #	Pin # AT (DTE 9 pin)
EG 1	
TXD (i) 2	3 . . . . . (o) TXD
RXD (o) 3	2 . . . . . (i) RXD
RTS 4	7 . . . . . (o) RTS
CTS 5	8 . . . . . (i) CTS
DSR/CD. 6/8	6/1 . . . . . (i) DSR/CD
SG 7	5 . . . . . (i/o) SG
DTR 20	4 . . . . . (o) DTR

Remember that on the 2010, pin 5 (CTS) is an OUTPUT handshake since the DB connector is configured as Data Communication Equipment. This line goes low when the 2010 wants you to stop sending. Pin 20 (DTR) is an INPUT handshake. When this line goes low, the 2010 will stop sending until DTR goes high again.

Remember that any TERMINAL you use may not use the "standard" RS–232 pin configuration if you are having trouble handshaking.

Appendix—A

Special Function Register Names

Both the Assembly and Unassembler commands make use of the names for the Special Function Registers and the Special Function Bits. A list will follow with examples. The Name is recognized by the assembler/Unassembler. The address is where this register appears in the internal ram. An asterisk (*) means that the register is both byte and bit addressable. A plus (+) means that this register is available only on the 8052 type chips.

Special Function Reg. Recognized by Assembler/Unassembler

Name	Function	Address (HEX)	Notes
ACC	ACCumulator	E0	*
B	B Register	F0	*
PSW	Program Status Word	D0	*
SP	Stack Pointer	81	
DPH	Data Pointer High byte	83	
DPL	Data Pointer Low byte	82	
P0	Port 0 (AD0-AD7)	80	*
P1	Port 1 (P1.0-P1.7)	90	*
P2	Port 2 (A8-A15)	A0	*
P3	Port 3 (P3.0-P3.7)	B0	*
IPC	Interrupt Priority Control	B8	*
IEC	Interrupt Enable Control	A8	*
TMOD	Timer/counter Mode control	89	
TCON	Timer/counter control	88	*
T2CON	Timer/counter 2 Control	C8	+*
TH0	Timer/counter 0 High byte	8C	
TL0	Timer/counter 0 Low byte	8A	
TH1	Timer Counter 1 High Byte	8D	
TL1	Timer Counter 1 Low Byte	8B	
TH2	Timer Counter 2 High Byte	CD	+
TL2	Timer Counter 2 Low Byte	CC	+
RCAP2H	TC2 Capture Register High	CB	+
RCAP2L	TC2 Capture Register Low	CA	+

SCON . . . . Serial Control Register . . . . 98 . . . . . .*
SBUF . . . . . Serial Data Buffer 99
PCON . . . . Power CONtrol 87

* Means both bit and Byte addressable
+ Means available only on 8052

See the data book for the 8031 for explanations of usage of these registers.

Breakdown Of Individual Registers. The Symbol column is the name of the bit in question. Position is the name and bit position of the main register where the bit name resides. Significance explains what the bit is used for. You should refer to the processor data sheet for more information about the registers and bits:

Symbol	Position	Significance
CY	PSW.7	Carry Flag
AC	PSW.6	auxiliary carry flag for bcd operations
F0	PSW.5	flag 0, available for general purposes
RS1	PSW.4	reg. bank select control bits 1 and 0,
RS0	PSW.3	set/clr to determine working reg. bank
	(rs1=0, rs0=0 is bank 0 - 0-7h)	
	(rs1=0, rs0=1 is bank 1 - 8-fh)	
	(rs1=1, rs0=0 is bank 2 - 10-17h)	
	(rs1=1, rs0=1 is bank 3 - 18-1fh)	
OV	PSW.2	overflow flag
—	PSW.1	Reserved
P	PSW.0	parity flag used to indicate the parity of the contents of the accumulator (even or odd parity)

Breakdown Of Individual Registers:

Symbol	Position	Significance
TF1	TCON.7	timer 1 overflow flag
TR1	TCON.6	timer 1 run control bit
TF0	TCON.5	timer 0 overflow flag
TR0	TCON.4	timer 0 run control bit
IE1	TCON.3	interrupt 1 edge flag
IT1	TCON.2	interrupt 1 type control bit
IE0	TCON.1	interrupt 0 edge flag
IT0	TCON.0	interrupt 0 type control bit

Breakdown Of Individual Registers:

Symbol	Position	Significance
TF2	T2CON.7	timer 2 overflow flag
EXF2	T2CON.6	timer 2 external flag
RCLK	T2CON.5	receive clock flag
TCLK	T2CON.4	transmit clock flag
EXEN2	T2CON.3	timer 2 external enable flag
TR2	T2CON.2	start/stop control for timer 2
C/T2	T2CON.1	timer or counter select - 0 = osc/12
CP/RL2	T2CON.0	capture / reload flag

Breakdown Of Individual Registers:

Symbol	Position	Significance
SM0	SCON.7	serial port mode
SM1	SCON.6	
SM0/SM1	mode description baudrate	
0 0	0—Shift Register Fosc/12	
0 1	1—8 Bit Uart Variable	
1 0	2—9 Bit Uart Fosc/12 Or 32	
1 1	3—9 Bit Uart Variable	
SM2	SCON.5	mult/proc mode in modes 2 and 3
REN	SCON.4	enable serial reception
TB8	SCON.3	9th data bit to be tx by mode 2, 3
RB8	SCON.2	9th data bit received in mode 2 or 3
T1	SCON.1	transmit interrupt flag
R1	SCON.0	receive interrupt flag

Breakdown Of Individual Registers:

Symbol	Position	Significance
EA	IE.7	disable all interrupts
—	IE.6	reserved
ET2	IE.5	en/disable t2 ov or cap interrupt
ES	IE.4	en/disable serial port interrupt
ET1	IE.3	en/disable t1 ov interrupt
EX1	IE.2	en/disable ext1
ET0	IE.1	en/disable t0 ov
EX0	IE.0	en/disable ext0

Breakdown Of Individual Registers:

Symbol	Position	Significance
—	IP.7	reserved
—	IP.6	reserved
PT2	IP.5	define t2 interrupt priority
PS	IP.4	define serial port interrupt priority
PT1	IP.3	define t1 interrupt priority
PX1	IP.2	define ext1 interrupt priority
PT0	IP.1	define t0 interrupt priority
PX0	IP.0	define ext0 interrupt priority

Breakdown Of Individual Registers: (Default 0xxx0000)

Symbol	Position	Significance
SMOD	PCON.7	double baud rate bit
—	PCON.6	reserved
—	PCON.5	reserved
—	PCON.4	reserved
GF1	PCON.3	general purpose flag
GF0	PCON.2	general purpose flag
PD	PCON.1	power down bit
IDL	PCON.0	idle mode bit

Port 3 Breakdown

Symb.	Position	Significance	2010 Name
RXD	P3.0	receive data (serial input port)	RXD
TXD	P3.1	transmit data (serial output port)	TXD
/INT0	P3.2	external interrupt 0	/DTR
/INT1	P3.3	external interrupt 1	/CTS
T0	P3.4	timer 0 external input	/CMM
T1	P3.5	timer 1 external input	/ROM
/WR	P3.6	external data memory write strobe	/WR
/RD	P3.7	external data memory read strobe	/RD

Port 2 Breakdown

Symbol	Position	Significance
A15	P2.7	upper address bus
A14	P2.6	
A13	P2.5	
A12	P2.4	
A11	P2.3	
A10	P2.2	
A09	P2.1	
A08	P2.0	

Port 1 Breakdown (alternate)

Symbol	Position	Significance
—	P1.7	Used for general
—	P1.6	purpose I/O on the
—	P1.5	2010M board, on
—	P1.4	P132–P139
—	P1.3	
—	P1.2	
T2EX	P1.1	T2EX on 8052 only
T2	P1.0	T2 on 8052 only

Appendix B

2010 Board Connector Pinouts

(note: refer to data sheets for processor, etc. as needed)

Expansion Bus 34 Pin Connector On 2010 Board:

AD0	1	34	AD1
AD2	2	33	RA8
NC	3	32	AD3
NC	4	31	AD4
NC	5	30	AD5
NC	6	29	AD6
VCC	7	28	AD7
VCC	8	27	A10
NC	9	26	ALE
A15	10	25	—PSEN
GND	11	24	—ROM
GND	12	23	A14
—CMM	13	22	A13
—WR	14	21	A12
A9	15	20	—RD
A11	16	19	A8
VDD	17	18	VDD

NOTES:

AD0-AD8 are pins 39-32 of the processor.

A8-A15 are pins 21-28 of the processor.

NC means No Connection – this is for prototyping.

—CMM is a line from the processor port 3.4 (pin 14) that goes low for external Reads and Writes to the I/O Map instead of the Ram.

—ROM is a line from the processor port 3.5 (pin 15) that goes low and causes PSENs to go only to the Ram.

ALE, PSEN, —RD, —WR all come from the processor. (30, 29, 17, 16)

VCC is regulated +5 volts DC from the on-board 7805 regulator.

VDD is unregulated +9 volts DC from D2 and C21.

GND is the system ground.

RA8 is a signal output from the PAL for 256Kb Rams.

40 PIN CONNECTOR ON 2010 BOARD

Port #	I/Oaddr	Pin #	Pin #	I/Oaddr	Port #
Port1	P1.6	01	40	P1.7	Port 1 on U2 8031
	P1.4	02	39	P1.5	
	P1.2	03	38	P1.3	
	P1.0	04	37	P1.1	
Port7	P82.6	05	36	P82.7	on U3 Z80–PIO this is port B
	P82.4	06	35	P82.5	The 82 refers to the address
	P82.2	07	34	P82.3	in memory map plus the bit
	P82.0	08	33	P82.1	number on the port.
Port6	P80.6	09	32	P80.7	on U3 Z80–PIO this is port A
	P80.4	10	31	P80.5	
	P80.2	11	30	P80.3	
	P80.0	12	29	P80.1	
Port5	PA2.6	13	28	PA2.7	on U4 Z80–PIO this is port B
	PA2.4	14	27	PA2.5	
	PA2.2	15	26	PA2.3	
	PA2.0	16	25	PA2.1	
Port4	PA0.6	17	24	PA0.7	on U4 Z80–PIO this is port A
	PA0.4	18	23	PA0.5	
	PA0.2	19	22	PA0.3	
	PA0.0	20	21	PA0.1	

P1.0 – P1.7 is PORT1 on the 8031 U2. P80.0 – P80.7 refers to the Z80–PIO U3 Port A data bits. The 80 refers to the memory map address 80xxH where you can address the A data port. The control port would be 81xxH. The 82 refers to the memory map address 82xxH where you can address the B data port. Control is 83xxH.

The .7 in P80.7 refers to the bit number 7 in that data port. PA0.7 – PA0.0 is the Z80–PIO U4 port A. PA2.7 – PA2.0 refers to the Z80–PIO U4 port B.
P139 – P132 corresponds to P1.7 through P1.0 on U2. P131 – P124 corresponds to U3 port B bits 7–0. P123–P116 corresponds to U3 port A bits 7–0. P115–P108 correspond to U4 port B bits 7–0. P107–P100 correspond to U4 port A bits 7–0.

Jumper blocks JB1 – JB4 allow you to ground certain pins of the 40 pin site (P2):

JB1—12

JB2—21

JB3—40

JB4—NC (for prototyping)

—NOTES—

—NOTES—

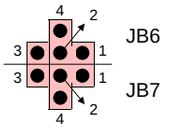
Appendix C

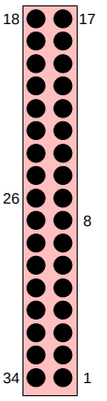
Jumpers Used on the 2010 Board

jumper	pin #	function	
JB1	1	pin 12 of P1	default: no jumper
	2	ground	
JB2	1	pin 21 of P1	default: no jumper
	2	ground	
JB3	1	Pin 40 of P1	default: no jumper
	2	ground	
JB4	1	nc (for prototyping)	default: no jumper
	2	ground	
JB5	1	U2 pin 31 —EA. ground for ext. access.	
	2	ground	default: jumper 2–1
JB6	1	Vcc	default: jumper 2–1
	2	U1 pin 27 —PGM, —WE or A14.	
	3	U2 pin 27 (A14)	
	4	U2 pin 16 (—WR)	
JB7	1	Vcc	
	2	U1 pin 1 Vpp or A14 or A15.	
	3	U2 pin 28 (A15)	default: jumper 2-1
	4	U2 pin 27 (A14)	

—Notes—

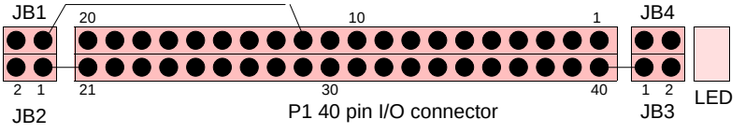
Hookup for 2764A or 27128A JB6 hook 1 to 2 JB7 hook 1 to 2
Hookup for 27256 JB6 hook 2 to 3 JB7 hook 1 to 2





P2 34 pin
Expansion Interface connector

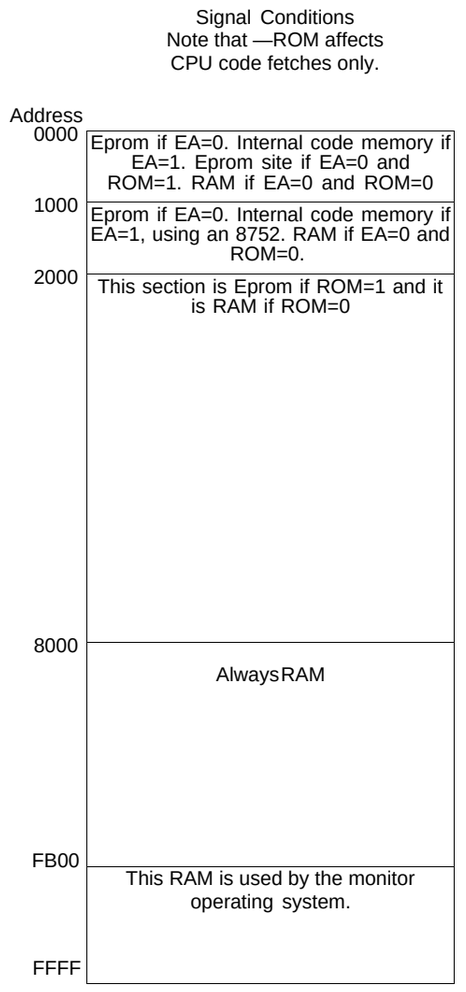
For 2764–27256 hook 1 to 2 For 8751 no connection

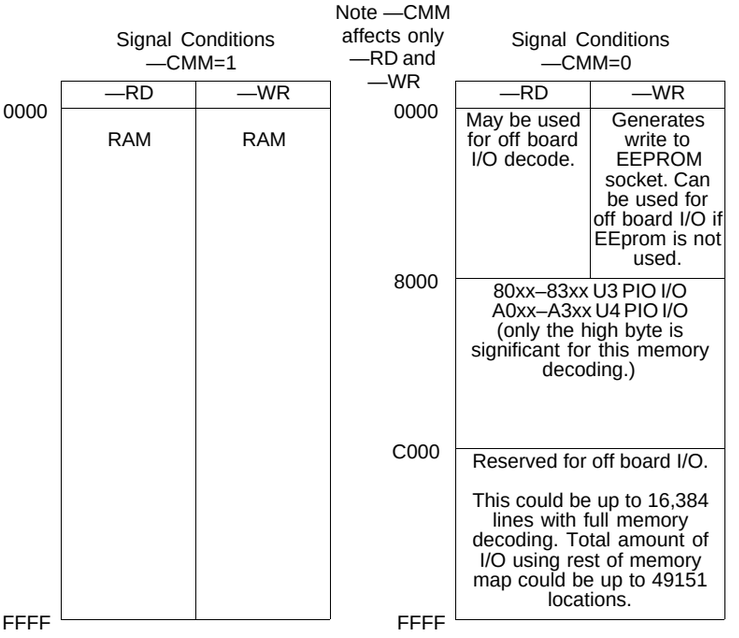


—Notes—

Appendix D—Memory Maps

Code Memory Usage (PSENs)

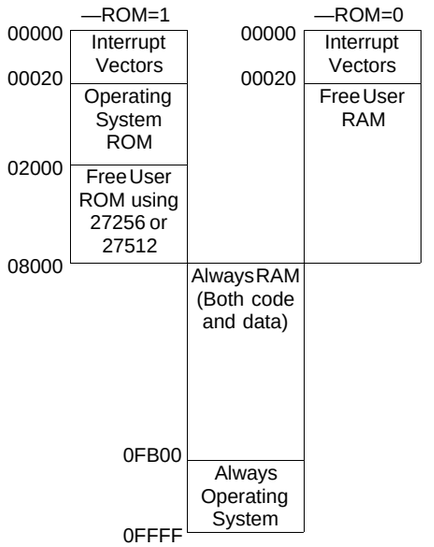




Example:

```
;Routine to demonstrate RAM reads and I/O reads. Assume CMM=1
        Mov     DPTR,#1234h
        Movx    A,@DPTR     ;Get what is in RAM at 1234h
        Clr     CMM         ;select I/O
        Mov     DPH,#80h    ;we don't care whats in DPL
        Movx    A,@DPTR     ;Get data from Z80–PIO U3
        Setb    CMM         ;select ram
;done
```

Code Memory



External 64k Bytes Of Ram

The external ram is used in different ways depending on what condition the —ROM line is in. Be careful not to use from FB00 to FFFFh and from 0 to 20h.

Interrupt Vectors are in the first 20H bytes of the memory map in either RAM or ROM. These Vectors jump into the code starting at location FB00H. If you need to insert your own vector to use a Timer/Counter or external interrupt, you can modify the locations at FB00 up jumped to by the code in either Rom or Ram at 0- 20H. Modify locations above FB00 so that your code can be in either Ram or Rom. Be really careful about TIMER1 (Ram Refresh).

Vectors used by the 2010M in low/high memory

Low	High	Function
0000	FB00	Monitor Entry Vector
0003	FB03	EXT0 interrupt vector
0005		jmp into OS (call with reg. loaded for funct. calls)
000B	FB09	TIMER0 interrupt vector
0013	FB06	EXT1 interrupt vector
001B	FB0C	TIMER1 interrupt vector
0023	FB12	serial interrupt vector
002B	FB0F	timer2 interrupt (8032 only)

Timers and Counters

TIMER0 is set up to run in mode 3, as two 8 bit timers.

TH0 of TC0 is used for Ram Refresh.

TL0 of TC0 can be used as a timer or counter for the user.

TIMER1 runs as the baud rate generator and is "disconnected" from the interrupt structure.

Autoexecute

The operating system checks the byte at 2000H to see if there is a user program that needs to run first (AUTOEXECUTE). If 2000H = A5 then the program located at 2001H executes from the Eprom. Program termination (RET) or function call F0 causes a return to the monitor. If the byte at 2000H is not A5 the monitor automatically begins execution. This requires Version 1.15 firmware or later.

INTERNAL 128 BYTES OF RAM

	0 7	Register Bank 0 (0 through 7). May be used by the user. If the Monitor is entered, however, these registers are usually destroyed.
	8 F	Register Bank 1 (8 through F). May be used by the user. These registers are not used by the monitor unless you use the "A" or "U" commands.
	10 17	Register Bank 2 (10 through 17). May be used by the user. Not used by the monitor
	18 1F	Register Bank 3 (18 through 1F). May be used by the user. Not used by the monitor
	20 2E	Internal RAM available to the user.
	2F	Serial Status Byte
	30	Serial Input Buffer
	31 35	Internal RAM available to the user.
	36 37 38	Used with Function calls 0E5, 0E6. Contains the Start Address. Used only by monitor commands that "get" addresses.
	39 3A 3B	Used with Function calls 0E5, 0E6. Contains the End Address. Used only by monitor commands that "get" addresses.
	3C 3D 3E	Used with Function calls 0E5, 0E6. Contains the Target Address. Used only by monitor commands that "get" addresses.
	3F	Timer Register used for RAM Refresh
	40	Always Reserved.
	41 7F	User Stack.

—Notes—

—Notes—

Appendix E

Pin Names For The 8031 and Z80–Pio Chips.

8031 MPU U2

Pin#	2010 Name	8031 Name/Function
1	P1.0	P1 pin 37 pulled up with 2.2k RP7–B
2	P1.1	P1 pin 04 pulled up with 2.2k RP7–C
3	P1.2	P1 pin 38 pulled up with 2.2k RP7–D
4	P1.3	P1 pin 03 pulled up with 2.2k RP7–E
5	P1.4	P1 pin 39 pulled up with 2.2k RP7–F
6	P1.5	P1 pin 02 pulled up with 2.2k RP7–G
7	P1.6	P1 pin 40 pulled up with 2.2k RP7–H
8	P1.7	P1 pin 01 pulled up with 2.2k RP7–I
9	RESET	RST 100ms reset by SW1,C3,D1,R1. SW1 is spst momentary.
10	—RXD	P3.0 receive data thru u6/12 from db25/2 (txd)
11	—TXD	P3.1 send data thru U6/10 to DB25/3 (RXD)
12	—DTR	P3.2 handshake out thru u6/11 to DB25–5 (CTS). also can be used for external interrupt by cut/add jumpers.
13	—CTS	P3.3 handshake in thru U6/9 from DB25–20(DTR).
14	—CMM	P3.4 see memory map
15	—ROM	P3.5 see memory map
16	—WR	P3.6 write signal to RAM, PAL, and I/O
17	—RD	P3.7 read signal to RAM, PAL, and I/O
18	XTAL1	crystal input from XTAL and C2
19	XTAL2	crystal output to XTAL, C2 and Z80-PIOs (clock)
20	GND	Vss system ground

(Continued next page)

Pin#	2010 Name	8031 Name/Function
21	A8	P2.0 upper address line A8
22	A9	P2.1 upper address line A9
23	A10	P2.2 upper address line A10
24	A11	P2.3 upper address line A11
25	A12	P2.4 upper address line A12
26	A13	P2.5 upper address line A13
27	A14	P2.6 upper address line A14
28	A15	P2.7 upper address line A15
29	PSEN	PSEN to U1 (eprom) and U7 (PAL) for memory accesses
30	ALE	ALE (address latch enable) to U6 (for A0-A7) and RAMs (for —RAS)
31	—EA	ground (JB5—1–2) for external eprom, open for 8751.
32	AD7	P0.7 address/data bit 7 pulled up with RP6-I, 2.2K SIP
33	AD6	P0.6 address/data bit 6 pulled up by RP6-H
34	AD5	P0.5 address/data bit 5 pu by RP6-G
35	AD4	P0.4 address/data bit 4 pu by RP6-F
36	AD3	P0.3 address/data bit 3 pu by RP6-E
37	AD2	P0.2 address/data bit 2 pu by RP6-D
38	AD1	P0.1 address/data bit 1 pu by RP6-C
39	AD0	P0.0 address/data bit 0 pu by RP6-B
40	VCC	Vcc +5 volt system power

Z80-PIOU3

1	AD2	data bit 2 (D2)
2	AD7	data bit 7 (D7)
3	AD6	data bit 6 (D6)
4	A14	—CE chip enable must be low to select chip
5	A8	CD (C NOT D) high = control, low=data
6	A9	BA (B NOT A) high = Port B, low = Port A
7	P80.7	U3 Port A data I/O bit 7 to pin 32 of P1
8	P80.6	bit 6 to pin 09 Port A pulled high with
9	P80.5	bit 5 to pin 31 a 2.2K resistor pack.
10	P80.4	bit 4 to pin 10
11	GND	system ground
12	P80.3	U3 Port A data I/O bit 3 to pin 30 of P1
13	P80.2	bit 2 to pin 11
14	P80.1	bit 1 to pin 29
15	P80.0	bit 0 to pin 12
16	—ASTB1	Port A strobe, pulled up by RP1–E
17	—BSTB1	Port B strobe, pulled up by RP1–G
18	ARDY1	Port A ready, nc
19	AD0	data bit 0 (D0)
20	AD1	data bit 1 (D1)

(Continued on next page)

21	BRDY1	Port B ready, nc
22	IEO1	interrupt enable output, nc
23	—DTR	—INT (used when programmed only). notice that this is rs232 handshake line. you must cut appropriate jumper when using —INT.
24	IEI1	interrupt enable input, hooked to RP1–H and IEI2
25	XTAL2	clock input pin
26	VCC	system power +5 V
27	P82.0	Port B data I/O bit 0 to pin 08 of P1
28	P82.1	bit 1 to pin 33 Port B pulled high with
29	P82.2	bit 2 to pin 07 a 2.2K resistor pack.
30	P82.3	bit 3 to pin 34
31	P82.4	bit 4 to pin 06
32	P82.5	bit 5 to pin 35
33	P82.6	bit 6 to pin 05
34	P82.7	bit 7 to pin 36
35	—RD	—RD
36	—IORQ1	—IORQ must be low to select chip (from PAL)
37	M11	M1 Pulled High By R5 2.2K resistor
38	AD5	data bit 5 (D5)
39	AD4	data bit 4 (D4)
40	AD3	data bit 3 (D3)

Z80-PIOU4

1	AD2	data bit 2 (D2)
2	AD7	data bit 7 (D7)
3	AD6	data bit 6 (D6)
4	A14	—CE chip enable must be low to select chip
5	A8	CD (C NOT D) high = control, low=data
6	A9	BA (B NOT A) high = Port B, low = Port A
7	PA0.7	Port A output data I/O bit 7 to 24 on P1
8	PA0.6	bit 6 to pin 17 Port A pulled high with
9	PA0.5	bit 5 to pin 23 a 2.2K resistor pack.
10	PA0.4	bit 4 to pin 18
11	GND	system ground
12	PA0.3	Port A output data I/O bit 3 to 22 on P1
13	PA0.2	bit 2 to pin 19
14	PA0.1	bit 1 to pin 21
15	PA0.0	bit 0 to pin 20
16	—ASTB1	Port A strobe, pulled up by RP1–D
17	—BSTB1	Port B strobe, pulled up by RP1–F
18	ARDY1	Port A ready, nc
19	AD0	data bit 0 (D0)
20	AD1	data bit 1 (D1)

(Continued on next page)

21	BRDY1	Port B ready, nc
22	IEO1	interrupt enable output, nc
23	—DTR	—INT (used when programmed only). notice that this is rs232 handshake line. you must cut appropriate jumper when using —INT.
24	IEI2	interrupt enable input, hooked to RP1–H and IEI1
25	XTAL2	clock input pin
26	Vcc	system power +5 V
27	PA2.0	Port B data I/O bit 0 on pin 16 on P1
28	PA2.1	bit 1 to pin 25
29	PA2.2	bit 2 to pin 15
30	PA2.3	bit 3 to pin 26
31	PA2.4	bit 4 to pin 14
32	PA2.5	bit 5 to pin 27
33	PA2.6	bit 6 to pin 13
34	PA2.7	bit 7 to pin 28
35	—RD	—RD system read signal
36	—IORQ1	—IORQ must be low to select chip (from PAL)
37	M11	M1 Pulled High By R5 2.2K resistor
38	AD5	data bit 5 (D5)
39	AD4	data bit 4 (D4)
40	AD3	data bit 3 (D3)

Controlling the U3 PIO

(port 6, P116–123 and port 7, P124–131)

(XX MEANS DON'T CARE WHAT THE NUMBER IS.)

Chip U3 is addressed at:

80xxH data port A

81xxH control port A

82xxH data port B

83xxH control port B

A "write" to external memory at those locations causes the data to be output to this Z80–PIO (U3) to those bits that are programmed for output.

A "read" from external memory at these locations cause the data to be read from the output latch (if a bit is programmed for output) or the output data lines if a bit is programmed for input.

All the outputs are pulled high with a 2K resistor pull up pack.

Controlling the U4 PIO

(port 4, P100–107 and port 5, P108–P115)

(XX MEANS DON'T CARE WHAT THE NUMBER IS.)

Chip U4 is addressed at:

A0xxH data port A

A1xxH control port A

A2xxH data port B

A3xxH control port B

See the note above on U3 writing and reading.

```
;Example Program Subroutines
;demonstration to show how to write to the Z80-PIO ; lines.
;DDR means Data Direction Register. 0= output ; 1= input
;DPH high byte of DPTR
;0FFh puts the Z80-PIO into control mode.
PRGM:   MOV     DPH,#81H    ;address control reg port a (81xx)
        MOV     A,#0FFH     ;data to u3, z80-pio into control mode
        MOVX    @DPTR,A     ;put z80-pio into control mode
                            ;(WRITE FF TO 81XX)
        MOV     A,#0F0H     ;BYTE TO CONTROL I/O=BIT0–3 OUTPUT,
                            ;BIT4–7, INPUT, WHICH IS PINS 15–12 FOR
                            ;BIT 0–3, AND PINS 10–7  FOR BIT 4–7
        MOVX    @DPTR,A     ;DEFINE I/O
        MOV     DPH,#83H    ;SAME FOR PORT B ON U3
        MOV     A,#0FFH
        MOVX    @DPTR,A
        MOV     A,#00       ;ALLOUTPUT
        MOVX    @DPTR,A
        MOV     DPH,#0A1H   ;CONTROL U4 PORT A
        MOV     A,#0FFH
        MOV     @DPTR,A
        MOV     A,#0        ;ALLOUTPUT
        MOVX    @DPTR,A
        MOV     DPH,#0A3H   ;CONTROL U4 PORT B
        MOV     A,#0FFH
        MOV     @DPTR,A
        MOV     A,#0AAH     ;MAKE EVERY OTHER PIN OUTPUT, INPUT
                            ;10101010 IN/OUT/IN/OUT/IN/OUT/IN/OUT
        MOVX    @DPTR,A
        RET                 ;REM THEY STAY THIS WAY UNTIL YOU
                            ;REPROGRAM
```

Now you can read and write to the data register at 80xx, 82xx, A0xx, A2xx. When you read a data register, pins programmed as inputs reflect the condition of the pins, and pins programmed as outputs reflect the latches of the port. Next, you can use the following subroutines to read and write to the ports.

```
readp:  clr     t0          ;set CMM to proper position
        mov     dph,#80h    ;data port
        movx    a,@dptr     ;get data from port
        setb    t0          ;t0 is synonym for p3.4
        mov     b,a         ;store data
        mov     a,#e9h      ;Display data at console, converted to ascii
                            ;from binary (function call)
        lcall   5           ;execute. shows data in ascii hex format
                            ;on your console, read from the specified port
        ret
writep: mov     dph,#80h    ;write only pins defined as output
        clr     p3.4        ;are affected. p3.4=CMM
        movx    @dptr,A     ;write
        setb    p3.4        ;done, return
        ret
```

Remember that when you are writing code with the built in assembler that labels are not allowed. Also, when using numbers, you cannot put a "H" (like 80H) after the hex number. The assembler assumes any number you are using is hex and not decimal or octal.

—Notes—

—Notes—

—Notes—

—Notes—

—Notes—

—Notes—

—Notes—

Model 2010 Silkscreen

—Notes—