Table of Contents

Chapter1

Introduction

1.1 Hardware Overview

The 2020 has 32 programmable I/O lines. Fourteen of those are general purpose quasi–static I/O lines from the 8032 itself – Port 1 has P1.0–P1.7 for 8, and from Port 3 you have P3.0–P3.5 for 6. You then have 8 dedicated input lines with user latch control (–HOLD, input to U10) and 8 dedicated output lines with user output enable (–BEN, input to U6).

In addition to those 32 I/O lines, are lines to be used to expand the functions of the 2020 board. AD0–AD7, ALE, RD, WR, A8–A12, A13/Reset (selectable), User I/O Enable, , Vcc and Gnd. These are brought to a 20 pin expansion bus (JP2). 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 new board could just piggy back on top of the 2020.

The 2020 can optionally have a built in 5 volt regulated power supply. It needs to be connected to a single ended power supply capable of at least the power requirement of the type chips used on the 2020. A bare board with some NMOS parts draws about 150Ma. If you attach expansion boards to it, you must use an adequate power supply. The RS–232 Option uses a MAX–232 chip to obtain the +–12 volts for the RS–232 supply.

With the proper power supply options, 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: 3.5 x 3.75 x .5 inches
89 x 95 x 12.5 millimeters

Weight: 2 Oz.
62 grams

Power Requirements:
"Bare Board" with NMOS parts and a 27C128 – About 150ma

Direct I/O Lines Programmable As Input Or Output:
14
Direct Input Lines with user latch:
8
Direct Output Lines with user hold:
8

Memory Mapped I/O using the -UIO signal:

Read at 00000–03FFFH is from U6 (depends also on –HOLD)

Write at 00000–03FFFH is to U10 (depends also on –BEN)

PROCESSORS: 8032 at 15 MHz (standard)

(Also see your processor specification sheet)

1.3 Software Overview

Much of the software for the 2020 using the monitor is strictly communications software. That is, you could use other programs to communicate with the 2020, but our software will recognize upload, download and other commands and handle them accordingly.

Most of the commands are handled by the 2020 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 CPX2.COM. A program called RINSTALL is provided to install it for the Baud Rate or COM port you are using. Use CPX2 to upload and download programs in Intel Hex format and to communicate with the 2020.

1.4 Software Specifications

CPX2 Communicates at from 1200 Baud through 19,200 Baud. It has commands to read and save Intel Hex files, specify load addresses, and load/execute a program.

1.5 Firmware overview

The built in monitor allows you to communicate with the outside world using the RS–232, 422 or 485 lines. The monitor provides an in–line assembler and system function calls. This will allow you to write simple assembler programs as well as download and execute programs and store them on disk.

1.6 Firmware Specifications

The firmware included takes care of board house-keeping with a "debug–like" shell, and any new commands added to the Monitor. Refer to the memory map for specific memory locations used.

Memory Map

Program (Eprom/Ram)

00000–0FFFFh

Data (RAM)

08000–0FFFFh

I/O (Memory Mapped, –UIO signals a write/read)

Reads: 00000–03FFFh input from U6 (depending on –HOLD)

Writes: 00000–03FFFh output from U10 (depending on –BEN)

User I/O:

04000–07FFFh –UIO signal on JP1–19 signals reads/writes to external I/O.

—NOTES—

Chapter 2

Using The Monitor

2.1 Commands

Commands on the 2020 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).

2.1.1 ' ' (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>U8000,8002<cr>
(extra cr/lf here)
008000 0000 00                 ...
<monitor>_
```

2.1.2 '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>U8000,8003<cr>
008000 0000 0000               ....
<monitor>_
```

2.1.3 '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>
U8000,8004<cr>
008000 0000 0000 00              .....
```

2.1. 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>
```

2.1. 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.

2.1. 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 8000h then abort:
```
<monitor>a8000<cr>

008000 LJMP  E803<cr>
008003 AJMP  000E<cr>
008005 LJMP  FB00<cr>
008008 MOV   R2,A<cr>
008009 MOV   R7,A<cr>
00800A MOV   R7,A<cr>
00800B LJMP  FB09<cr>
00800E JMP   FB03<cr>
008011 MOV   R7,A<cr>
008012 MOV   R7,A<cr>
008013 LJMP  FB06<cr>
008016 MOV   R7,A<cr>
008017 MOV   R7,A<cr>
008018 MOV   R7,A<cr>
008019 MOV   R7,A<cr>
00801A MOV   R7,A<ESC>\
```

2.1. 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 8000H 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>E8000<cr>
08000 02-ff E8-ff AA-aa BB-<sp> CC-<sp> DD-ff EE-<cr>
08007 E8-<cr>
08008 FB-<cr>
08009 EC-9<sp> AA-<cr>
0800A EC-<ESC>\
<monitor>_
```

2.1. 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 Gsssss.

Example:
```
<monitor>G8234<cr>
```

The above will begin execution at 8234H. Make sure you have some code to execute there.

2.1. 9 I<cr> (Issue command)

The Issue command will cause the prompter to be issued.

2.1.10 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>F8020,802E,FF<cr>
<monitor>L8020,802F<cr>
08020 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FF00 ................
<monitor>_
```

2.1.11 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>
08010 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
08020 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
08030 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
08040 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
08050 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
08060 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
08070 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000 ................
08080 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>lA095<cr>
0A095             FF 0000 FFFF 0000 FFFF 0000       ...........
0A0A0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
0A0B0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
0A0C0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
0A0D0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
0A0E0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
0A0F0 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
0A100 FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000  ................
0A110 0000 FFFF 00                             .....
```

128 bytes are LISTed to the screen in this example. The IP was set to A095 by giving the start address (sssss).

```
<monitor>l,A1ff<cr>
```

(NOTE: L, will give current pointer address)

```
0A115             00 FFFF 0000 FFFF 0000 FFFF       ...........
0A120 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A130 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A140 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A150 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A160 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A170 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A180 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A190 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A1A0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A1B0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A1C0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A1D0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A1E0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF  ................
0A1F0 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 (A115 at the time) was used for the start address (sssss).

```
<monitor>LA1F0,A1FF<cr>
0A1F0 0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF ................
<monitor>
```

2.1.12 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>M9000,901f,8000<cr>
<monitor>L8000,801f<cr>
09000 AA55 AA55 AA55 AA55 AA55 AA55 AA55 AA55 ................
09010 0102 0304 0506 0708 090A 0B0C 0D0E 0F10 ................
<monitor>L9000,901F<cr>
09000 AA55 AA55 AA55 AA55 AA55 AA55 AA55 AA55 ................
09010 0102 0304 0506 0708 090A 0B0C 0D0E 0F10 ................
```

2.1.13 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>OI8000,800f<cr>
:1080000000FF00FF00FF00FF00FF00FF00FF00FF0D
:0080000000
<monitor>_
```

2.1.14 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.

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>P8000,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>_
```

2.1.15 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. There are no other characters other than 0–9 and A–F output with this command. Warning– Remember that the source pointer is not set properly unless you specify an address between 8000 to FFFFh.

Example:

```
<monitor>R8000<cr>
0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FF
FF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000
FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF00
00FFFF0000FFFF0000FFFF0000FFFF0000FFFF0000FFFF
<monitor>_
<monitor>R8000,800F<cr>
01AA5A010E02FB00FAFFFF02FB0902FB
<monitor>_
```

2.1.16 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>u8000<cr>
008000 LJMP  E803
008003 AJMP  000E
008005 LJMP  FB00
008008 MOV   R2,A
008009 MOV   R7,A
00800A MOV   R7,A
00800B LJMP  FB09
00800E LJMP  FB03
008011 MOV   R7,A
008012 MOV   R7,A
008013 LJMP  FB06
008016 MOV   R7,A
008017 MOV   R7,A
008018 MOV   R7,A
008019 MOV   R7,A
00801A MOV   R7,A
<monitor>u8000,800A<cr>
008000 LJMP  E803
008003 AJMP  000E
008005 LJMP  FB00
008008 MOV   R2,A
008009 MOV   R7,A
00800A MOV   R7,A
<monitor>_
```

2.2 Using The U And A Commands

The Unassemble and the Assemble commands are used to assemble and unassemble code in the Ram on a 2020. 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.

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
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

Symbol	Position	Significance	2020 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	P3.2
/INT1	P3.3	external interrupt 1	/CTS
T0	P3.4	timer 0 external input	/DTR
T1	P3.5	timer 1 external input	P3.5
/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	2020 board, on
–	P1.4	JP1, pins 7–14
–	P1.3	
–	P1.2	
T2EX	P1.1	T2EX on 8052 only
T2	P1.0	T2 on 8052 only

Chapter 3

System Function Calls

Function calls may be made to the operating system of the 2020. In general a byte (function call number) is loaded 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.

```
mov    A,#0e3
lcall  5
```

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".

```
mov    A,#0e4
lcall  5
```

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,#4C      ;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,#44      ;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.

```
mov    A,#0e8
lcall  5
```

0E9H Binary to ascii output to console

Call 0E9h will output the (binary) byte in the "B" register 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 0EBh to point back at the beginning.

0EDh is 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 character to console

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

```
mov    b,#41
mov    a,#0f2
lcall  5
```

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—

Chapter 4

Communications Software

4.1 CPX2 Installation

On your disk you have 2 programs, CPX2.COM and RINSTALL.COM. To install CPX to run on your computer, first run RINSTALL. RINSTALL will show you a list of choices to make for Baud Rate and COM line. Choose the correct selection and exit RINSTALL. A typical choice would be for 9600 Baud on COM1: or COM2:. CPX2 will then be ready to use.

4.2 Using CPX2

To use CPX2, run it and you will see a short preamble telling you how CPX2 is installed. After that it tries to communicate with your board.

Most of the time, CPX2 is simply a communications program. However, by pressing CONTROL–F during communication, you can enter a filename to send a file to the 2020. 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

In the following examples where SSSS and EEE are specified, remember 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 2020. The 2020 will load the file to the memory locations specified in the hex file in Ram.

Enter Command – > **test [r,@8200–87FF <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 8200 to 87FF in Intel Hex format to the file. The only extension that will be created is .HEX.

4.3 Automatic Production Of An Eprom With Autoexecute

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 M2020R9 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   M2020R9  [M1,V,TN
PGMX    %1    [V,TN]
rem Done
```

and to run It:

C:\>**BURNME MYFILE**

– OR –

```
PGMX   M2020
PGMX   %1
```

and to run it:

C:\>**BURNME MYFILE**

– Notes –

Chapter 5

You may apply power to the board in several different ways. First, to use 5v you can apply +5v to pin 39 of JP2 (20x2) or pin 18 of JP1 (10x2) and ground (GND) to either pin 40 of JP2 or pin 20 of JP1. Optionally you may add U11 (7805 or 78L05) a voltage regulator—following good engineering practices for power drain and heat dissapation—and then apply up to +30v to pin 38 of JP2 and ground.

You can use the board in many different ways. There are a number of options that affect the way you use the board, so jumpers will be listed first in numerical order.

JP1—This is a 10 pin by 2 (20 total) that allows you to connect to various signals direct to the processor like an expansion port. It contains AD0–7, A8–A12, A13 or reset, read, write, ale and user I/O control plus Vcc and Gnd. The end near the "J" in JP1 is Pin 1.

JP2—This is a 20 pin by 2 (40 total) that allows you to connect to your I/O signals, RS–232, 422, 485, etc. See the schematic for details. The end near the "J" is Pin 1.

JP3—This is the Enable Outputs jumper. If you put a jumper on this one, it permanently enables output from the 8 bit output port on JP2 pins 19–26. If you leave off the jumper you can cause output by controlling JP2 pin 27, the -BEN signal.

JP4—If you use the DS3695 chip, cut the trace between JP4–2 and JP4–3 to allow its use. The jumper pins are not normally installed for this option, since most will probably use DS3696. See JP6 also. The end near the "J" is pin 1.

JP5—Eprom Select. Jumper as marked to use a 2764, 27128 or 27256. The end near the 5 in JP5 is pin 1. One to two is for 2764 or 27128 and two to three is 27256 or 27512 (1/2).

JP6—Install a jumper (or wire if no pins present) to use DS3695 chip instead of DS3696.

JP7—Allows use of JP1–11 as either another address line (A13) or board RESET (RST). It comes jumpered for RST, so cut the jumper between 2 and 3 and install pins (if none are installed) or a jumper to enable A13/RST. Pin 1 is the end near the writing JP7. One to two enables A13—two to three (default before jumper is cut) enables RST—on JP1–11.

JP8—Controls RS–232 power feature. To enable extremely low power operation feature, cut the trace between the pads of JP8. The power option for the RS–232/485 must be present, and will enable you to turn off the power going to the MAX232 or DS3696 chips by writing data to P3.5 on the processor. P3.5 high will disable power, low will turn on power.

JP9—When using RS–422 or RS–485, jumpering this pin will allow Full Duplex communication.

OPTIONS: RS–232:

On a "bare board", you may or may not have to install the following parts for RS–232: C15, C16, C17, C18 (1MF @16 volts) and MAX232 chip. P3.0 becomes -TXD, P3.1 becomes -RXD, P3.3 becomes -CTS, P3.4 becomes -DTR with the GTEK MONITOR OPTION. Of course your code may or may not make use of any of these pins. If you want to use the POWER SWITCH OPTION, see that section for use.

OPTIONS: RS–485 or RS–422:

On a "bare board", you may or may not have to install the following parts for RS–485 or RS–422: U2, U12 (DS3696 or DS3695– see jumpers JP4, JP6 and JP9), R7 and R8 (both 120 ohms 1/4 watt). If you want to make use of the POWER SWITCH OPTION, see that section.

OPTIONS: POWER SWITCH (for Communications):

On a board that has a communications option (RS–232,485 or 422), install Q1 (2N2907), R2 (10K), R3 (2.2K) and make sure JP8 is clear and/or the trace between pins 1 and 2 of JP8 is cut. This enables the capability of switching power to the communications circuit to reduce power requirements during idle times. P3.5 on the processor controls Q1.

OPTIONS: POWER REGULATOR:

Install U11 (7805 or 78L05 depending on power requirements) and heat-sink if appropriate.

OPTIONS: 32K RAM:

Install a 62256 (32K x 8bit) RAM chip in U7 socket. A Hitachi HN62256L will provide the lowest power drain.

OPTIONS: GTEK MONITOR:

Install the provided 27C128 or 27C256 chip in the board and follow the provided instructions with the chip.

Memory Map

Program (Eprom/Ram)

00000–0FFFFh

Data (RAM)

08000–0FFFFh

I/O (Memory Mapped, -UIO signals a write/read)

Reads: 00000–03FFFh input from U6

Writes: 00000–03FFFh output from U10 (depending on –BEN)

User I/O: 04000–07FFFh –UIO signal on JP1–19 signals reads/writes to external I/O.

Refer to Schematic – RS-232, RS-485, RS-422 connections:

Connection names refer to Data Terminal Equipment: At the 40 pin header, EIATXD (pin 16) is an output, EIARXD (pin 15) is an input, EIADTR (pin 17) is an output and EIACTS (pin 18) is an input. EIA means these signal levels can range from +10 to –10 volts referenced to ground when used with the RS–232 standard. These pins are inactive when there is no MAX232 or other comm chips present. For RS–485 using U2 (usually half–duplex, transmitting and receiving on the same line using 1 DS3695/6 chip), Transmit + is pin 16 (also called EIATXD) and Transmit – is pin 18 (also called EIARXD). JP9 should be jumpered for half–duplex. R7 is a termination resistor, usually 120 ohms. For RS–422, also add U12 (DS3696) and R8, usually 120 ohms. Receive + is pin 15 and Receive – is pin 17. When using DS3695 chips, cut the jumper for JP4 between 2 and 3 and jumper pins 1 and 2 on JP4. and jumper JP6, pins 1 and 2.

Model 2020 Bill Of Materials

Item	Quantity	Reference	Part
1	2	C1, C3	20 pF ceramic disk Cap
2	2	C2, C4, C10	106 nF Electrolytic Cap
3	6	C5, C6, C7, C8, C9, C14	104nF disk or Monolythic Cap
4		(not used) C11, C12, C13	Not used
5	4	C15, C16, C17, C18	105nF Electrolytic Cap
6	1	D1	1N4148 switching diode
7	1	JP1	Expansion Header (10x2)
8	1	JP2	I/O Header (20x2)
9	1	JP3	External Output Enable Jumper
10	1	JP4	3695/96 Select Jumper
11	1	JP5	EPROM Size Select Jumper
12	1	JP6	3695/96 Select Jumper
13	1	JP7	A13/RST Select Jumper
14	1	JP8	Comm Power Enable Jumper

15	1	JP9	Full Duplex Select Jumper
16	1	LED1	Red LED Power Indicator
17	1	Q1	2N2907 PNP transistor
18	2	R1, R2	10K Ohms ¼ w (Power Option)
19	1	R3	2K Ohms ¼ w (Power Option)
20	2	R4, R5	3,900 Ohms ¼ watt
21	1	R6	470 Ohms ¼ watt
22	2	R7, R8	120 Ohms ¼ w (RS–422/485 Opt)
23	1	RP1	3,900 Ohms ¼ watt SIP
24	1	U1	8032 or 80C32 MPU
25	2	U2, U12	DS3696/95 (RS–422/485 Option)
26	1	U3	MAX232P (RS–232 Option)
27	1	U4	74HCT00
28	1	U5	74HCT02
29	2	U6, U9	74HCT373
30	1	U7	62256L (CMOS RAM Option)
31	1	U8	27C128/C256 (Monitor Option)
32	1	U10	74HCT374
33	1	U11	7805 /L05 (Power Option)
34	1	X1	11MHz MPU crystal

Chapter 6

Memory Maps

The external ram is used in different ways. Be careful not to use from FB00 to FFFFh and from 8000 to 8020h.

Interrupt Vectors are in the first 20H bytes of the memory map in either RAM or ROM, and from FB00h up. The vectors at 20h (rom) jump into the table starting at location FB00H (see below). If you need to insert your own vector to use a Timer/Counter or external interrupt, you can modify the locations at FB00 or 8000–801F, jumped to by the code in Rom at 0–20H.

Vectors used by the 2020 in low/high memory

ROM	Jumps to RAM at:	Jumps to RAM at:	This Interrupt Vector is:
0003	FB03	8003	INT0
000B	FB09	800B	TIMER0
0013	FB06	8013	INT1
001B	FB0C	801B	TIMER1
0023	FB12	8023	SERIAL
002B	FB0F	802B	TIMER2 (8032 only)

Timers and Counters

TIMER0 is not used by the monitor.

TIMER1 is not used by the monitor.

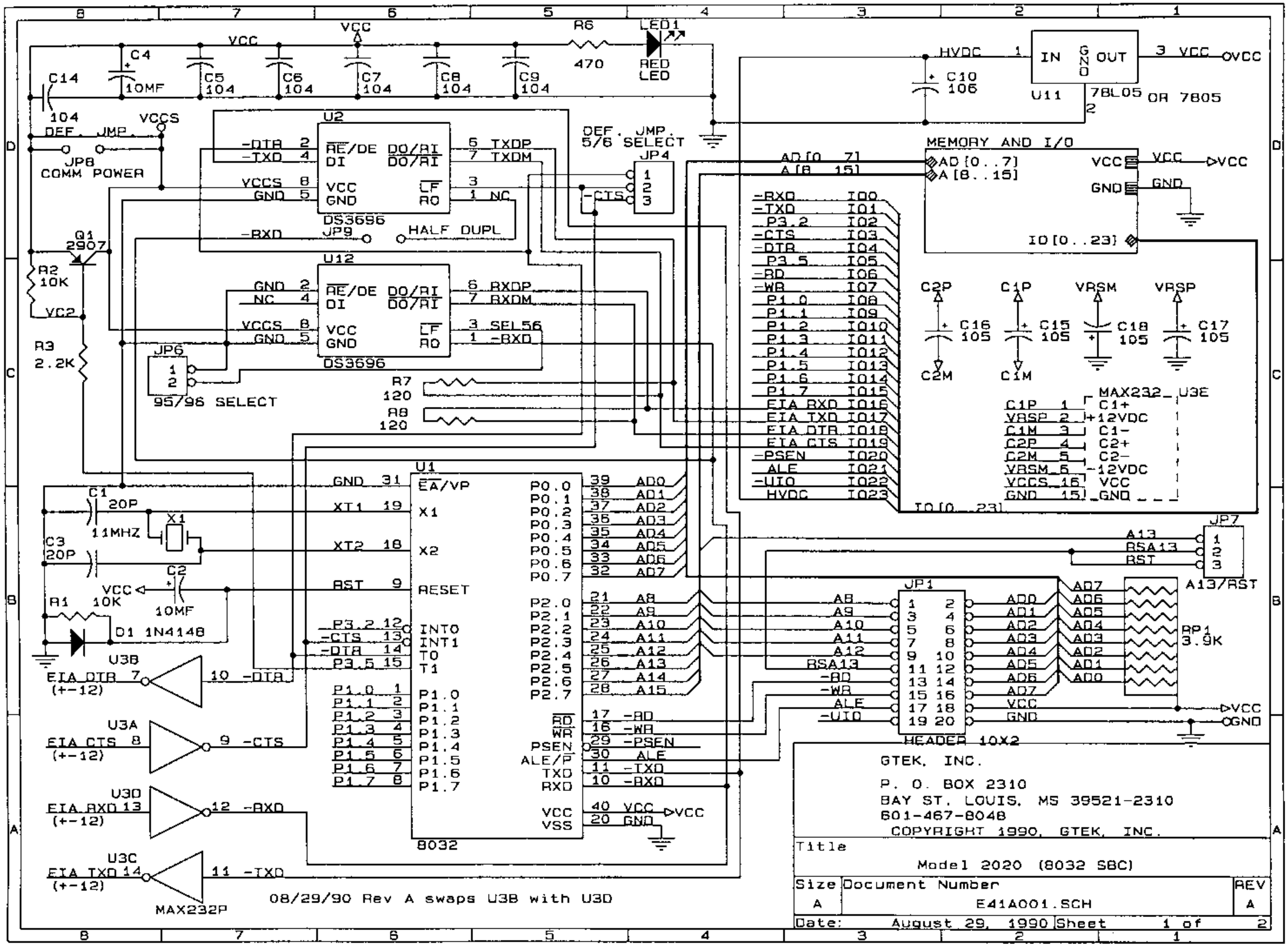
TIMER2 is used for the Baud Rate Generator.

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.

	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	
	40	Always Reserved.
	41 7F	User Stack.

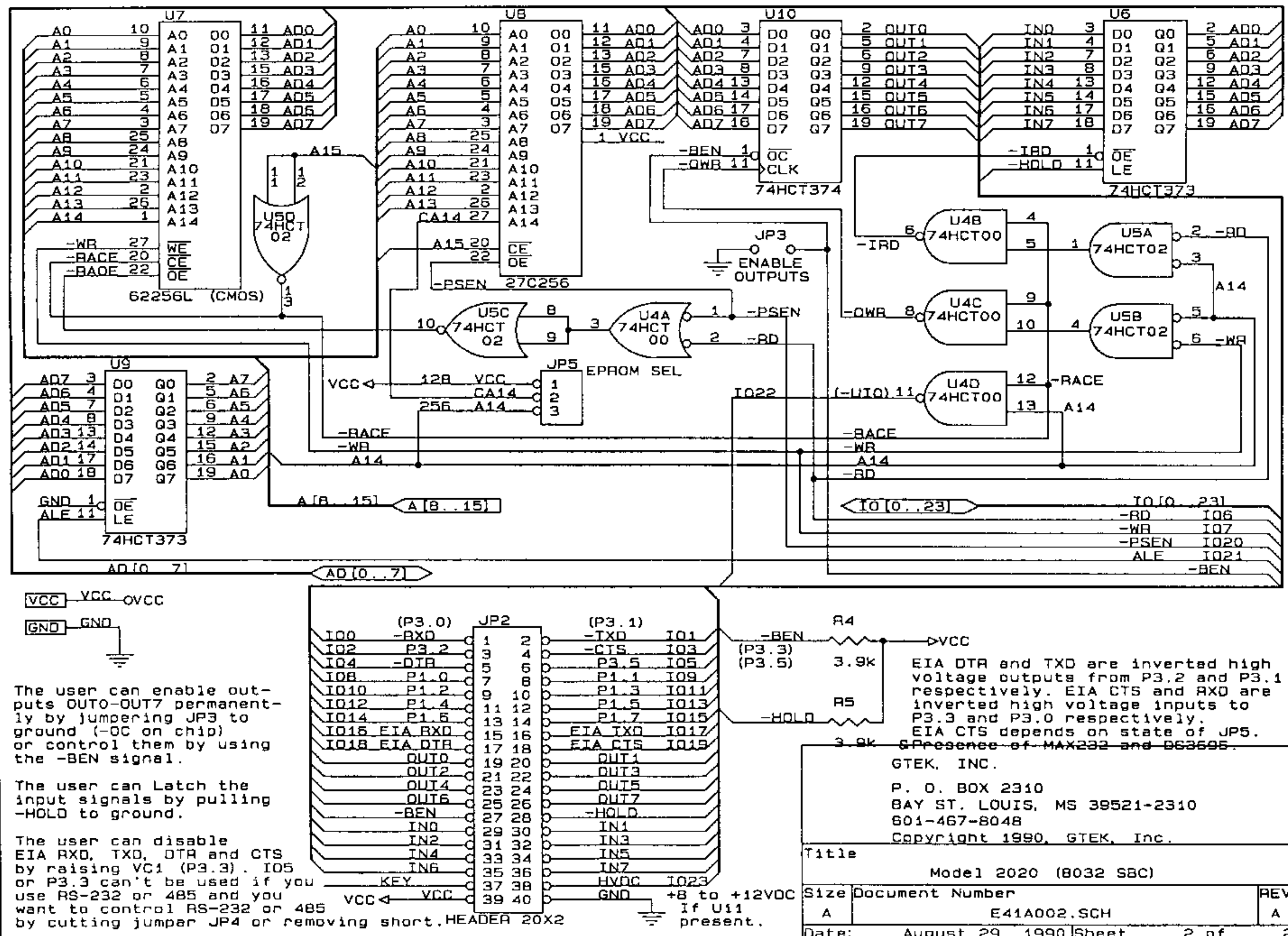
Use of Internal Memory with the GTEK monitor.



GTEK, INC.

P. O. BOX 2310
BAY ST. LOUIS, MS 39521-2310
601-467-8048
COPYRIGHT 1990, GTEK, INC.

Title	Model 2020 (8032 SBC)	
Size A	Document Number E41A001.SCH	REV A
Date: August 29, 1990	Sheet 1 of 2	



EIA DTR and TXD are inverted high voltage outputs from P3.2 and P3.1 respectively. EIA CTS and RXD are inverted high voltage inputs to P3.3 and P3.0 respectively.
EIA CTS depends on state of JP5. ~~&Presence of MAX232 and DS3695.~~

The user can enable outputs OUT0-OUT7 permanently by jumpering JP3 to ground (-OC on chip) or control them by using the -BEN signal.

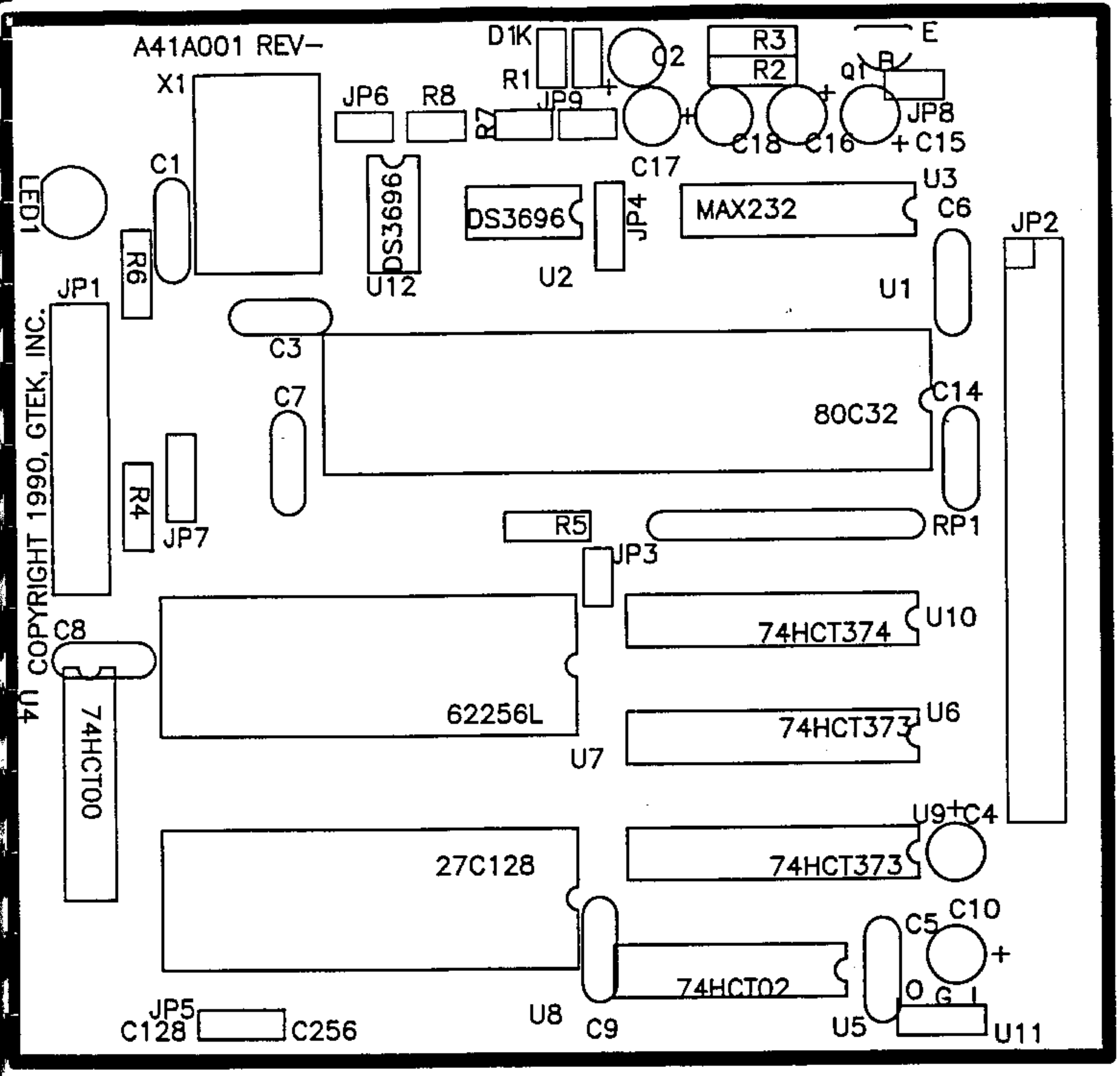
The user can Latch the input signals by pulling -HOLD to ground.

The user can disable EIA RXD, TXD, DTR and CTS by raising VC1 (P3.3). IO5 or P3.3 can't be used if you use RS-232 or 485 and you want to control RS-232 or 485 by cutting jumper JP4 or removing short.

GTEK, INC.

P. O. BOX 2310
BAY ST. LOUIS, MS 39521-2310
601-467-8048
Copyright 1990, GTEK, Inc.

Title		
	Model 2020 (8032 SBC)	
Size	Document Number	REV
A	E41A002.SCH	A
Date: August 29, 1990	Sheet 2 of 2	



COPYRIGHT 1990, GTEK, INC.
A41A001 REV–
ASSEMBLY LAYER
SILK SCREEN (TOP)