Model 2010B Intel 8052AH Basic SBC Operator's Manual

**Second Printing
April 1, 1988
Part # 01–000–282**

Please see section on installation before using

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 con verter 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 approximately an other 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 mm

WEIGHT: 6 ounces (170 grams)

POWER REQUIREMENTS:

9 Volts AT 500 milliamps

Direct I/O Lines Programmable as Input or Output:

40

Indirect I/O Lines Through Expansion Bus:

C000-FFFFH, and 0000-7FFFH if the EEprom is not used.

PROCESSORS:

8031 at 11 MHz (standard)

(Optional) 8751, 8751H, 87C51 at 11 MHz

8032 at 11 MHz

8052AH BASIC

(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 B51.COM. A program called PINSTALL is provided to install it for the Baud Rate or COM port you are using. The software used to handle upload/download from the BASIC option is called B51.COM. Use B51 to save/load your basic programs either in an ASCII format for immediate use with an EEprom, or in a tokenized format so that you can dedicate the controller with a BASIC program in an Eprom.

1.4 Software Specifications

Although the 2010 is capable of communicating at Baud Rates of from 300 to 57,600 Baud, B51 communicates at 9600 Baud. Commands are available to save or load an ASCII BASIC program, and save a tokenized BASIC program to execute from a burned Eprom.

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. When using the B51 program to save tokenized basic to be put into an Eprom, you should use the MKE.BAT program to put the monitor and other things into the correct location in the Eprom.

1.6 Firmware Specifications

The firmware included with the 8052AH BASIC chip takes care of the Ram Refresh and new commands added to 8052AH BASIC. Refer to the memory map. Locations 0-1FFFH in code memory are reserved for 8052AH BASIC. 2000H Through 7FFFH is reserved for BASIC and the firmware monitor. When using a 2864 or 2764 (8K memory device), it is inserted at locations 2000H through 3FFFH. When using a 16K device, it is inserted at 2000H through 5FFFH. When using a 32K device, it is inserted at 2000H through 7FFFH. Note that you lose 8K to the 8052AH BASIC monitor rom.

—Notes—

—Notes—

Chapter 2

Getting Started

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 acci dentally discarded.

2.2 Installation

The 2010 board (depending on the options ordered) generally comes set up to plug in and run. Yours should be set up with the Basic monitor on a 2864. Jumpers will be set for this installation with no other option jumpers.

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 B51 (B51 is an optional program) communications program.

5—Issue commands.

Remember that if you don’t have the B51 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.

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 8052AH BASIC. 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 already 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: leave this jumper open for use with 8052AH BASIC. Ground it to use the 8052AH as an 8032.

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

Default: 4–2 for 2864. Hook 1–2 for 2764,27128. Hook 4–2 for 28256 to allow /WR to program EEprom. Hook 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

Default: 4–2 for 2864. Hook 1–2 for Vcc to Vpp of 2764,27128,27256. Hook 4–2 for 28256 to allow A14 onto pin 1 (A14). Hook 3–2 for 27512 to allow A15 onto pin 1 (A15).

Memory (Usable size)	JB5	JB6	JB7	Type:
2764/A 8K Eprom	1–2	2–1	2–1	ML
2764/A 8K Eprom	none	2–1	2–1	BASIC
2864 8K EEprom	1–2	2–4	2–1	ML
2864 8K EEprom	none	2–4	2–1	BASIC
27128/A 16K Eprom	1–2	2–1	2–1	ML
27128/A 16K Eprom	none	2–1	2–1	BASIC
27256 32K Eprom	1–2	2–3	2–1	ML
27256 24K Eprom	none	2–3	2–1	BASIC
28256 32K EEprom	1–2	2–4	2–4	ML
28256 24K EEprom	none	2–4	2–4	BASIC
27512 32K Eprom	1–2	2–3	2–3	ML
27512 24K Eprom	none	2–3	2–3	BASIC

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 side 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.

—*Notes*—

Chapter 3

New Commands and Additions

3.1 NEW COMMANDS

All these commands can be executed either from an executing basic program or from the immediate mode. Some commands are better executed from the immediate mode, such as for the baud rate.

In the programs shown below, a command that you are to type in from the command prompter is shown in bold. A "enter" is shown as < cr> . A > REM is simply a comment that does not have to be typed.

3.1.1 AUTOEXn

Special command to perform an automatic command on boot up.

AUTOEX0 Causes an auto baud–rate seek on power up. You must strike a space bar to lock onto the baud rate. (default)

AUTOEX1 Saves the current baud rate. No space will have to be struck before any execution begins.

AUTOEX2 When executed will cause 8052AH BASIC to boot to the READY prompter instead of loading and executing program 0. (default)

AUTOEX3 When executed will cause 8052AH BASIC to load and execute program number 0, if there is one saved by that name. If there isn't one, then it will return to the READY prompt.

AUTOEX4 When executed will cause 8052AH BASIC to use our SLOW EEprom program routine to be used for certain types of EEproms that program slowly when using the SAVE and ERASE commands.

AUTOEX5 When executed will cause 8052AH BASIC to use our FAST EEprom program routine to be used with certain types of EEproms that can be programmed fast. (default)

Once any of the above commands are used, they become permanent in the EEprom. To change them, simply issue the opposite command. Eg. if you have used AUTOEX1, to change it use AUTOEX0.

EXAMPLES: Program in PROGRAM 0 that you want to execute on power up with no user intervention:

```
> AUTOEX1
> REM Store current Baud Rate: Else your program
> REM won't execute without striking a space bar
> REM Really necessary if you don't have a
> REM terminal hooked to the 2010!!!
> AUTOEX3
> REM Tell 8052AH Basic to Execute program 0 on
> REM power up.
```

3.1.2 DIR

When the *DIR* command is used, a list of the programs available stored on the EEprom, along with the size is printed to the console.

EXAMPLES:

```
> DIR< cr>
4    size    389
2    size    185
0    size    1084
1    size    988
> _
```

Note: The order in which you saved the programs is how they will appear on screen. There may be up to 254 files if you had room for them on the EEprom.

3.1.3 EGETn

EGET Is a command that PUSHes an 8052AH BASIC 6 byte floating point number from storage in the EEprom onto the argument stack. You can retrieve up to 32 numbers in this fashion (0-31). The numbers are stored in EEprom (or Eprom) from 2740H through 27FFH (192 bytes). You can create a file externally on disk to be stored on the Eprom so that you can EGETn those numbers after power has been off or for re-boots.

EXAMPLES of EGET:

```
> PUSH  32:   EPUT  0 :   EGET  0:   POP  A:   ?  A< cr>
32
READY
> _
```

3.1.4 EPUTn

EPUTn Is a command that POPs an 8052AH BASIC 6 byte floating point number from the argument stack and programs the EEprom in the storage area (2740H- 27FFH). The number may then be obtained by an EGETn and a POP command.

EXAMPLES:

```
> PUSH  123.45678:   EPUT0:   EGET0:   POP  A:   ?  A< cr>
123.45678
READY
> _
> REM now that number is in permanent
> REM storage (until it’s changed)
> REM in the EEprom at location 0 of the EGETs.
```

3.1.5 ERASEn

ERASEn is a command that will ERASE a program from the EEprom program storage area. It is the opposite of SAVEn. The 2010 will remove the program by rewriting the entire EEprom program storage area so that there are no gaps after the program is removed from storage. The range of n is 0–254 (0–FEh).

Examples:

```
> DIR< cr>
4     size     389
2     size     185
0     size     1084
1     size     988
READY
> erase0< cr>
READY
> dir< cr>
4     size     389
```

```
2    size    185
1    size    988
```

3.1.6 INn

INn is a command that will input 8 bits of data and PUSH it onto the argument stack from the 4 ports on the Z80-PIO's. The valid numbers for the ports are 4, 5, 6 and 7. U4 Port A is what we call port 4, U4 Port B is what we call port 5, U3 Port A is port 6 and U3 port B is port 7. (See Appendix A for pinout)

Examples:

```
> IN4:   POPA:   ?  A:   IN4,5,6,7:   POP A,B,C,D< cr>
255
READY
> PRINT A,B,C,D< cr>
255 255 255 255
READY
> _
```

3.1.7 LOADn

LOADn is a command that will cause file number n to be loaded to current Ram memory. A NEW is automatically done before the LOAD, so files will not "merge" in memory. The range of n is 0 to 254. If LOADn is executed on a program line, another program is LOADed and run. You could chain programs together like this. All variables are lost during this process, but you can use *EPUTn, EGETn, LD@ and ST@* commands to save your variables.

Examples:

```
> LOAD9< cr>
READY
> REM Loads file 9 from EEprom to Ram.
> RUN:   REM Run it.
```

3.1.8 OUTn

OUTn is a command that will output 8 bits of data (to port numbers 4, 5, 6, or 7) to the Z80-PIO's, that was POPped from the argument stack.

Examples:

```
> C= 29:   PUSH  22,25,32,C:   OUT  4,5,6,7< cr>
> REM Port 4 pins contain data 29, Port 5 pins contain data 32
> REM Port 6 pins contain data 25, Port 7 pins contain data 22.
> IN  4,5,6,7:   POP  PORT7,PORT6,PORT5,PORT4< cr>
READY
> PRINT  PORT4,PORT5,PORT6,PORT7< cr>
29 32 25 22
READY
> REM PORT4 etc in the above line are just variables we used.
> REM Remember that PORT1 is a Basic Keyword!
```

3.1.9 SAVEn

SAVEn is a command that will save the program that is currently in Ram to the EEprom in file number n. The range of n can be from 0–254. If there is currently a program residing in that file number, then an error message will be issued. If there is not enough memory to save that file, then an error message will be issued.

Example:

```
> SAVE3< cr>
```

3.1.A TKO

TKO is a command that will cause each byte of the program in memory to be output to the console in an ASCII-HEX form. One binary byte becomes 2 ASCII bytes. This command is used by the program B51 to capture the program in Tokenized form to be put on an Eprom using an external Eprom Programmer such as the GTEK model 9000. This command should not be issued from the console. It should be used from the "Enter Command Line" command within B51.

Example:

```
> ^ F
```

Enter Command Line —> **filename [TKO< cr>**

3.2 Differences In V1.1 Intel Basic Commands

3.2.1 Differences Caused by Ram Refresh

Ram refresh is handled in a top priority mode every 2 milliseconds during the *TIMER1* interrupt. This will cause all calculations for the *BAUD, PWM*, every command that uses *TIMER1*, to be 79 machine cycles longer. See the specific commands affected for differences.

3.2.2 Values for Basic Constants

MTOP = E000H

XTAL = 11,000,000 Hz.

TCON = 118

T2CON = 52

TMOD = 16

3.2.3 Sign on message:

```
GTEK, INC.
Model 2010 Basic 51
Version 1.1 b
> _
```

3.2.4 EE/Eprom Code Memory from 2000H through:

2864 EEprom through 3FFFH (8K).
2764 Eprom through 3FFFH (8K).
27128 Eprom through 5FFFH (16K).
27256 Eprom through 7FFFH (24K).
28256 EEprom through 7FFFH (24K).

In relation to the above, SAVE and ERASE will not work with Eprom for code memory. LOAD will load the program from Eprom to ram to run.

3.2.5 Unusable Intel Basic V1.1 Commands

These commands are still there, but use of them may cause you problems because Eprom [EEprom] is at 2000H instead of 8000H.

RAM—Hardware dependent. Use *LOAD* to load program into Ram to *LIST*, etc., instead.

ROM—Hardware dependent. Use *LOAD* to load program into Ram to *LIST*, etc., instead.

XFER—Hardware dependent. Use the *LOAD* command.

PROG, PROG1, PROG2, PROG3, PROG4, PROG5, PROG6, FPROG, FPROG2, FPROG3, FPROG4, FPROG5, FPROG6— Hardware dependent. Use SAVE and ERASE instead.

UI0, UI1, UO0, UO1—Do not use, hardware dependent.

RROM—Hardware dependent. Use *LOAD* and *RUN*.

PGM—Hardware dependent. Do not use.

3.2.6 Intel V1.1 Basic Commands Used Differently

a) The *BAUD* command must be modified. Use the following formulae to calculate the Baud Rate for *List#* and *Print#*:

79 is number of machine cycles for refresh. 12/XTAL is period for 1 machine cycle= about 1.090909 uS. ABN is Actual Baud Number. BR is the baud rate of the device to be output to:

$$ABN= \frac{1}{\frac{1}{BR} - \left(\frac{79 \times 12}{XTAL}\right)}$$

or

ABN = 1 / ((1/baud) - (79*(12/XTAL))

Use the 2010 to calculate it for you:

```
> 10 BR= 1200: REM FOR 1200 BAUD DEVICE< cr>
> 20 ABN= 1/((1/BR)-(79*(12/XTAL)): ? ABN< cr>
> RUN< cr>
```

```
1338
READY
```

> **BAUD 1338< cr>** : REM LIST DEVICE TO 1200 BAUD

You should not use baud rates lower than 600 Baud, because since the same timer for Ram Refresh is used for baud rate running at 300 baud (about 3.3 milliseconds), that is longer than the time between refresh cycles. The ram could probably withstand the rate, but a marginal Ram chip might not make it much longer than the 2 millisecond rate specified for that chip. See Page 28 in the *Intel 8052AH BASIC* manual. The *Intel 8052AH Basic manual* is an optional purchase.

b) Do not use *CALL (0-127)* unless you are using an Eprom greater than 8K and have provided the proper vectors. use *Call [integer]* instead. Page 29 of *8052AH BASIC* manual.

c) The *PWM* statement (Page 62 of *8052AH BASIC* manual) will run differently, due to the Ram Refresh having higher priority. The minimum valid number usable for the number of clock cycles the wave will remain high or low is still 25, but since a Ram Refresh Cycle will occur for every transition, you must figure that there are approximately 79 additional machine cycles added to the number of machine cycles you use for the *PWM* statement.

PWM 100,100,1000 would generate 1000 cycles of a square wave that has a period of (100mc + 79mc)* 2 * 1.0909us = 390uS on P1.2 (2560 HZ). It is not possible to obtain the same frequency as on page 62 because the number that you would use for mc would be Machine Cycles = (217us/(2*1.0909us)) - 79 = 20, which is too low to use in the PWM statement since 25 is the minimum. Referring to the program on page 173, use the one following instead:

```
10 PRINT"1= FREQUENCY FOR PWM"
11 PRINT"2= RELOAD FOR FREQUENCY"
12 PRINT"3= QUIT -",
14 INPUT A
20 IF A= 1 THEN GOSUB 30 : A= 0
22 IF A= 2 THEN GOSUB 110 : A= 0
23 IF A= 3 THEN END
24 GOTO 10
30 T= 12/XTAL
```

```
40 C= 79
50 INPUT "ENTER RELOAD OR 0 TO RETURN - ",B
60 IF B= 0 THEN  RETURN
70 A= 1/((B+ C)*T*2)
80 PRINT "FREQUENCY IS",
90 PRINT USING(#####.###),A :  PRINT
100 GOTO 50
110 T= 12/XTAL
120 C= 79
130 INPUT "ENTER FREQ. OR 0 TO RETURN - ",A
140 IF A= 0 THEN  RETURN
150 B= ((1/A)-(C*T*2))/(T*2)
155 PRINT "RELOAD VALUE IS ",
156 D= INT(B): D= B-INT(B): IF D.5 THEN 160
157 B= B+ 1
160 PRINT USING(####),B :  PRINT
170 GOTO 130
```

The lowest reload value you may use is 25 (as per the *8052AH BASIC* manual). The largest you should attempt to use is 835 due to the Ram Refresh Rate.

d) *PCON, RCAP2, T2CON, TCON, TMOD* Probably should not be used at all, or with extreme caution. If you modify or assign any variables to these registers, you might damage the Ram Refresh ISR and / or the Serial Communications.

e) *TIMER0 (RTC- CLOCK1), TIMER1 (Ram, PWM, LIST#), TIMER2* (Serial Baud Rate Generation), Probably should not be used at all or with extreme caution. If you modify or assign any variables to these registers, you may damage the Ram Refresh ISR and/or the Serial Communications.

—Notes—

—Notes—

Chapter 4

Communications Software

4.1 B51 Installation

On your disk you have B51.COM and PINSTALL.COM. Run PINSTALL to install B51 for the COM port and 9600 baud. The 2010B should be set so that it will begin communication right away. If you don't immediately get the prompter when you run B51, type a space bar first! This will cause the 2010B to lock onto the baud rate. Remember that you have to be sending at the new baud rate and re–boot the 2010B to reset the baud rate with a space.

When you use B51 with your 2010, remember that if you have been using it at another Baud Rate and have done an AUTOEX1 at some other Baud Rate than 9600, you will not be able to communicate with the 2010 until you change it to 9600 with the other program. If you are set with AUTOEX0 (auto baud seek) then you won't have any problem communicating.

4.2 Using B51

Most of the time B51 is a simple communication program, until you execute a CONTROL–F. At that time it will ask you to enter a command:

> **^ F**

Enter Command —> **filename /option/option< cr>**

Options for saving are /S and /T. A /S is used for saving the "filename" as an ascii file. A /T (which must always be used with /S) will save the "filename" in a tokenized format. If you use the /S/T option the filename must always be a filename by the name of TFn where n is 0 through 9.

An ASCII save (/S) will create a filename.B51 file on your disk. A tokenized save (/S/T) will create a filename.TKO file on your disk. You don't have to specify any extension with your filename.

The reason you must use TFn for a filename with the tokenized format is that there is a BATCH file on your disk called MKE.BAT. MKE.BAT will take tokenized files and combine them with the operating system and other system files into a file called BURNME in a binary format. This is so that you can install them in an Eprom using an Eprom Programmer such as GTEK’s Model 9000, to be used on the 2010.

Example: To load an ascii basic file with extension of .B51

> ^ F

Enter Command —> **test< cr>**

The above command will look for a file called test.B51 on your disk and then issue a *NEW* command to the 2010. It will then send the ascii file to the 2010 one line at a time, waiting for the prompter to come back before sending any more characters at the end of a line. It is possible that if you "saved" the file before with lines that were full (79 characters), you will not be able to get the file back into the 2010. It will cause the 2010 to begin beeping with each character that is sent, and not accept any more characters. If that is the case, you will have to edit the line with a word processor before you can send it back, to eliminate ALL spaces on the line. This is the typical cause of the problem if you have previously "saved" the file in the ascii format. If you wrote it with a word processor, you have exceded the line limit and should break the line into two lines.

Example: To save program in ram to disk in ascii format

> **^ F**

Enter Command —> **test /s< cr>**

The above will look for a file called test.B51 on the disk. If it does not exist, then it will cause the current program in the 2010 to be *LIST*ed to the disk in a file called test.B51.

Example: To save program to disk in tokenized format

> ^ F

Enter Command —> **tf0 /s/t< cr>**

The above will look for a file called TF0.TKO on the disk. If it does not exist, then it will cause the current program to be sent to the disk in a "tokenized" format. You will see ascii-hex numbers being listed to the screen during the process. You cannot get the TF0.TKO file back into the 2010 without burning the file into an Eprom. Of course it does not destroy the program that is currently in the 2010 during the /s/t process.

4.3 USING MKE.BAT

After the TF0.TKO file is on the disk, if you would like to install that program into an Eprom to run as a dedicated program (before you /s/t you should have done an AUTOEX1), you may issue the MKE.BAT command from the DOS system prompt. The line is too long to print as 1 line, but here is MKE.BAT:

```
copy b9035v2.bin/b+ puts.bin/b+ xb96.bin/b+ %1.tko/b+
%2.tko/b+ %3.tko/b+ %4.tko/b+ 5.tko/b+ %6.tko/b+ %7.tko/b+
%8.tko/b+ %9.tko/b+ eofchar.bin/b  burnme/b
```

Example: To send data for eprom to GTEK programmer

C> **MKE TF0 TF1 TF2< cr>**

The above will take the 3 tokenized basic files called TF0, TF1 and TF2 and add them to the system information for the Eprom and put them in a file called burnme. The above MKE file with the specified parameters as executed by the batch file becomes:

```
copy b9035v2.bin/b+ puts.bin/b+ xb96.bin/b+ TF0.tko/b+
TF1.tko/b+ TF2.tko/b+ eofchar.bin/b burnme/b
```

The burnme file is a binary file that then may be used to create an Eprom with those 3 basic files on it. If you have AUTOEX3 in force then the file you named TF0 will become the program that begins running on a BOOT (applying power or reset).

On a GTEK Model 9000 programmer this BURNME file may be sent directly to the Eprom. On the other model GTEK programmers (7128, 7228, 7956), you must first run the program GHEX on the BURNME file if you are not using the program PGMX7 (or PGMX), since it is a Binary file and it should be an Intel Hex file instead for those models running PGX or other program. On other programmers you should follow their procedures for burning a binary file into an Eprom.

When you install the Eprom into the 2010 in place of the EEprom, REMEMBER TO SET THE JUMPERS IN THE RIGHT POSITION! If you don’t then some unexpected things might happen, like it don’t work at all!

—Notes—

—Notes—

Chapter 5

2010B RS–232 Interface

The model 2010B 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. 2010B and Intel Basic does not use this pin.
6—(DSR)	—>	Data Set Ready. Always at + 12 volts when power is applied to the 2010B. Intel Basic and B51 does not use this pin.
7—(SG)	< —>	Signal Ground.
8—(CD)	—>	Carrier Detect. Not hooked up.
20–(DTR)	< —	Data Terminal Ready. Not used by Intel Basic or 2010B.

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 (opt.)
CTS (o) 5	5 . . . . (i) CTS (opt.)
DSR/CD (nc) 6/8	6/8 (i) DSR/CD (opt.)
SG (i/o) 7	7 . . . . (i/o) SG
DTR (i) 20	20 . . (o) DTR (opt.)

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 (opt.)
CTS 5	8 . . . . (i) CTS (opt.)
DSR/CD 6/8	6/1 (i) DSR/CD (opt.)
SG 7	5 . . . . (i/o) SG
DTR 20	4 . . . (o) DTR (opt.)

—Notes—

Appendix A

2010B 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:

The following notes refer mostly to a 2010M. They are here for reference only.

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	bit 6	05	36	bit 7	on U3 this is port B
	bit 4	06	35	bit 5	
	bit 2	07	34	bit 3	
	bit 0	08	33	bit 1	
Port6	bit 6	09	32	bit 7	on U3 Z80–PIO this
	bit 4	10	31	bit 5	is Port A
	bit 2	11	30	bit 3	
	bit 0	12	29	bit 1	
Port 5	bit 6	13	28	bit 7	on U4 Z80–PIO this
	bit 4	14	27	bit 5	is port B
	bit 2	15	26	bit 3	
	bit 0	16	25	bit 1	
Port4	bit 6	17	24	bit 7	on U4 Z80–PIO this
	bit 4	18	23	bit 5	is port A
	bit 2	19	22	bit 3	
	bit 0	20	21	bit 1	

On the 2010B, using the INn commands and the OUTn commands you can get to the above "ports" in column 1. A command of IN4 would refer to Port 4 (Z80–PIO U4 Port A, bits 0–7). All the bits are pulled high by a resistor pull–up pack, and the port will have a value of 255 (0FFh) when read with nothing hooked to it (when first booted). If bit 7 was shorted to ground the data would read as 127 (07Fh) when read. An OUT4 would cause the data from the argument stack to be output to Port 4.

When the 2010B does a Basic "OUTn" instruction, the output pins of the Z80–PIO are not set to output as referred to in the Z80–PIO data sheet unless the bit datum is 0. They are always set as inputs if the bit datum is 1. The resistor packs pull the pins to Vcc instead of the bits being "set" to output a 1 on the port. This avoids contention problems where you might be mixing inputs and outputs on the same Z80–PIO port.

This means that if you are driving a transistor base with one of the bits, for example, when you "read" the port with a Basic "INn" instruction, that bit will read as a 0 because the maximum voltage that will be on that transistor base is only going to be .6 volts, which is a TTL low. Of course if you were to output a 0 to that bit, you would cause the transistor (NPN) to turn off. Outputting a 1 to that bit would cause the transistor to turn on again since the base is pulled up to Vcc through a 2.2K resistor. If you have to have a different bias level, you should use either a darlington type transistor or cascade them.

When the 2010B powers on, all of the Z80–PIO port bits will be 1 because they are all programmed for "read" on boot. None will be 0 unless you have done an "OUTn" instruction with that bit set low.

After you have done an "OUTn" instruction and you read that port, you are NOT reading the output pin of a particular bit if it was set to 0. You are reading the output latch instead. Other bits that were set to 1 are still set as inputs and you are still actually reading the output PIN and not the latch.

In the case of PORT1, which is an 8 bit port that comes directly from the 8052AH chip, the bits are handled as they were coming from the processor normally. If you output a 1 on a bit on this port, you are driving the bit high also, even though it also has a resistor pull–up pack.

Examples:

```
'Read I/O after an OUTn instruction
PUSH 85     'Bit pattern 01010101 (bits 7–0)
OUT 7       'output pattern to port
IN 7        'POP data from port to argument stack
POP A       'get data from argument stack. (01010101)
? A         'prints 85 unless bit 6,4,2 or 0 were pulled low.
'Bits that are 0 in this case were read from the output LATCH.
'Bits that are 1 in this case were read from the output PINS.

'Outputting and Inputting data from Ports 4–7 and PORT1.
PUSH 39     'push argument onto stack
OUT 4       'output data to PORT 4 (Z80–PIO U4 Port A,
            'which is pins 17–24 on P1 connector. see table
            'The OUT also fixes the argument stack
A= 10: B= 20: C= 30: D= 40: E= 50
PUSH A,B,C,D        'Push arguments onto stack (LIFO)
OUT 4,5,6,7         'Output arguments to ports 4, 5, 6, 7
PORT1= E            'output to PORT1
            'PORT1= 50, Port 4= D, 5= C, 6= B, 7= A
IN 4,5,6,7  'Push data from ports 4, 5, 6, 7.  The data
            'is "Pushed" onto the argument stack
E= PORT1    '"Read" PORT1 data
POP A,B,C,D         'Pop data from stack (LIFO)
                    'A= Port 4, B= 5, C= 6, D= 7
? A,B,C,D,E 'prints 10 20 30 40 50. Note the order is not
            'reversed, because of the "Last In First Out"
            'nature of the stack.
```

On a 2010M (machine Language board), and a 2010B, P1.0 – P1.7 is PORT1 on the 8031 U2. On a 2010M, 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—

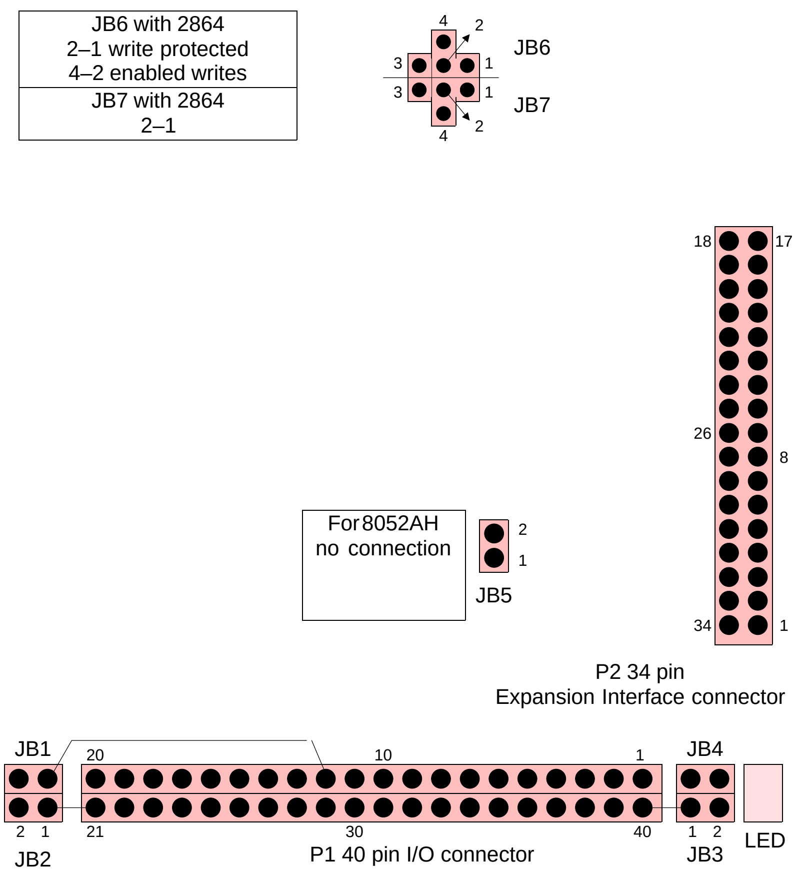
Appendix B

Jumpers Used on the 2010B Board

jumper	pin #	function
JB1	1	pin 12 of P1
	2	ground
JB2	1	pin 21 of P1
	2	ground
JB3	1	Pin 40 of P1
	2	ground
JB4	1	nc (for prototyping)
	2	ground
JB5	1	U2 pin 31 —EA. ground for ext. access.
	2	ground
JB6	1	Vcc
	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)
	4	U2 pin 27 (A14)

Default jumpers for the 2010B are: JB1, JB2, JB3, JB4, JB5 no jumper. JB6 pins 2–1 for write protected 2864. (jumper 4–2 for writes to EEprom before you use SAVE, ERASE, EPUTn, etc.) JB7 pins 2–1.

—Notes—


JB6 with 2864
2–1 write protected
4–2 enabled writes
JB7 with 2864
2–1
JB6
JB7
4
2
3
1
For 8052AH
no connection
JB5
18
17
26
8
34
1
P2 34 pin
Expansion Interface connector
JB1
20
10
1
JB4
2 1
21
30
40
1 2
JB2
P1 40 pin I/O connector
JB3
LED

—*Notes*—

Appendix C

Example Ascii Basic Program

This is an exercise in writing a basic program that can be saved on the EEprom and/or saved to the disk. We will go from writing the program to saving it on the EEprom and then saving it on the disk. Follow the steps in order. Commands or data that you will have to type yourself will be in bold. A < cr> means to strike the "enter" key. A < sp> means to strike the space bar. Comments will be between two left and right arrows (also called "greater than" and "less than" signs).

1—Communicate with the 2010B with B51 from the DOS prompter:

C> **B51< cr> < sp>** < in case you are using auto baud rate det.>

—Log on message—

> _

Type in the following program from the READY prompt. hit cr at the end of each line.>

```
5 Rem This program may have run when you first powered up your 2010B
10PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
20PRINT "THIS IS THE AUTO BOOT FEATURE OF THE 2010.":PRINT
30PRINT "THE JUMPERS ON THIS BOARD HAVE BEEN SET SO THAT NO"
40PRINT"WRITES CAN BE DONE TO THE EEPROM. TO WRITE TO THE EEPROM,"
50 PRINT "YOU MUST MOVE THE JUMPER ON  JB6 FROM VCC (2-1) TO"
60PRINT "TO /WR (2-4)."
70PRINT:PRINT:PRINT :PRINT:PRINT
```

2—You can now list the program.

> **list< cr>**

```
5 Rem This program may have run when you first powered up
your 2010B
10 PRINT:PRINT: PRINT: PRINT: PRINT: PRINT: PRINT: PRINT:
PRINT: PRINT: PRINT
20 PRINT "THIS IS THE AUTO BOOT FEATURE OF THE
2010.":PRINT
30 PRINT "THE JUMPERS ON THIS BOARD HAVE BEEN SET
SO THAT NO"
40 PRINT"WRITES CAN BE DONE TO THE EEPROM. TO WRITE
```

```
TO THE EEPROM,"
50 PRINT "YOU MUST MOVE THE JUMPER ON  JB6 FROM VCC
(2-1) TO"
60 PRINT "TO /WR (2-4)."
70 PRINT: PRINT: PRINT :PRINT: PRINT
READY
> _
```

To save this program you just typed into ram onto the EEprom, first make sure that you have moved the jumper (2–4) and type:

> **SAVE 0< cr>**
> **dir< cr>**

```
0      size       1089
READY
> _
```

Your program is now saved on the EEprom in file # 0. If you then type:

> **autoexec3< cr>**

```
READY
> _
```

This will cause the program you just entered to automatically begin running when you apply power to the 2010. To save your program to an ascii file on your disk (using B51), type:

> **^ F**
Enter Command Line —> **BOOTMSG /s< cr>**
READY
> _

This will cause B51 to *list* the program in ram to a disk file by the name of *BOOTMSG.B51*. You can then load the program by typing:

> **^ F**
Enter Command Line —> **BOOTMSG< cr>**
READY
> _

The above causes B51 to look on the disk for a file by the name of *BOOTMSG.B51* and if found will issue a *NEW* command to the 2010B and "enter" the data in the 2010B as if you had typed it from the keyboard. When done, control is returned to the command prompter of the 2010B. If you would like to "load" a program from the EEprom type:

```
> load 0< cr>
READY
> run< cr>
```

The above has the following effect:

```
THIS IS THE AUTO BOOT FEATURE OF THE 2010
THE JUMPERS ON THE BOARD HAVE BEEN SET SO THAT NO
WRITES CAN BE DONE TO THE EEPROM. TO WRITE TO THE EEPROM,
YOU MUST MOVE THE JUMPER ON JB6 FROM VCC (2–1) TO
TO /WR (2–4).
```

—Notes—

—Notes—

—Notes—

—Notes—

—Notes—

—Notes—