nohau
CORPORATION

EMUL 51

EMUL51™-PC

nohau
CORPORATION

USER ORIENTATION 1

INSTALLATION 2

COMMANDS 3

PODS 4

TROUBLESHOOTING 5

TUTORIAL 6

SIMULATOR 7

INDEX I

1 USER ORIENTATION

2 INSTALLATION

3 COMMANDS

4 PODS

5 TROUBLESHOOTING

6 TUTORIAL

7 SIMULATOR

I INDEX

We will be pleased to answer any questions you may have, or discuss your comments. If you prefer, you may write to us at the following address:

NOHAU CORPORATION
51 E. Campbell Ave.
Campbell, CA 95008

Phone (408) 866-1820
FAX (408) 378-7869

(NOHAU is pronounced "know how")

DISCLAIMER

Nohau Corporation's EMUL51-PC is sold with a one-year warranty on the hardware. The software is sold with no warranty, but upgrades will be distributed to all customers up to one year from the date of purchase.

Nohau Corporation makes no warranties, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. In no event will Nohau Corporation be liable for consequential damages.

Edition #12

Table of Contents

1 USER ORIENTATION

Hardware Overview 1-3
User Interface 1-6
Command Language Overview 1-9
Code Window Source Level Debugging 1-14
How the Standard Trace Works 1-22
How the Advanced Trace Works 1-28
How to Build a "State Machine"
Using the Sx Functions 1-37
Breakpoints 1-46
Macros 1-49
Data Windows 1-55

2 INSTALLATION

Dear 8051 User: 2-3
Materials and Supplies 2-3
Power Requirements 2-3
Inspection of Jumpers 2-4
Emulator Board 2-6
Standard Trace Board 2-10
Advanced Trace Board 2-13
Addressing the Emulator and Trace Boards 2-14
Installing the Emulator and Trace Boards 2-16
Mapping 2-17
System Requirements 2-18
Getting Started 2-18
Files Provided 2-26
EMUL51-PC/BOX 2-27
BANK SWITCHING 2-33

3 COMMANDS

AASM Assembly mode 3-5
ABR Automatic Breakpoint Remove 3-5

ACC Accum. display and set 3-7
ADJUST Windows 3-7
ASM Assemble one line 3-7
AV View Auto Variable 3-8
B *(see ACc)*
BANK Bank 3-8
BANKADDRESS Bankaddress 3-9
BANKBYTE Bankbyte 3-9
BANKMODULE Define or display banked modules 3-9
BANKSEGMENT Banksegment 3-10
BANKSY1 Set or show bank SY1 signal 3-10
BB Break on internal bit 3-10
BIC Break on Internal Contents 3-11
BR Breakpoint registers 3-14
BRM Breakpoint Mode Register 3-14
BRS Breakpoint in Go Slow mode 3-15
BUFFERSIZE Buffersize 3-15
BYB Break on internal byte 3-16
CALCULATE Calculate checksum 3-17
CB Code memory 3-18
CLB Clear Breakpoint RAM 3-19
COUNT Execution of block 3-20
CST *(see OST)*
CW Code Window On/Off 3-20
DASM Disassemble 3-21
DBYTE *(see CByte)*
DEFINE Define macro symbol 3-22
DIR Macro directory 3-23
DISABLE Disable an EMUL51 function 3-23
DOM Module domain 3-26
DP Disassemble from PC 3-26
DPTR Data Ptr Reg display and set 3-26
ENABLE Enable an EMUL51 function 3-27
END End 3-29
EVALUATE Evaluate an expression 3-29
EVENT (Sim. only) Specify sim. event 3-30
EXIT Exit to DOS 3-31
FILL Fill memory with value 3-31
Format Formal Commands 3-32
GB Go till Breakpoint 3-33

GI Go till contents match 3-33
GO Start emulation 3-34
GR Go Register 3-36
GS Go Slow 3-36
HEAP Heap 3-37
HELP Help 3-38
IB Internal Break on pattern 3-38
IF Conditional 3-38
INCLUDE Load command file 3-39
INTERRUPT Display interrupts 3-40
J *(see SN)*
LC Loop Counter 3-40
LIN Line step 3-41
LIST Open log file 3-41
LOAD Load program/symbols 3-42
LOB Load Binary 3-44
LOP Load Ppa setup command 3-44
LOVER Line OVER 3-44
LOW Load Window setup 3-44
MACRO Display MACros 3-45
MACDELAY (Sim. only) Macro after delay 3-45
MAPC Map Code 3-46
MAPX Map eXternal data 3-46
MEM Mem 3-46
MOD Module 3-47
N *(see SN)*
NOHIT Nohit 3-47
NOSNOW Remove screen snow 3-47
OST Open Slow Trace buf 3-47
P1 *(see ACc)*
P3 *(see ACc)*
PBYTE *(see CByte)*
PC PC Reg display and set 3-48
PPA Analyzer 3-48
PUT Save macros in DOS file 3-49
QRA Trace Qualifier Register 3-49
QUIET Quiet 3-51
RBIT Bit memory 3-51
RBS Bank select 3-52
RBYTE (see CByte)

RCS Relative Computer Speed 3-52
REG Display regs. 3-52
REMOVE Remove symbols or macros 3-53
REPEAT Repeat block 3-54
RESET Reset to initial status 3-55
R0 Display and set working reg. 3-55
R1-R7 *(see R0)*
S *(see Step)*
SAP Save Ppa setup command 3-56
SAVE Save program or symbols 3-56
SAW Save Window set up 3-57
SCOPE SCOPE Command 3-57
SEB Set Breakpoints 3-57
SECONDS Display emulation cycles 3-58
SERIAL (Simulator only) Map Serial Port 3-58
SHELL Shell 3-60
SL SL 3-60
SN Step Next 3-60
SNAP SNAP Command 3-62
SO Source level debug 3-63
SP *(see ACc)*
SST Single Step Trace 3-63
STACK Display contents of stack 3-64
STEP Execute one instruction and stop 3-64
SUFFIX Display or set radix 3-65
SWD Setup data window 3-65
SWR Setup register window 3-65
SWX Setup ext. data window 3-66
SY0a SY0a Reg. display and set 3-67
SYMBOLS Display symbols 3-67
SYMLOAD Load symbols from a converted
assembly listing 3-67
SYSTEM Execute DOS Command 3-69
TABS Tab 3-69
TB Trace Begin 3-69
TBR Break on Trace trig 3-70
TC Trig Counter 3-70
TD Trace dissassemble 3-71
TDF Frame display 3-71
TDL TD in List mode 3-72

TE Trace End 3-73
TM0 *(see DPTr)*
TM1 *(see DPTr)*
TR Trace Register 3-73
TS Trace setup 3-74
TSGET Load Trace Setup 3-74
TSPUT Store Trace Setup 3-74
UNTIL *(see COunt)*
VERSION Version Number 3-75
WAIT Wait for keystroke 3-75
WATCH ? and w[x]? - To Display C Variables 3-75
WHILE *(see COunt)*
WRITE Display express. or string 3-76
XBYTE *(see CByte)*
XTAL XTAL 3-76
:name *(see MACro)*
25, 43, 50 Screen Mode Commands 3-77
? *(see WATCH)*

4 PODs

Pod Selection 4-3
Connection and Setup 4-10
Adapters 4-10
"Generic" Board: 40-Pin External Mode POD,
POD-31, POD-C31, POD-32, POD-C32, POD-44,
POD-C154, POD-C252/C51FA, POD-C321, POD-C652 4-13
POD-31S 4-17
POD-51 / POD-C51 4-21
POD-C51B, POD-C652B 4-31
POD-C51FX 4-37
POD-C51GB-PGA 4-45
POD-C51SL-16 4-49
POD-C52 4-55
POD-C054 4-63
POD-C152-DIP 4-69
POD-C152-PGA 4-73
POD-407-14.75 4-79
POD-CL410 4-85
POD-C451-DIP 4-91

POD-C451-PGA 4-95
POD-C451B-PGA 4-99
POD-C452-PGA 4-105
SAB C502 Emulation 4-109
SAB C503 Emulation 4-111
POD-C515A-PGA-18 4-113
POD-C517A-PGA-18 4-123
POD-C517AB-PGA-18 4-125
POD-C517B-PGA 4-131
POD-C528 4-143
POD-532-PGA, POD-535-PGA, POD-C535-PGA 4-151
POD-C537-PGA 4-161
POD-C550-PGA 4-167
POD-C552-PGA, POD-C562-PGA 4-173
POD-C552B-PGA 4-177
POD-C558-16 4-183
POD-C575 4-191
POD-CL580 4-197
POD-C592-PGA 4-203
POD-C598-KIT 4-209
POD-C751 4-211
POD-C752 4-215
POD-CL782 4-221
POD-5001-16 4-227

5 TROUBLESHOOTING

Basic Standalone Troubleshooting 5-3
Error Messages 5-8
Serial Box 5-8
Problems Operating in Target Systems 5-8
Source Code Display Problems 5-11

6 TUTORIAL

User Interface 6-3
Emulator Commands 6-15
Standard Trace Commands 6-22
Advanced Trace Commands 6-41
Advanced Trace Features 6-56
Search Function 6-72

7 SIMULATOR OPTION

Overview 7-3
Invocation 7-3
User Interface 7-4
Simulator Engine 7-4
External Environment Simulator
(EVENT - MACRO) 7-4
External Environment Simulator
(SERIAL CHANNEL) 7-6

Appendix A

INDEX

NOHAU Sales Offices, Reps, And Distributors

1 USER ORIENTATION

Hardware Overview 1-3
User Interface 1-6
Command Language Overview 1-9
Code Window Source Level Debugging 1-14
How the Standard Trace Works 1-22
How the Advanced Trace Works 1-28
How to Build a "State Machine" Using the Sx Functions 1-37
Breakpoints 1-46
Macros 1-49
Data Windows 1-55

1 USER ORIENTATION

The EMUL51-PC is a PC-based emulator for the 8051 family of microprocessors. This section of the manual begins with a hardware overview, then describes the menu-driven user interface and the command language. After that, differences between EMUL51-PC and other emulators are listed. The final pages of this section summarize several of the major EMUL51-PC features, including:
Trace, Breakpoints, Static Windows and Macros.

Hardware Overview

The basic hardware consists of the emulator board which is mounted in one of the PC's slots. The pod board is the small board that replaces the microcomputer chip on the target system. The ribbon cable connects these two boards. The optional trace board contains a piece of ribbon cable to connect the trace board to the emulator board. All three boards have jumpers to configure them for different functions.

The pod board has an LED which is on when EMUL51-PC is in emulation mode. Pushbutton switch S1 can be used to reset the microprocessor. The target systems reset can be disconnected by removing a jumper. This is useful if your target has a watchdog, controlled under software, which resets the chip if the software is not running.

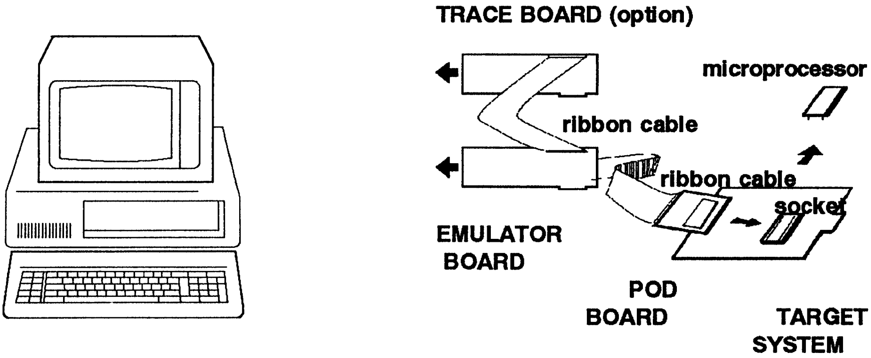


Figure 1. Basic Hardware Components

Additional sockets may be added to the socket pins on the solder side of the board. This may have to be done if components on the target system are too high. There are also different adapters available. Please refer to a current price list.

Box Option

To accommodate users having PCs without spare card slots or who may need to move the emulator frequently, as well as PS/2 users, a Box option of the EMUL51-PC is also offered, (Figure 2). The Box has its own power supply and accepts the same emulator and trace boards.

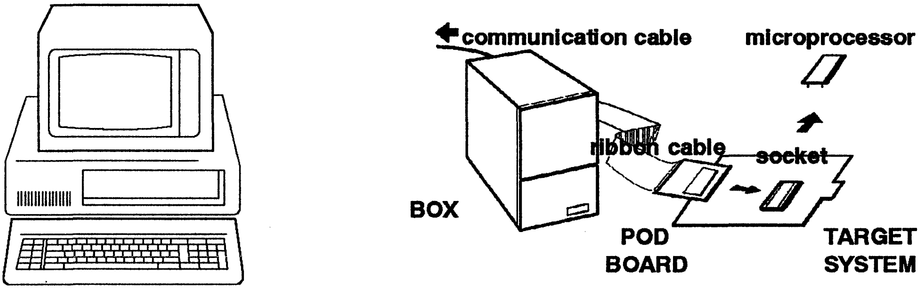


Figure 2. EMUL51-PC Box Option

Dimensions of the Box are 5 1/2 in. wide, 9 in. tall and 15 1/8 in. deep.

The Box-P version (not available in all areas) communicates with the PC over a dedicated parallel port which extends the I/O bus of the PC. The Box-P parallel bus will run at full speed, but does require a "non-standard" parallel card to be installed in the PC.

The Box-S version communicates with the PC oveer a standard serial port. Communication between the PC and emulator is slower with the serial Box-S, but we have reduced transfer time to a minimum through use of sophisticated block transfers of data and by 'PROM:ing' time-critical code in the box. Transfer speeds are given below.

Serial Channel LOAD: 1.5 sec. per k with 10 MHz AT
SAVE:30 sec. per k with 10 MHz AT
Boards in PC LOAD: 0.2 sec. per k with 10 MHz AT
SAVE: 0.5 sec. per k with 10 MHz AT

The Box-M version has a serial channel and an internal modem. This allows the PC to be at a location remote from the Box and system under test.

Additonal details about the various Box versions can be found in the Installation chapter.

User Interface

There are probably as many ideas of how a user interface should work as there are users. We have combined a fast command language for the frequent user with a menu-driven, context-sensitive help interface for the beginner and infrequent user. The idea is that by using the menus you will automatically learn the commands.

The following shortform overview shows how to move about the screen using both menus and commands. For a more detailed hands-on session, please refer to the Tutorial chapter.

1. **The Menu line.** The top line on the screen shows the 11 major command groups. You can use the right and left arrow keys or the mouse to move between the 11 groups. At the bottom of the screen you will always see a help line explaining each item in more detail. By pressing RETURN, or clicking the mouse, a "pulldown" menu with detailed command choices will be seen.

2. **Pull Down Menus.** The first item will be highlighted. You can now use the up and down arrows or the mouse to choose a command. Also here the bottom line will explain each command in detail. If you use the right or left arrow key, or the mouse, you will move to the top item on the next menu. To "escape" up to the menuline from a pull down menu, use the ESC key. To go to the command line, type ESC while at the menu line.

3. **Help window.** If you press the F1 Function Key at any time, the help window will open and a detailed description of the current item will appear. You can then use the arrow keys and PgUp and PgDn keys when you study the help text. ESC closes the help window.

4. **Executing a command from the menu.** If you press RETURN, or click the mouse, at the command you want to perform, the corresponding command will appear at the bottom of the screen. You will now also see an additional "syntax" help line telling you what possible parameters you now must enter. This extra help line may be turned on and off by pressing the F9 Function Key.

5. **Moving back to the menu from the command line.** If you want to move back from the command line to the menu line press the ESC key. You can also type MENU at the command line to get to the menu line. A third way is to simply use the mouse to point at the desired group and click it to pull

down the desired menu. See F7 below for more information.

6. **Editing the command lines.** You can use the LEFT ARROW and RIGHT ARROW keys to insert or over-type the text already on the command line. You can toggle between INSERT Mode (underscore cursor) and OVERTYPE Mode (block cursor) using the "Ins" key. See F7 below for more information.

7. **Function keys.** The function key specifications will appear at the bottom of the screen when you are in command mode. You can activate them by pressing the desired function key or point at the desired function at the bottom of the screen with the mouse and click it. They have the following functions:

F1 Help Puts the HELP window with appropriate help text up on the screen. This is especially useful when you are in the pull down menus, although the "function key line" at bottom of the screen is not visible at that point.

F2 CpyChr Copy characters one by one from the previous command line.

F2 SrcAsm When the CODE window is open, F2 is used to toggle between Assembly Mode and Source Mode.

F3 CpyCmd Copy Command. Copy the previous command line.

F4 AdjWin Adjust Windows. Move current window and/or change size. (See help line on bottom line).

F5 NxtWin Change current window. Window border will indicate current window.

F5 Speed? When the CODE Window is open, F5 is used to toggle between the full speed GO mode and the GO SLOW mode, which is a single step mode, (see the GS command). GO mode will execute the instruction where the breakpoint is set, while GO SLOW will stop at the breakpoint.

F6 WOnOff Window On / Off. Remove window and show full screen. Toggles On and Off, (see help window at WIN on the top menu line).

F7 PrvCmd Previous Command. Allows you to go back and look at the most current 20 commands. You may edit and reuse any command. These 20 commands are stored in a circular buffer, so pressing F8 (NxtCmd) when you are at the current command will give you the oldest command. When editing a line, you may toggle the cursor between INSERT Mode (underscore cursor) and OVERTYPE Mode (block cursor). Use the "Ins" key to toggle.

F8 NxtCmd Next command. Step forward in the command buffer, (see description of F7 also).

F9 HlpOnf Help On / Off. Switch syntax help at second last line On or Off.

F9 GoCurs When the Code Window is open, F9 sets a breakpoint at the cursor location in the Code Window and executes a GO command.

F10 Repeat Repeat last command. Works like F3, but includes a RETURN to complete the command.

Shortcuts for toggling some windows:

CTRL-R register window (Not REGISTER WINDOW on top of screen)
CTRL-W watch window
CTRL-A code window
CTRL-D data windows

Command Language Overview

The Commands chapter contains an alphabetical listing of all available commands. In this Overview we divide the commands into four levels. For each level the commands within that level will be listed, followed by free-form descriptions of some typical uses. For more detailed descriptions of each command, refer to the Commands chapter.

Level 1 - Commands you have to know in order to perform the most basic operations

ASM (A)	One-line assembler command
BRx	Breakpoint registers, (x is 0 to 9)
CBYTE (CB)	Display or set contents of user code bytes
DASM (D)	Disassemble program memory
DBYTE (DB)	Display or set contents of internal data
EXIT (EXI)	Terminates the EMUL51-PC and returns to DOS
GO (G)	Starts emulation
LOAD (LOA)	Loads user program and/or symbol table
MAPC	Map code memory to emulator or target
RBIT (RBI)	Display or set bit memory
RBYTE (RB)	Display or set contents of internal registers
REGISTERS (R)	Display some internal registers
RESET (RES)	Reset BRx, EM or CHIP
STEP (S)	Single Step
TD	Displays contents of trace buffer
XBYTE (XB)	Displays or sets contents of external memory
CW	Open the CODE Window.

After EMUL51 is loaded, especially before you become familiar with all commands, you will use the F1 Help key a lot. Spend some time to learn how to get to the different paragraphs.
You will probably not need to alter the mapping, because the code memory is by default mapped to the emulator and the external data memory is mapped to the target system. If you want to emulate from a PROM on your target system, you need to use the MAPC command.

You will load your .obj or .hex file, which has been prepared from the assembler or compiler. Use the LOAD command.

Then you may use the CW, Code Window, command to look at what you loaded.
Now you may want to start the program. Use S to single step or G to start the program. When you use the G command you will probably want to break on some address to be able to view registers, external memory, and so on. Say you wanted to break on address 012AH. Then type G T 12A. The program will start on the current PC (program counter) value, which by default is 0000H. If you want to specify the start address, (changing PC for instance to 0100H), type G F 100 T 12A. Now your program will start executing at address 0100H and it will break at 012AH. It will, of course, only break if 012AH is the first byte of an instruction, and if the program gets to execute that instruction. If it doesn't break, you can always break from the keyboard using the ESC key.

After a break you can look at the trace by typing TD, which will show you the last few instructions that the processor executed before the break. To look further back in the trace buffer, use TD -1000 (for example). This will start displaying 1000 "frames" back in the buffer. Each instruction consists of 2 - 4 frames.

Now you will probably also want to use the DB, RB, CB, XB, or RBI commands to take a look at the contents of different memory locations. (Or alter them).

Use Ctrl + R to view the most common registers. (See "Static Windows")

Now if you need to change your code, you may use the A command. First type A ORG 200. This will change the internal assembly pointer to location 0200H. Then type A MOV A,#10 for instance, or any other instruction you want to assemble into the code memory. The assembly pointer will automatically be advanced the correct number of bytes. (See the AA Command)

To continue from where the break took place simply type G, or if you want to go to a new breakpoint, say 0234H, type G T 234.

It is not unlikely these are all the commands that you will ever use. But if you take the time to learn the commands in LEVEL 2, LEVEL 3, and LEVEL 4, we think that you will find it very rewarding!

Level 2. Commands that take advantage of the powerful EMUL51-PC TRACE features.

TB	Trace begin, (used to restart trace)
TBR	Used to activate a breakpoint via trig condition
TC	Displays or sets trace counter
TD	Trace display, disassembled form
TDP	Trace Disassembly with Port and address
TDF	Trace display, "Frame form"
TDL	Trace display, disassembled form and No pause
TDFL	Trace display, "Frame form" and No pause
TDPL	Trace Disassembly with Port and address, no pause
TDS	Trace Display Single step, (OST, CST)
TDFS	Trace Display Frame Single step, (OST, CST)
TE	Trace End, stops trace while emulating
TR	Displays or sets the trace register
TS	Trace Setup window

Even though you may do some very complicated things with the trace, it is quite simple to use. The TR sets up how the trace conditions will be used, and TC sets up the trig counter. Use the TS command to set up TR, TC, and TBR.

TB and TE are used restart and break the trace when you are in emulation, (without disturbing emulation). TD displays the trace buffer in disassembled form, and TDF displays the trace buffer in frame form. TBR, if ON, causes a breakpoint to occur when a trace trigger occurs.

For more details, see "How Trace Works" later in this chapter, and the descriptions of individual commands in the Commands chapter.

Level 3. Commands associated with the MACRO features.

BB	Set up Break on direct write to Bit address
BIC	Set up Break on Internal Contents
BYB	Set up Break on direct write on BYte address
COUNT (COU)	Begins command block to be executed n times
DEFINE (DEF)	Define macro or symbol
DIR	Displays directory of macros
DISABLE (DIS)	Disable macro expansion or symbols
ENABLE (ENA)	Enable macro expansion or symbols
IB	Set up Instruction Break
IF	Macro IF condition
INCLUDE (INC)	Reads sequence of commands from a file

MACRO (MAC)	Displays macro definitions
:"macro name"	Invokes previously defined macro
PUT	Stores macro definitions on file
REMOVE (REM)	Remove macro or symbol
REPEAT (REP)	Begins block of commands to be repeated
UNTIL	See COUNT and REPEAT for details
WAIT	Waits for a key stroke
WHILE	See COUNT and REPEAT for details
WRITE (WRI)	Displays evaluate expression or string

The MACRO commands will save you a lot of typing! This is especially the case when you need to type a sequence of commands over and over. By using the DEF :"macro", where "macro" is a name that you supply, you can define a new macro by simply typing the commands and ending the macro with the EM (end macro) command. Then to execute the macro you type :"macro name," and the commands will automatically be executed. You can save macros on disk by using the PUT command. To load a macro back in, use the INC command. To write macros you may also use any editor under DOS. If you need to alter a macro without leaving the emulator program, use the SYS command to invoke your editor from within the EMUL51! By taking advantage of all the macro commands, you may simplify your testing enormously.

Please also refer to the detailed descriptions of each command in the Commands chapter!

Level 4. Miscellaneous other commands that can be very useful

ACC	Displays or sets the accumulator
BRM	Breakpoint mode register
BRS	BReakpoints for GS command
CALC	Calculates checksum in code memory
CLB	CLear Breakpoint RAM
DOMAIN (DOM)	Displays or sets default module name (PL/M51)
DPTR	Displays or sets the DPTR registers DPH + DPL
EVALUATE (E)	Evaluate expression
FILL	Fill, (write value to a range of addresses)
GI	Go till Internal register contents match
GS	GO Slow, ("Run" using single step)
INTERRUPT (INT)	Displays interrupt status
LIN	Line step, (single step on line numbers)
LIST	Output display copy to specified file
LOW	Load "window" setup

NOSNOW	Use if you get "snow" on screen using window
P1, P3	Displays or sets P1 Register or P3 Register
PPA	Program Performance Analysis
SAVE (SAV)	Saves user program on file
SAW	Save "window" setup
SECONDS (SEC)	Displays execution timer
SN	Step Next, ("Straight" single step)
STACK (STA)	Displays the contents of the stack
SUFFIX (SUF)	Default radix for entered data, (default HEX)
SWD	Set Window Data
SWR	Set Window Registers
SWX	Set Window eXternal data
SY0A	SY0 input Active high or low
SYMBOLS (SYM)	Display all user defined symbols
SYSTEM (SYS)	Used to execute DOS commands from EMUL51-PC
TM1, TM2	Displays or sets TM0 or TM1
VER	Software version number

Here we would like to point out the usefulness of the LIST command. It can be used to save everything that is written on the screen to a specified file, except windows or trace information. (Use the TDL command, etc., to save trace information.) Then you can use your editor to examine the contents. You can save a disassembly, work on it with your editor, and in this way create a source file from the contents of a PROM.

The INT command is very useful to give you a quick overview of the interrupt status of the processor. The E command provides a built-in symbolic calculator! The rest of the commands are self-explanatory.

For detailed descriptions of each command, please refer to the Commands chapter.

Code Window
Source Level Debugging

The EMUL51-PC lets you open a "Code Window" with the command CW which toggles the CODE Window On and Off. (You can also use CTRL+A to toggle the CODE window on / off. The commands ENA CODE and DIS CODE can also be used to open / close the Code Window).

format:

```
  CW
```

Display Modes When the Code window is open two modes of display are available. By default "assembly mode" is enabled. In this mode the disassembler is used generate the text in the Code window.

The second mode is the "source mode". This mode depends on a combination of the contents of the symbol table and the corresponding source files or list files.

The function key F2 is used to toggle between these two modes. Following are detailed descriptions of how the two modes are used:

Assembly Mode Disassembly is started at the Program Counter (PC). The next instruction to be executed is high-lighted. The cursor is on the command line. The user interface works as usual, so you may enter any command or use ESC to go up to the menu line.

The contents of the code window will follow the program counter (PC) when you use the STEP commands and the GO commands. By pressing the ARROW UP key you can move into the CODE WINDOW. The four leftmost positions in the CODE WINDOW will then be high-lighted. By pressing F9 a breakpoint will be placed at the cursor position and a GO will be generated. By using PgUp and PgDn you can move the contents of the CODE WINDOW to new addresses.

You may also use the mouse to do PgUp, PgDn. Just point to the arrows in the corners of the code window and click the left mouse button. The "down arrows" will scroll the CODE WINDOW one line. If you point in the border between the arrows the window will scroll one page. Above the middle of the border it will scroll up and below the middle it will scroll down.

If you use the disassembly command (D, DASM) you can change the contents to start at any address. (D 500 would disassemble code from 500H and put it in the CODE WINDOW). After STEP or GO the registers will not automatically be displayed as they are when the CODE WINDOW is not turned on.

But you can use CTRL+R to toggle on a window overlaying the STACK WINDOW to see the most common registers. The REGISTER WINDOW on top of the screen is of course updated as usual between each command.

You can redefine some items there to correspond to registers of interest. The Assembly mode is really "mixed" mode as it will display source lines intermixed with assembly if they are available. (See description of Source mode below). For each disassembled line a check is made if there is an active breakpoint from the BRx registers. If that is the case a "comment" is displayed to the right of the mnemonic preceded by a ";". Also a comment is generated for all JMP instructions; either ";jump" or ";no jump" to indicate if the jump will be taken or not.

Source Mode Source mode is usable when you are working with Archimedes/IAR C-Compiler, Keil/Franklin C-Compiler, Intermetrics/Whitesmith C-Compiler, BSO/Tasking C-Compiler, Intel's PL/M-51 compiler or when the "OBJ" file contains LINE numbers and is supported by EMUL51-PC.

Each time the disassembler finds a code symbol (from the symbol table) it examines the symbol and if the first character is a '#' it assumes that it is a line number. It also notes the module name. It will then look for a file with the same name as the module name and with extension ".C" or ".LST", depending on the "SO" command (by default ".C" is assumed). If it is found it will locate the current line number and display that line. In Assembly Mode or for that matter any time the disassembler is used, only "code generating" source lines will be displayed. In Source Mode, however, "non code generating" source lines will automatically be inserted. If you should put a breakpoint (as described in Assembly Mode "ARROW UP") on a "non code generating" line, the breakpoint will automatically be generated on the first assembly line in the next "code generating" source line.

On top of the CODE WINDOW the name of the module currently displayed, is shown.

You can only switch to Source Mode when a source line is visible in the Code window. Therefore if you try to switch to Source Mode when no source line is visible in the Code Window you will stay in Assembly Mode. To get to a point where a source line is visible any of the following methodes can be used:

1. Use PgDn, PgUp or DOWN ARROW until a source line shows up.
2. If you want to show a certain module, first use the DOMAIN (DOM) command to specify the module (DOM ..module name). Then use the Disassembler command (D .#line number in specified module).
3. If you want to "single step" on line numbers, use the L command. This command automatically sets breakpoints on all line numbers, then executes the code until it hits the first instruction generated by a source line.

GS Command The GS (Go Slow) command is very useful both in Assembly mode and in Source mode. The Code window will be automatically updated as the program is executed, giving you a very good picture of what is happening, assembly instruction by assembly instruction or high level statement by high level statement. The "-" key will decrease the speed of the execution, and the "+" key will increase the speed.

In assembly mode the HB ("Highlighted Bar" in the CODE window) always shows the next instruction to be executed. This is in principal also the case in source mode. The problem here is when the sourceline corresponds to only one assembly line. As explained earlier the L command causes breaks at the first instruction in each sourceline. The EMUL51-PC always executes the instruction at the breakpoint before breaking. So if there was only one instruction, the next instruction would be the first instruction of the next sourceline. The HB will be at the sourceline closest to the next assembly instruction. Therefore in cases where the sourceline corresponds to only one assembly instruction the HB will show one sourceline ahead of the "current line" (which was actually already executed). In most C-Compilers this problem can be solved by using a compiler switch which always generates an extra NOP at every sourceline. In PL/M-51 this is not possible so in this case working in mixed mode may be the best solution.

AV Command

The AV (Auto Variable) command lets you view C stack variables when you are inside the function where the variable is "alive." The EMUL51-PC supports different C-Compilers in different ways, depending on what information is supplied by the compiler in the "OBJ" file. NOHAU is continuously updating the software to take advantage of the compiler manufacturer's debug information. This is the latest update:

"C-variable support" available for:

Archimedes:	C-Compiler 3.0 and later. Linker 4.0 and later.
Franklin:	C-Compiler 2.12 and later. Linker 2.4 and later.
Whitesmith:	C-Compiler 3.32 mod 0 and later.
BSO:	C-Compiler 1.1A and later. Assembler 3.0B and later. Linker 5.0A and later.

Generating Correct Object Files

The following is a set of examples showing how correct object files are generated for the EMUL51-PC to support C-variables.

FRANKLIN / KEIL:

Example 1:

Used INSIDE the file:

```
#pragma code symbols debug objectextend
```

Example 2:

Used in batch file:

```
Assembler      ;a51 sample1.a51 debug oe
C-Compiler:    ;c51 sample1.c debug oe
Linker:        ;l51 sample 1.obj to sample
```

IAR / ARCHIMEDES:

For compilation, switches are as usual but either -r0 or -r1 should be added. -r1 provides an extra NOP for each C-line. This is useful for the EMUL51-PC's handling of breaking on the first instruction of the C-line, as the instruction on which the breakpoint is put will also be executed. If you use -r1, be sure to change it to -r0 before burning PROM otherwise the code would be unneccesarily big!

For linking, switches are also as usual but the -F(format) switch is changed to -r

INTERMETRICS/WHITESMITHS:

Example:

```
'C'     c -v -dprom -dxdebug -dlistcs -o sample sample.c
'Asm'   c -v -dprom -dxdebug -dlistcs -o sample sample.s
```

When you are ready to program you PROM remove the -dxdebug switch so that your code with not be unneccesarily big!

BSO/TASKING:

```
Compiler:     cc51 -Ml -s -g sample.c
Assembler:    asm51 sample.src
Linker:       link51 \lib\cstart.obj,sample.obj,lib\c51l.lib to
              sample.abs
Formatter:    oct_ieee sample.abs sample.out
```

When you are ready to program you PROM remove the -g switch so that your code with not be unneccesarily big! Also you can use a different formatting program to produce a straight HEX file.

Watch Window for Source Variables

All models are supported. A WATCH WINDOW can be activated to display up to four C variables. It overlays the REGISTER WINDOW. (Refer to w[x]? command under Source Deb on top menu line.)

There are no restrictions on how local variables should be declared. There is also a way to display and set both local and global variables. This is done with the "?" command. Structures, arrays, arrays of structures and members of structures can be shown. Following are some examples to illustrate what can be done:

Syntax: ?[*..]name[,[#][x | s | d]] [=value] with the details explained as follows:

- no <space> between the '?' and the variable name!
- pointers can be preceeded by an appropriate number of indirections ('*').Only last component of the 'name' need be a pointer.
- 'name' is a simple C-expression for a variable with the following features:
    - nesting of pointers, arrays and structures.

- a single element in an array to be given by a decimal number in brackets.
- no parenthesis permitted.
- the default format can be changed by a trailing:

,#	a repeat counter
,x	hex format
,#x	combination of above
,s	string format
,d	decimal format

otherwise 'floats' are printed as 'floats', 'ints' as decimal 'ints', etc.

- new values can be assigned to simple variables (using the same format for input and output).

Example: C-declarations:

```
struct { int number; char name[8]; } list[3] = {
        1,  "ADAM",
        4,  "DAVID",
        15,  "ROBERT" };

struct tag {
        struct tag *next;
        char  symbol[20];
        } head,*p=head;

int    i = 0x1000;
int    tbl[6] = { 10,11,12,13,14,15 };
int    *ptbl  = tbl;
int    *pptbl = ptbl;
```

```
        Command             Output

        ?i                  4096
        ?i,x                0x1000
        ?tbl[4]             14
        ?*ptbl              10
        ?*ptbl,3            10,11,12
        ?**pptbl            10
        ?ptbl               D:ADDR:
```

where 'D' is memory type and 'ADDR' is hex-value of pointer

```
?list               {{1,"ADAM",4},{"DAVID"},{15,"ROBERT"}}
?list[2].name          "ROBERT"
?list[0].name[3]    'M'
?head.next->symbol  "nextsymb"
?p->next->symbol    "nextsymb"
?*p                 {X:1234,"headsymbol"}
```

Only simple items can be assigned to new values, but it works in complex expressions. e.g.:

```
?list[3].name[2] = 'A'  ( or  = 0x41 )
```

sets a single character in the string 'name[]' which is a member of a struct. 'list' is an array of structs.

Note. The input format is always the same as the output format, but can always be overridden by '0x' for input of HEX code.

```
int *tbl;
?*tbl = 100
but   ?*tbl,4 = 1024
*tbl: {100,2,3,4}
```

assignment ignored; prints 4 elements and ignores the assignment since only single simple item can be assigned.

If a symbol is multiply defined, the local variable is taken before a global variable.
When the watch window is enabled (visible) you may define new watchpoints by typing 'w[x]' before the expression described above. The Watch window is enabled with ENA WAT or Ctrl+W.

> Syntax: w[z]?[*..]name[,[#][x|s]] [=value] where [z] is an optional number to specify in which location in the Watch window the variable is to appear. If the [z] is omitted the current contents in the Watch window will scroll up.

Definition of locations:

1.
2.
3.
4.

Example: w2?tbl

Displaying stack variables using the AV command.

AV can be used with or without parameters. Without parameters all stack-variables will be displayed. With the variable name only the desired variable will be displayed. By default AV displays values in hex bytes. You may, however, instruct AV to display values in any format by adding the first character in the type in which you want the value to be displayed, except that "s" means "string". (See the AV paragraph)

Examples:

```
AV i i          ;display variable i as an integer.
AV i c          ;display variable i as a character.
AV sune f       ;display variable sune as a float.
AV              ;display all stackvariables.
```

How the Standard Trace Works

The trace board makes it possibile for you to record what the microprocessor is doing. However, if everything is recorded you may end up with some useless information. To prevent this from happening, the EMUL51-PC Trace facility lets you set up the trace to record only the information you want. This part of the manual describes how to program the trace, and how the collected information can be displayed and optionally saved to a file. This is applicable only if the trace board is installed and invoked.

Standard Trace Setup You can enter the Trace Setup screen either from the "Trace" item on the top menu line, or from the command line with the TS command. The following description refers to the Trace Setup screen, (see Figure 3).

Six programmable fields at top of the screen control how the A (QRA0-9) and B (QRB0-9) conditions will be used when the trace is collecting data during emulation. The TRACE field controls the filtering mechanism, and by default ALL frames are collected.

Pressing the SPACE bar causes the following options to appear:

A	collect frame only when an A-condition is true
B	collect frame only when a B-condition is true
A & B	collect frame only when both A and B are true

A "true" condition can be described as follows. Each machine cycle of the 8051's execution will present different address, data and port information. We call the information from one such cycle a "frame." One frame consists of these 48 bits: 16 bits address, 8 bits miscellaneous signals, 8 bits data, 8 bits from P1 and 8 bits from P3. For each machine cycle executed by the 8051, the trace board compares the 48 bits with the preprogrammed A- and B-conditions. Actually each condition consists of five (four in V5.6 and earlier) separate fields: ADDRESS, MISC, DATA, P1 and P3. Comparisons are made for each field separately, and the result of the comparison for each field is vertically OR'ed (QRA0 - QRA9). The result for each field is then AND'ed with the other fields (ADDRESS * MISC * DATA * P1 * P3). This is done independently for both the A- and B-conditions. To be true, each "FIELD" comparison has to match exactly with the current data from the active machine cycle. X's denote "don't care," and the corresponding signal can then be either "0" or "1". The "Y" denotes binary.

Values written into the fields without the ending Y are assumed to be hex.

To summarize what has been said so far, the top line of the Trace Setup screen controls what to do with the result of every machine cycle comparison. And as described above, the TRACE field controls the filter mechanism that determines whether or not to save the collected frame.

The next field on the Trace Setup screen is the TRIG field. By default the trigger is off, NOTRIG. Pressing the SPACE bar brings up these additional choices:

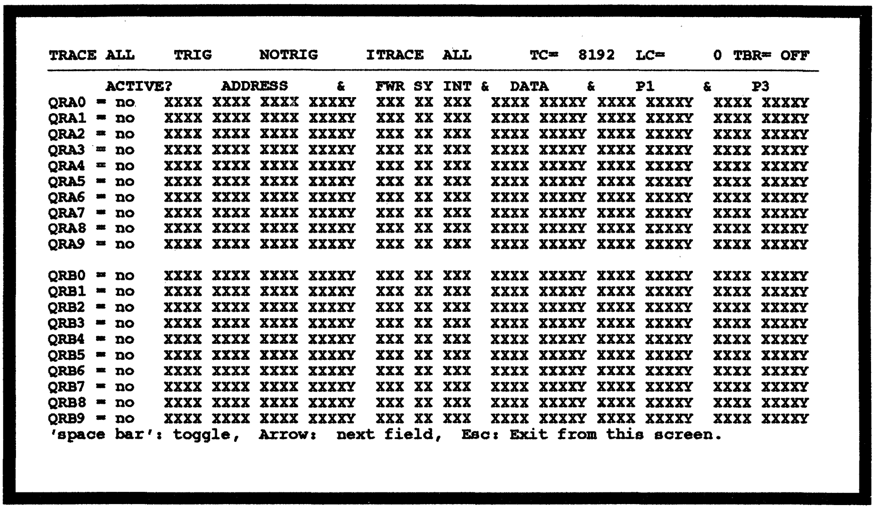
A — Trigger on a true A-condition.

B — Trigger on a true B-condition.

A THEN B — First find a true A-condition, then trigger on the next true B-condition.

A LOOP — Trigger when LOOP number of true A-conditions have been found.

A LOOP THEN B — First find LOOP number of true A-conditions, then trigger on the next true B-condition.



```
TRACE ALL     TRIG      NOTRIG      ITRACE  ALL       TC=  8192  LC=     0 TBR= OFF

         ACTIVE?     ADDRESS      &     FWR SY INT &  DATA     &       P1      &    P3
QRA0 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRA1 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRA2 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRA3 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRA4 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRA5 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRA6 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRA7 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRA8 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRA9 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY

QRB0 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRB1 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRB2 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRB3 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRB4 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRB5 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRB6 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRB7 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRB8 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
QRB9 = no     XXXX XXXX XXXX XXXXY   XXX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
'space bar': toggle,  Arrow:  next field,  Esc: Exit from this screen.
```

Figure 3. Trace Setup Screen

The ITRACE field is similar to the TRACE field, but it controls a different filter function. By default ALL frames are saved. Pressing the SPACE bar causes two more options to appear:

MAIN	Trace frames from the main program only.
INT	Trace frames from interrupts only.

The TC (Trigger Counter) field holds the number of frames to be collected after a trigger is detected before tracing stops. By default this number is 2048 (on a 4K trace board) or 8196 (on a 16K trace board). This means that you will use half the trace buffer for pre-trigger frames and half the buffer for post-trigger frames. Programming a small number into TC would give you more pre-trigger information, while a larger number would give you more post-trigger information.

The LC (Loop Counter) field is used to program the number of loops to be used by the trigger options (A LOOP and A LOOP THEN B).

The TBR (Trace BReakpoint) field is used to connect the trace trigger with the breakpoint logic. By default it is OFF. Pressing the SPACE bar causes it to toggle between ON and OFF.

The trace will automatically start when emulation begins. When emulation is on, you may view the results of a trace, set up new trace conditions, break the trace, and restart it again. When emulation ends, the trace will automatically stop.

Instead of the normal prompt (*), the trace status is shown at the bottom of the screen. When a key is pressed, trace status disappears and you may now enter a trace command. The trace status will reappear as soon as the command is executed. If the ESC key is pressed during trace status, emulation breaks unconditionally.

Trace Display Commands

The following two commands may be used only in emulation mode:

TE Trace End; ends a trace "manually".

TB Trace Begin; starts a new trace. Can be used only if the trace was stopped either by a trigger or by the manual TE command. Works only when the emulator is running.

The collected trace data can be viewed only after the trace has been halted, either by a trigger or by the TE command. After the emulation is halted, it can of course also be viewed, because the trace is automatically halted when emulation is stopped.

The following commands are used to display the trace:

```
TD [start]                        ;displays trace result in disassembled
                                  form.
TDF [start]                       ;displays trace in frame form.
TDL <start> <TO | LEN> <STOP>[-filename]
                                  ;List without pause disassembled
                                  form optional to disk.
TDFL <start> <TO | LEN> <STOP> [-filename]
                                  ;As TDL, in frame form.
TDP [start]                       ;As TD, but displays with Port info
                                  and ext. address.
TDPL <start> <TO | LEN> <STOP>[-filename]
                                  ;As TDL, with Port info and ext.
                                  addresses
TDS [start]                       ;"Slow trace," from file opened with
                                  OST and closed with CST.
TDFS [start]                      ;"Slow trace" in frame format from file
                                  opened with OST and closed with
                                  CST.
```

If [start] is omitted, a default number of frames are displayed (not to exceed one screen). PgUp can then be used to display more information.

Observe that if you use the filter (TRACE field), you must be careful with the TD, TDL, and TDP display commands. This is because the disassembler is used on the information in the trace buffer. Therefore if frames are missing from instructions, the disassembly may be misleading. If, for instance, you program the filter so that only one address is collected (over and over), and use the TD command to display the result, the message **** PROCESSING INTERRUPT **** will be shown. This happens because the address following the F frame, (First byte of an instruction), is the same as the F frame. This normally indicates an interrupt and would happen if the address you traced was in fact an F frame. Otherwise the TD may not show anything. In other words, TDF and TDFL should be used here!

For the exact format of each command, refer to the detailed descriptions in the Commands chapter.

Trace Memory

The trace memory is 48 bits wide and 4K deep (or 16K deep if you have that option). One frame, (48 bits), consists of the following:

- 16 bits address
- 8 bits miscellaneous (VF, (Valid Fetch), WR (WRite to external memory), RD (ReaD from external memory), SY1, SY0, INT2, INT1, INT0. (The last three bits indicate interrupt level.)
- 8 bit data
- 8 bits from P1
- 6 bits from P3
- 2 extra bits, (marked E0 and E1 on the pod). On some pods these are the other two bits of P3.

If P1 and/or P3 do not need to be traced, remove the jumpers on the pod. That means that 18 of the trace inputs can be used to trace external signals. They are easily connected with the optional EZ-hook wires.

The lower 8 bits of the address are sampled on the trailing edge of ALE. The higher 8 bits of the address and P3 are sampled on the leading edge of PSEN, READ or WRITE. P1, data and the 8 miscellaneous signals are sampled on the trailing edge of PSEN, READ or WRITE.

Frame Data

The display of frame data is organized in the following way:

Left-most column:	Frame number
Second column:	F = Valid fetch frame R = External memory read frame W = external memory write frame - = none of the above

Third column: Address in hex

Fourth column: Data in hex

Fifth column: Interrupt level (0 is main program)

Sixth column: Port 1 / Port 3

Seventh column: SY1, SY0, E1, E0 (WR and RD on some pods)

Eighth column: Machine cycle count. The Machine cycle count can be regarded as a "time stamp". If a 12 MHz clock or crystal is used, "machine cycles" are equivalent to microseconds. If the filter function is used, the "count" is only accurate on a relative basis within a block of trace frames with contiguous addresses. In other words: the counter does not recognize "gaps" in the trace caused by "filtering out" frames.

For more information on the Standard Trace please refer to Tutorial Chapter 6, Module 3.

How the Advanced Trace Works

The Advanced Trace Board (ATR), was designed based on customer feedback from users of our regular trace board. Here is a list of its features:

1) 64k "frames" deep. Optionally 256k deep.

2) 64 bits wide (16 bit address, 8 bit data, 2 x 8 bit Port data, 8 miscellaneous signals and 16 bit timestamp).

3) Time stamping with up to 34 bit resolution; 16 bit prescaler.

4) Multi-level trigger mechanism with each level consisting of boolean equations of events, counter, and "states".

5) Ratio of pre- and post- trigger information fully programmable within the 64k (256k) trace buffer.

6) Filter mechanism using boolean equations of events, counter and "states".

7) Programmable post filter.

8) "Break when Trace Done", "On Trig", and "On S2" functions.

9) Search function on frame within the trace buffer.

10) 32 bit counter / timer. It counts when a Boolean function of events, counter and "states" is TRUE.

11) 16 bit loop counter whose output can be used in boolean functions.

As the name "Advanced Trace" indicates, this trace function can be used for some very advanced debugging. To take full advantage of all functions, the user must also be "advanced". However, most users may never need to make use of all the features which are described below, together with a definition of terms and some examples. Please refer to the Tutorial Chapter for additional examples.

Definitions of Terms

<frame>

64 bits of data collected every memory cycle. Consists of 16 bit address, 8 bit data, 2 x 8 bit port data (or external signals), 8 miscellaneous signals and 16 bit timestamp.

<event>

Once every memory cycle, a **frame** is compared with eight events, A - H. Each comparison will result in TRUE or FALSE. An event consists of 48 bits (the timestamp is not included in the frame). As shown below, each bit can be programmed as a '0', '1' or 'x' (don't care). Ranges, hex, or binary are also allowed. Binary is denoted by a trailing 'Y' for binarY.

<Boolean expression>

A Boolean expression consists of a number of "Boolean operators" working on "Boolean operands". The Boolean operators are:

AND, OR, NOT

- AND between two operands means that both must be TRUE for the result to be TRUE.
- OR between two operands means that one must be TRUE for the result to be TRUE.
- NOT before an operand means that the operand must be FALSE for the result to be TRUE, or the operand must be TRUE for a result to be FALSE.

- THEN between two operands means that the first operand must come true first, then the second operand follows sometime thereafter. This command is only allowed in the **TRIG** and the **RECORD** fields.
- SET-CLEAR This function will allow you to tell the trace when to start recording on a given condition (SET), and then to stop recording (CLEAR) on a different conditon.

The hierarchy of the Boolean operators is as follows:

NOT, /	Highest
AND	.
OR, THEN	.
SET-CLEAR	Lowest

APPLICATION NOTE: The Set-Clear function will have a delay of a single memory cycle when the SET becomes true until it is actually recorded in the trace buffer. To capture the actual SET frame or first byte of the operation, you need to use something like the following example:

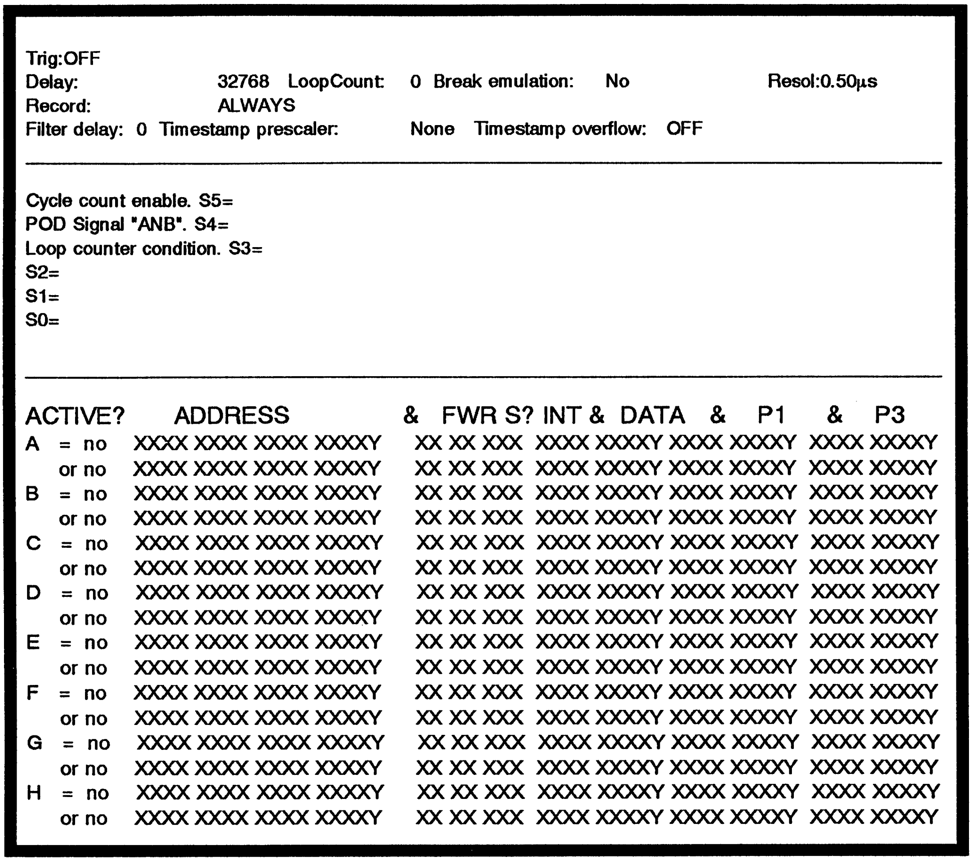
```
RECORD: Yes, if S0 or A
S0 = Set A Clear B
```

The Trace Setup screen (TS)

The TS screen is entered either from the "Trace" item at the top menu or from the command line with the TS command.

Following is a list of the program fields in the TS window:

Trig
Delay
LoopCount
Break emulation
Record
Filter delay
Timestamp prescaler
Timestamp overflow



```
Trig:OFF
Delay:              32768  LoopCount:  0 Break emulation:    No            Resol:0.50µs
Record:             ALWAYS
Filter delay: 0 Timestamp prescaler:   None  Timestamp overflow:  OFF
________________________________________________________________________________

Cycle count enable. S5=
POD Signal "ANB". S4=
Loop counter condition. S3=
S2=
S1=
S0=

________________________________________________________________________________
ACTIVE?    ADDRESS               & FWR S? INT &  DATA   &    P1     &    P3
A  = no  XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
B  = no  XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
C  = no  XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
D  = no  XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
E  = no  XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
F  = no  XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
G  = no  XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
H  = no  XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY    XX XX XXX  XXXX XXXXY XXXX XXXXY XXXX XXXXY
```

Description of "Program Fields" These "program fields" define an event or events that control the trace capture. A more detailed description of a "true event" will follow after the description of the "program fields".

Trig The Trig field is either "OFF" or "Yes, if". You can toggle between these two choices with the space bar. The "Yes, if" condition is considered the 'ON' state of this field. The field to the right of "Yes, if" should be a Boolean expression operating on the qualifying registers (A-H), S0 - S5 and the output of the loop counter CO (loop COunter done). The Boolean expression may also contain "THEN".

Examples:

```
A
NOT A
A OR B
A AND B
A THEN B
A AND NOT B
A THEN B THEN C THEN D THEN E THEN F THEN G THEN H
A OR B THEN C AND NOT D OR E
A OR S3
A AND CO
A THEN B OR S3
A THEN B THEN CO AND C
```

Delay When emulation starts, the trace starts tracing automatically. When the trace triggers according to the Trig, it continues to trace for the number of cycles programmed in the Delay field. This means that if you have a small number in Delay, the trace will stop near the trig point. If you have a larger number, the trace will continue for a longer time before it stops. The number can be between 2 and 2,147,483,647 (7FFFFFFF in Hex, 32 bit counter). This means that the trig point can be outside the actual trace that you are able to display. The default value of Trig Delay is half of trace memory size (32768 for the 64k ATR board).

Loopcount This field determines how many times S3 must count before CO becomes TRUE. The number in this field can be between 0 and 65,535. The default is 0. See "Loop Counter Event" for further details.

Break Emulation This field can be toggled between "No" (default), "On Trig", "When Done" and "On S2". "No" means that the trace will not affect emulation. "On Trig" means that a breakpoint will be forced when the trig occurs. "When done" means that a breakpoint will be forced when the trace stops, therefore the break will occur "Delay" frames after Trig. "On S2" means that the break will occur when S2 becomes TRUE.

Record The Record field can either be "ALWAYS", "Yes, if" or "Source Only". You can toggle between these three choices with the space bar. The field to the right of "Yes, if" should be a Boolean expression operating on the *events* A-H, S0 to S5 and CO. If "ALWAYS" is chosen, all frames will be recorded. If "Yes, if" is chosen, only frames that are TRUE according to the Boolean expression will be recorded. If "Source Only" is chosen, only frames that have a corresponding line number will be recorded. When the trace is displayed later, high level statements from the source file can be shown. Since only one frame for each statement is recorded, up to 65535 source lines can be recorded! (262,143 if you have a 256k trace.) The "THEN" operator is not allowed here.

Examples:

```
A
A OR B
A AND B
S3 (where S3 could be SET A CLEAR B to record everything between two events).
```

Filter Delay When Record is in "Yes, if" mode, frames will not be recorded when the Boolean expression is FALSE. However, in some cases you may want to continue to record a number of frames after "Record" goes from TRUE to FALSE. This is done by entering a number between 0 and 15 in the "Filter delay" field. This is particularly helpful when recording MOVX instructions. The third frame of a MOVX instruction emits an address to the location where XDATA is read or written. This address can be outside the "Record expression", but if you have entered a 1 in the "Filter delay" field the read / write external data will be recorded. Another use of this feature is to continue to trace after conditional jumps in the code.

Timestamp To view the timestamp, please use TDF or TD commands. By default, timestamp is shown in cycles. F4 toggles to show time in absolute or relative mode. The XTAL command is used to set the crystal rate which will calculate the time. By default, XTAL is 12.0000 MHz. You may also use the x parameter when you invoke the emulator to change the crystal frequency:

```
EMUL51 -p8031 -m128 -e110 -t100 -f64 -x11.23456
```

indicating that a XTAL frequency of 11.23456 is used.

Timestamp Prescaler The timestamp mechanism consists of two 16-bit counters. Only the 16 bit value from one counter is recorded in the trace. The second counter is the prescaler. If the prescaler is none, you have full accuracy, which has a resolution of half a cycle. Therefore, you can count up to 65,535 "half cycles" or 32,767 cycles before the counter rolls over to 0. With a 12MHz crystal this is equivalent to 32.767 ms. With the prescaler programmed to 2, you would be able to count double the amount of time, or 65,535 ms, but with half the resolution; i.e., 1 cycle or 1 ms. With the prescaler at 65,535, you would count 65,536 x 65,535 half cycles or for approximately 2,147 seconds (or 35 minutes) at the resolution of 32 ms. The default is none. Integers between 1 and 131070 are accepted. The prescaler can only use *EVEN* integers. For example, if you enter a 3 the program will round down the number to 2 (1 to none, 5 to 4, 99 to 98, etc.)

Timestamp Overflow This field can be either "OFF" or "ON". When it is "ON", extra frames may be recorded when the timestamp counter rolls over from FFFF to 0. This will only happen when the RECORD field has determined that no frame will be recorded. In other words, a frame is forced to be saved in order to collect a timestamp at the rollover point. If no frames are recorded (due to the RECORD field), 64k of "overflow" frames can be recorded (256k with the ATR256).

Let's say that you want to find out how long it takes between two addresses. Assume that the first address was not encountered again until the second address was encountered. With full resolution (Prescaler = none), you would be able to record up to 65,535 - 2 "overflow" frames. Since the time between two "rollovers" at full resolution (prescaler = none) is about 32ms (at 12MHz), up to 35 minutes (32 x 65533 / 1000 x 60) can be recorded with a resolution of half a cycle (500 ns). With the 256k ATR, this can be four times more, or almost 2.5 hours. With less resolution you could go for years!

Since it takes a while to calculate the timestamp, there will be a delay of several seconds between issuing a trace display command until the display acttually appears on the screen.

Cycle Count Enable. S5= This field consists of a <Boolean expression> of <events>, S0 to S5 and CO. When the expression is true, a 32 bit counter is enabled to count cycles. This can be used to measure cycles or time "on the fly". This is displayed during emulation in the Trace Status Window (described later in this document). On PODS with the FLF pin, the S5 state is available. It can also be found on pin 22 of the 50 pin ribbon cable connector.

After emulation is done, the SEC command will show total cycles counted during emulation (counter on the emulator board), the "ATR cycle counter", and a ratio of the two. (Refer to the SEC command description about how SEC works.)

Example:

Using the TEST.A03 program, we want to determine what percentage of the total time is spent in the interrupt routine. Use default settings in the TS window except as follows:

Cycle count enable. S5= NOT A

```
    ACTIVE?       ADDRESS       & FWR SY INT &    DATA     &      P1       &       P3
A  = YES  XXXX XXXX XXXX XXXXY  XX XX 000Y  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
```

Note that the INT bits are 0.

```
* res chi
* g from 0
```

Let the emulation run for about 5 seconds, then press the ESC key.

This is the result (depending on when you press the ESC key):

```
* sec
18836603 cycles  TIMER 11986923 microsec  RATIO 63.636331%
```

POD Signal "ANB". S4= This field consists of a <Boolean expression> of <events> and S0 to S5 and CO (loop count). When the expression is TRUE, the ANB pin of the POD board is low. (ANB is not available on all PODs, but this signal can be found on pin 21 of the 50 pin ribbon cable connector.)

Loop counter condition. S3= This field consists of a <Boolean expression> of <events>, S0 to S5 and / or qualifiers A-H that are used to describe how the loop counter will operate. When the condition goes from FALSE to TRUE, the loop counter will count down.

S2=
S1=
S0=

These fields can only be used if the "THEN" operator is not used in the "Trig" field above. If "THEN" is used, these fields will implement the "state machine" to handle the "THEN". If "THEN" is not used in the "Trig", S0 - S2 can be freely employed as Boolean expressions utilizing events and S0 to S5 and CO (loop count). You could then implement your own "state machine" of up to eight "states". If you don't use S3 - S5 for loop counter or pod signal, they can be employed for other conditions that you wish to set up. In this case, you could implement a state machine of up to 64 states.

How to Build a "State Machine" Using the Sx Functions

Here is an example showing how S2 - S0 can be used as a three bit counter (8 states) to count up each time the A Event equals a First Opcode fetch (FO) at address 200H. In this case, we use the result of S2 - S0 to trigger the trace when S2, S1 and S0 are all TRUE. You can expand this example by using more events (A - H) in the equations. The result could be used to turn on and off the RECORD field.

Use TEST.A03 where address 0 is the first address in the interrupt service routine. The trace setup fields are left in their default values except the following:

```
Trig:Yes, if (S0 AND S1 AND S2 )
S2=S2 and /(S0 and S1 and a) or /S2 and (S0 and S1 and a)
S1=S1 and /(S0 and a) or /S1 and (S0 and a)
S0=(S0 AND /a) OR (/S0 AND a)
```

The expressions in S0, S1 and S2 could have been written using the Boolean XOR, but the XOR is not implemented. The equivalent expressions are:

```
S2=S2 xor (S0 and S1 and A)
S1=S1 xor (S0 and A)
S0=S0 xor A
```

```
  ACTIVE?    ADDRESS     &  FWR S? INT &  DATA    &    P1     &    P3
A  =  YES           200   01 XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
```

Example of how to use SET CLEAR with the Sx functions:

```
S0= SET A CLEAR B
```

This means that S0 will be set to TRUE when A becomes TRUE. It will then stay TRUE (even if A becomes FALSE) until B becomes TRUE at which time it becomes FALSE. A and B in the example above may be Boolean expressions. This can be used in a number of ways but here we will show an example where it is used to create a "window" for recording.

The trace setup fields are left in their default values except the following:

```
Record:  Yes, if   S0 or a                                   S0=set a clear b

 ACTIVE?    ADDRESS     &  FWR S? INT &  DATA    &    P1     &    P3
A  =  YES                   10C  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no   XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
B  =  YES                   10E  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
```

The reason A must be in the RECORD expression is that the result of SET A will appear in S0 delayed for one frame, otherwise you would miss address 10C.

The outcome is that all activities taking place between execution of address 10C and address 10E will be recorded. This is obviously different from saying that all addresses between 10C and 10E will be captured. If, for example, there is a subroutine call between 10C and 10E, it will be recorded even if the addresses of the subroutine are outside the range.

If SET / CLEAR is used under S5, you get a convenient way of measuring exactly how long it takes to execute a certain portion of your code!

Example of a "six bit state machine" using SET / CLEAR and demonstrating how the loop counter works and how the cycle counter works:

In this example TW2.HEX is used. This program consists of a one second delay starting at address 0, then a number of NOP's up to address 200 where a jump back to 200 is done (LJMP 0). Then, in the TS Window the A event is set to be address 200.

```
* LOA TW2.HEX
* TSG TW2.TSG
```

When you watch the TS window, you will see that the expressions of S5, S4 and S3 do not fit on the screen. You have to go into the field and use the arrow keys to see the complete expression.

Run the program:

```
* RES CHI
* G
```

Since S3 (loop counter event) is now exercised, we have programmed the loop counter to 10. Watch how the loop counter is decremented each time S3 goes from 0 to 1. In the STATES Window, the bits displayed are from the most significant bit (MSB) on the left, to the least significant bit (LSB) on the right.

This example is shown just to help you see how the state machine can work as a counter, and is an expansion of the previous example with the three bit state machine. There is no reason that you really need to analyze how each of the equations in the (S) registers affect the other.

S5 is also exercised, so when S5 becomes 1 the cycle counter will start to count. Since the delay in TW2.HEX is about one second, it will take about 64 seconds for all six states to all become 1. Half of this time S5 is 1, therefore the cycle counter will reach 32,xxx,xxx before trigger occurs. (TRIG is AND'ed between all six "S" bits.)

Event Fields

Active ? These fields are utilized to activate any of the qualifying registers A-H.

Address Used to specify an address, address range, or a wild card address to qualify the capture for the trace buffer.

FWR This column is employed to specify if the address is a valid fetch (F), write (W), or a read(R).

_ F = 01
_ R = 10
_ W = 11
_ None of the above = 00

SY This column is used to specify logic HIGH (1) or Low (0) on the SY1 and SY0 pins, respectively.

INT This column specifies the interrupt level.

000	interrupt level 0. (Main)
001	interrupt level 1.
010	interrupt level 2.
011	interrupt level 3.
100	interrupt level 4.
101	interrupt level 5.
110	interrupt level 6.
111	interrupt level 7.

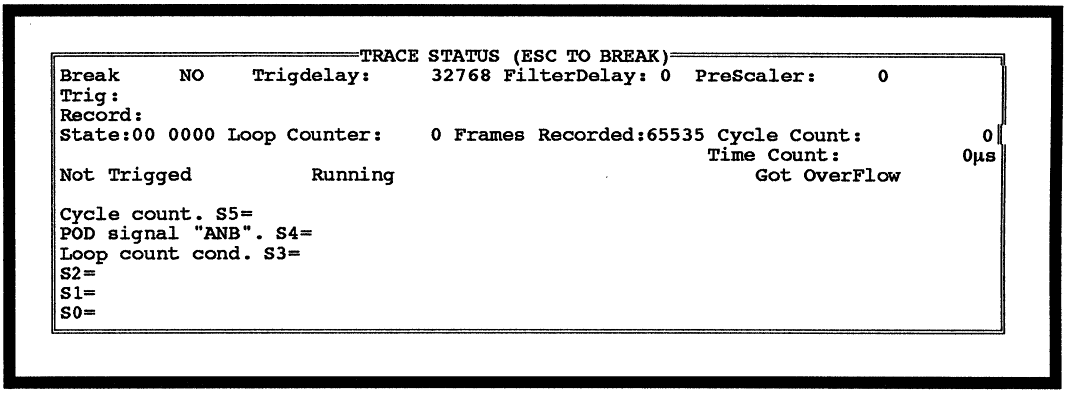
DATA This column specifies the DATA value.

P1 Here you specify the value on the probes labeled P1 (by default Port 1). These pins on the POD can also be removed and connected to points on the target system.

P3 This field is like P1, except for bits 6 & 7, (which on most PODS are for the inputs E0 and E1, respectively).

The lower 8 bits of the address are sampled on the trailing edge of ALE. The higher 8 bits of the address and P3 are sampled on the leading edge of PSEN, READ or WRITE. P1, data and the 8 miscellaneous signals are sampled on the trailing edge of PSEN, READ or WRITE.

The Trace Status Window: The trace status window will always appear while emulating if the trace board is installed and both it and the emulator are invoked. Now we'll take a closer look at the Trace Status Window that appears at the top of the screen when the emulator is emulating (see Figure 4).



```
══════════════════════════TRACE STATUS (ESC TO BREAK)══════════════════════════
Break      NO     Trigdelay:      32768 FilterDelay: 0  PreScaler:      0
Trig:
Record:
State:00 0000 Loop Counter:    0 Frames Recorded:65535 Cycle Count:          0
                                                          Time Count:        0µs
Not Trigged           Running                                Got OverFlow

Cycle count. S5=
POD signal "ANB". S4=
Loop count cond. S3=
S2=
S1=
S0=
```

Figure 4. Trace Status Window

Labeled fields in the **TRACE STATUS WINDOW** have the following meanings:

Break: Tells setting of the break emulation register.

Trigdelay: Shows the number of maximum POST trigger frames.

FilterDelay: Shows the number of frames to be captured after the filter condition(s).

PreScaler: Shows if a prescaler value has been set for the TIME STAMP function (to be covered later).

Trig: Shows the equations for trace trigger.

Record: Shows what the filter condition is.

State: Shows the status of the STATE machine.

Loop Counter: Shows the status of the LOOP COUNTER.

Frames Recorded: Shows the number of frames that have been recorded.

Cycle Count: Shows the number of cycles executed. Set by S5 in the trace setup.

Time Count: Shows the execution time that is conditioned by (Cycle count) S5.

See figure 4. The fields showing the *Not Trigged, Running,* and *Got OverFlow* are normally shown as a blank highlighted field and will be displayed if a trigger has happened, if the trace is running or stopped, of if the TimeStamp counter has gone into overflow.

Cycle count. S5= Shows the condition that has be set up for the *Time Count:* field.

POD signal "ANB". S4= On some pod boards there is a pin that has the same label. This field can be set up with some qualifier to produce an output signal that can be captured on an oscilloscope.

Loop Count cond. S3= Shows what condition has been set up for the loop count qualifier.

S2, S1, S0 General registers that can be used for triggering or filtering the trace.

How to Edit Trace Setup Fields In the trace setup (TS) screen, there are some special keys that are used for movement and editing fields as shown on the following list:

Ctrl-Home	Goto top field in trace setup screen.
PgUp	Same as Ctrl-Home

Ctrl-End	Goto bottom of trace setup (event H).
PgDn	Same as Ctrl-End
Arrow UP	Goto previous field.
Arrow DOWN	Goto next field.
Insert	Toggle, insert or overtype mode. Insert mode cursor = flashing underscore Overtype mode cursor = full flashing cursor
Enter	Enter the change and/or goto next field.
Tab	Goto next field
Shift+Tab	Goto to previous field

What is a "true" event? Each memory cycle of the 8051's execution will present different address, data and port information. We call the information from one such cycle a "frame". One frame consists of 16 bits address, 8 bits data, 8 bits miscellaneous signals, and 8 bits from P1 and 8 bits from P3 for a total of 48 bits. The 16 bits from the time stamp are not involved in an EVENT. For each machine cycle executed by the 8051, the trace board compares the 48 bits with the preprogrammed A through H events. Actually, each event consists of five separate fields -- ADDRESS, misc, DATA, P1 and P3 -- and compares are done for each field separately. The result of the compare for each field is vertically OR'ed (QRA0 - QRA1). The result for each field is then AND'ed with the other fields (ADDRESS & miscellaneous & DATA & P1 & P3). This is done independently for all the A through H-events. Each "FIELD" compare has to match exactly with the current data from the active machine cycle to be true. X's denote "don't cares" and the corresponding signals can be either "0" or "1". The "Y" denotes binary. Values written into the fields without the ending Y are assumed to be hex. (See "How to Edit Trace Setup Fields" above.)

Trace Display Commands There are two commands that may be used only in emulation mode. They are:

TE, Trace End. Ends a trace "manually".

TB, Trace Begin. Starts a new trace and can be used only if the trace was stopped either by a trigger or by the manual TE command. Works only when the emulator is running.

The collected trace data can be viewed only after the trace has been halted, either by a trigger or by the TE command. The trace is also automatically stopped and can be viewed after emulation is halted.

The following commands are used to display the trace:

```
TD    [start]  ;displays trace result in disassembled form.
TDF   [start]  ;displays trace in frame form.
TDL   <start> <TO | LEN> <STOP> [-filename].
               ;List without pause disassembled form optionally to disk.
TDFL  <start> <TO | LEN> <STOP> [-filename]
               ;As TDL. In frame form.
TDP   [start]  ;As TD but displays with Port info and ext. address.
```

If [start] is left out, a default number of frames is displayed (not to exceed one screen). PgUp and PgDn can then be used to display more information.

Observe that if you use the filter (RECORD field), you must be careful when you employ the TD, TDL, and TDP display commands. This is because the disassembler is utilized on the information in the trace buffer. Therefore, if frames are missing from instructions, the disassembly may be misleading. For instance, if you program the filter so that only one address is collected (over and over), and use the TD command to display the result, the disassembly may not show any information or the display may not make any sense. In these cases, the user must employ the TDF and TDFL commands.

Other commands that affect the operation of the trace displays are as follows:

```
TSD   [=ON] [=OFF]          ;This command is used to turn on/off
                            the timestamp feature of the the trace
                            board.
```

XTAL [=Clock Frequency in MHz] ; This command is used to adjust the display of the TIME COUNT in the trace status window relative to the micro's clock speed.

For the exact format of each command, refer to the detailed description for the command.

Trace Display The display of frame data is organized in the following way:

Leftmost column:	Frame number	
Second column:	F =	First Op Code Fetch frame
	R =	External memory read frame
	W =	external memory write frame
	- =	none of the above
	O =	Frame was forced to be stored. Indicates that the timestamp counter overflowed.
Third column:	Address in hex.	
Fourth column:	Data in hex	
Fifth column:	Interrupt level. 0 is main program.	
Sixth column:	Port 1 / Port 3	
Seventh column:	SY1, SY0, E1, E0	
Eighth column:	Time stamp (CyclesA, CyclesR, TimesA, TimesR, use F4/F5 to shift between them)	

25th Line Prompt (Control Keys) On the bottom of the trace dispay screen a highlighed bar shows the key options that can be used. The following is a description of those keys:

➤ PU	= PgUp
➤ PD	= PgDn
➤ ↑↓	= Arrow keys (up, down)
➤ -dig	= enter a negitive or positive frame number. (goto FRAME#)
➤ F2	= Switch betweens high level source or mixed mode dispaly.
➤ F3	= Switch from mixed (source+assembler) to frames (TDF).

➢ F4 = Switch timestamp display between **absolute** to **relative**.
➢ F5 = Switch cycle count between **absolute** to **relative**.
➢ F6 = Pop up search setup window.
➢ ESC = ESC key to exit display.

Breakpoints

Emulation is started with the GO command; it can be ended in a number of ways.

1. You can press the ESC key. Doing this will immediately stop emulation and allow access to the monitor.

2. A previously set up breakpoint is encountered. A breakpoint can only be set up to break on addresses. It can break on three kinds of addresses: Opcode address, Read external memory address, and Write external memory address. Breakpoints are programmed in the BRx registers, where x is 0 - 9. (See BR for details.) The BRM register holds the address type(s) that the breakpoints work on. (See BRM for details.) Breakpoints are activated either in the GR (Go Register) register, by setting the GR, or inside the GO command. (See GR and GO.)

3. An external signal SY0. It works in a number of combinations with breakpoints, set up in GR. SY0 can be programmed active high or active low. This is set up in the SY0A register. (See SY0A.)

4. A trace condition. With the TBR register you can use the trace conditions to activate a breakpoint. This means that you can use any combination of the 48-bits wide trace frame to trigger a breakpoint. (See TBR). Observe that all 48 bits are used. If, for instance, you want to trigger on an address, you will probably want to set VF (Valid Fetch, indicating first frame of an instruction) to a "1." Because the trace is delayed a couple of PSEN cycles, the actual break will occur one instruction after where the trace condition was set. This means that the PC (program counter) will show two instructions after the trace condition that caused the break.

5. If you happen to write a 0 to RD (P3.7) or WR (P3.6) in external mode pods, a break point is forced. An error line "BREAK OCCURRED BECAUSE RD OR WR IS LOW" is displayed as the first line after the break occurred.

6. The L command sets up breakpoints on all LINE numbers defined in the symbol table. The code is then executed until any of the breakpoints is encountered. Previously activated breakpoints in the Breakpoint RAM are erased before this command is executed. Used

in PL/M-51 and C-51.

7. The BB command sets up breakpoints on all locations where a specified bit address is written to. Use the GB command to execute.

8. The BYB command does the same where a specified byte address is written to. Use the GB command to execute.

9. The iB command lets you specify a pattern of three bytes where a breakpoint will be programmed. The first byte should be the hex number for an instruction OP-code, while the second and third bytes can be either numbers or X (don't care). Use the GB command to execute.

10. BIC command will set up breakpoints on all instructions which may possibly alter the contents of a specified address to a specified value or range of values. The breakpoints set up with BIC should be used in conjunction with the GI command, which will use the breaks to check if the specified condition is reached. If not, the program will resume. Each break will steal approximately 200 microseconds from the program before it returns. If the condition was reached, a regular breakpoint will be taken.

11. The BB, BYB, IB, and BIC commands may be used to implement many "overlayed" breakpoint patterns in the breakpoint RAM. The breakpoint RAM is always cleared when a GO or an L command is executed. It can also be cleared with the CLB command.

12. The SEB, ABR and GB commands are used to trap the cause of a program that "jumps out of bounds". (See the ABR paragraph.)

Observe that the instruction where the breakpoint is set will also be executed before the break occurs. This means that the PC will show the next instruction to be executed, (except TBR). (See 4, above.)

Timers

On external mode pods, if the timers are active when a break occurs, they will keep counting for a few cycles before they are stopped by the emulator software. When emulation is then started again, the timers will be turned back on by the emulator software a couple of cycles before the actual emulation begins. The number of cycles lost is shown below:

	T0, T1	T2
EMULATION TO MONITOR:	8 cycles	15 cycles or 15 x 6 (in baud rate generator mode)
MONITOR TO EMULATION:	6 cycles	15 cycles or 15 x 6 (in baud rate generator mode)

If this is a problem, you will have to manually adjust the timers with these numbers. A macro could be used to do this.

These timer delays and prestarts do not occur in "bondout" pods, nor in most "hooks" pods in internal-code mode.

Macros

Macros are used to automate the emulation commands. A macro consists of a number of commands which, instead of being input from the keyboard, are read from the macro block. Macros can be saved on a file and later read back into the emulator. They are extremely useful when you need to repeat the same sequence of commands over and over again.

Another way to repeat and store command sequences is to write them in a regular ASCII file and then use the INCLUDE (INC) command to execute them. This is also normally faster than macros. (See the INC command for details).

To define a macro you use the DEFINE command. See the DEF paragraph for details. If you create more complex macros it is best to use an editor to create a regular ASCII file with the macro definition. If you use the SYSTEM command this can be done without leaving the emulator. When you have edited your macro setup you can load it into the emulator with the INC command.

Example: (This could be the contents of a file, except the "*" and ".*")

```
*DEF :R                     ; This defines a macro called R.
.*RESET CHIP                ; This is the first command in the macro.
.*GO FROM .START TO .STOP
.*TD -100
.*EM                        ; this ends the macro.
*
```

To run the macro, you simple enter the following:

```
*:R
```

Logical Operators and Macros

The following is a table of logical and mathematical operators that the EMUL51-PC understands. These can be used on the command line or inside a macro.

Mathematical

+	Add
-	Subtract
*	Multiply
/	Divide

Logical

>	Greater-Than
<	Less-Than
<>	Less-Than or Greater-Than
EQ	Equal-To
<EQ	Less-Than or Equal-To
>EQ	Greater-Than or Equal-To
AND	Logical AND operation
IOR	Inclusive OR
XOR	Exclusive OR
NOT	
MOD	Remainder of Division function

The macro can be made more flexible by including parameters. You may use up to 10 different parameters in each macro. The parameters are entered into the macros with the % character followed by a number from 0 to 9. Here is an example.

```
*DEF :FLEX
.*XBYTE %0 = %1
.*GO FROM %2 T %3
.*XBY %4 L 10
.*EM
*
```

Because we used five parameters in the macro (%0 - %4), we must supply the macro with these five parameters when we execute it. The parameters must be separated by commas, as shown in the example below.

```
*:FLEX 1000,23,.START,.STOP,2000
```

In the above example, the %0 corresponds to 1000, the %1 corresponds to 23, and so on. Each parameter can be used any number of times inside the macro, and the parameters may appear in any order inside the macro.

If you use %np in a macro, it will be replaced with a number indicating the number of parameters (np) that were fed to the macro.

Instead of creating the macros with the DEFINE command in the emulator, you can use your editor to edit the "DEFINE block" in a regular ASCII file. It can then be loaded into the emulator with the INCLUDE command. This is the preferred method, especially if you create more complex macros, since the macro is easy to correct if it doesn't work immediately. Even if you do create the macro directly in the emulator with the DEF command, you can save it to a file with the PUT command. After that, you can edit it, (using the SYSTEM command to invoke the editor without leaving the emulator), maybe use the REMOVE command to remove the old macro, and then INCLUDE the edited macro. An example is given below.

```
* DEF :M                   ;defining a macro in the emulator
.* LOA TEST.A03
.* RES CHI
.* GO FOREVER
.* EM                      ;end macro
* PUT A:START.MAC :M       ;save macro to a file
* SYS EDIT A:START.MAC     ;edit macro
```

(editing the macro SYS EDIT A:START.MAC).

```
* REM :M                   ;remove old macro
* INC A:START.MAC          ;load the new edited macro
```

Comments in a macro are allowed. Any line that starts with /* is treated as a comment and the command interpreter ignores it.

Example:

```
/* This is a comment line.
DBYTE 56 = 45
G F 0 T 1234
/* Another comment
```

The WRITE command is often used in a macro to have text mixed with numbers printed to the screen. It is possible to format the printing of the numbers to the screen: We use the same conventions as are used in the "C" language, except that the "%" sign has been replaced with the "~" sign (tilde).

The format string is placed after the number to be formatted. Here are two examples; for more details about the WRI command, refer to the Commands chapter.

```
WRI 'This is an integer: ', XBY 1000,~d
WRI 'This is a 4-digit hex number with leading zeros: ',DB 26,~04X
```

To list macros use the DIR command. See the DIR paragraph.

To save a macro use the PUT command. See the PUT paragraph.

To load a macro use the INC command. See the INC paragraph.

To enable or disable macro expansion, please refer to the ENA and DIS paragraphs.

Macros can be nested up to seven levels deep.

Macros can be programmed to form their own small programs using IF, ORIF, THEN, ELSE structures, or REPEAT, WHILE, UNTIL, or COUNT, WHILE, UNTIL. For details, see IF, REP, and COU in the Commands chapter.

Macro Examples These examples are shown in a MACRO form, but remember that you can do the same on the command line of the emulator.

Macro Example 1:

This macro is an example of reading some values from the emulators resources, i.e. register, Xdata RAM, Internal data RAM, and processing a division operation. Then storing the answer into the processor's RAM.

```
Define :m1
/* read xdata location 4000h
def .x4000 = xby 4000
/* read data location 30h
def .d30 = dby 30
/* find answer and store it at Xdata 1000
xb 1001 = .x4000 / .d30
/* and store remainder at Xdata 1001
xb 1000 = .x4000 mod .d30
/* remove user defined symbols
rem sym .x4000
rem sym .d30
em
```

Macro Example 2:

This is the same as the above macro but instead of storing the value to the processor's memory we will print it to the screen.

```
Define :m1a
/* read xdata location 4000h
def .x4000 = xby 4000
/* read data location 30h
def .d30 = dby 30
/* find answer and save to a new variable
def .answer = .x4000 / .d30
/* and store remainder in another new variable
def .remainder = .x4000 mod .d30
write 'Answer = ' .answer '    R: ' .remainder
/* remove user defined symbols
rem sym .x4000
rem sym .d30
rem sym .answer
rem sym .remainder
em
```

Macro Example 3:

This is an example of testing data with some kind of masking operation. This could be used with various types of applications. One example would be part of an automated test sequence.

```
Define :m2
/* read data at location 4000h (Xdata)
def .x4000 = xby 4000
/* mask data with a 3Fh
def .mvalue = .x4000 XOR 3f
/* write the values to the screen
write 'Original value = ',xby 4000,'Masked with 3F = ', .mvalue
/* now and with 7Fh
def .avalue = .mvalue AND 7f
/* write the values to the screen
write 'Mask with 3f = ',.mvalue ,'   ANDed with 87 = ', .avalue
if .mvalue <EQ .avalue
write 'Error: Masked value less than or equal to ANDed value'
else
write 'MVALUE > AVALUE'
end
rem sym .x4000
rem sym .mvalue
rem sym .avalue
em
```

Macro Example 4:

Here is an example of a test of a bit at a memory location. This example could be used in some type of automated test sequence or error checking process.

```
Define :m3
/* test bit 3 of data memory
def .bvalue = dby 30 and 8
if .bvalue EQ 0
/* set the bit
dby 30 = dby 30 ior 8
```

```
else
/* clear the bit
dby 30 = dby 30 and F7
end
rem sym .bvalue
em
```

When using the macro language in the EMUL51-PC you can use a wide variety of resources from the target or processor. You could also choose not to use these resources and declare variables to be used for various things; as an example a counter. (See Macro Example 5)

Macro Example 5:

```
Define :m4
/* define a variable as a counter
def .counter = 0
/* reset the micro processor
reset chip
/* set up a loop condition based on this counter
repeat
/* go from current PC until a symbol in the program
go till .breakpoint
/* increment the users counter
.counter = .counter +1
/* repeat for 20 loops
while .counter <EQ 19
/* end the repeat
end
/* remove the user defined symbol
rem sym .counter
/* end the macro
em
```

These are a few of the things that you can accomplish with the macro language in EMUL51-PC. Other points of interest:

- Macros can be nested up to 7 levels deep.
- TSPut, TSGet, LOW, SAW, and INC commands are considered macros by the EMUL51-PC software.

Data Windows

The data windows are automatically updated between commands. This causes a short delay after each command. As the specified memory locations are all read, read (RD) signals will go out to the target system if the external data is mapped to the target system, (which it is by default). If this causes a problem, you can disable the window by using the command DIS XDATA.

Sub-windows There are four data windows. The window at the top has 18 locations that can be programmed by the user to reflect any memory location in any of the address spaces of the 8051. By default the 18 locations are set up to reflect the most common special function registers (RBYTE) of the 8051. For details on how to change each individual location, see SWR in the Commands chapter. Here is an example of how to change location 5 to reflect address 1E85H in external memory. The symbol name that we have chosen is A_VALUE:

```
SWR 5 XBYTE A_VALUE 1E85
```

The **DATA** window reflects 24 contiguous memory locations in the internal data memory. The first location is 20H by default, but can be set up to any position between 0 and E7H. For details see SWD in the Commands chapter. This example changes the first address to 60H:

```
SWD 60
```

The **XDATA** window reflects 24 contiguous memory locations in the internal data memory. The first location is 4000H by default, but can be set up to any position between 0 and FFE7H. For details see SWX in the Commands chapter. This example changes the first external address to 0H:

```
SWX 0
```

Both the DATA and XDATA windows may be moved around on screen. You may also change their size by using the function keys listed on the bottom of the screen.

The **STACK** window always reflects the stack. It shows a maximum of nine positions of the stack, starting with the byte to which the stack pointer is currently pointing. The contents of the stack are then shown in descending order.

Enable, Disable The static window is by default enabled. It can, however, be disabled using the DISABLE command or the F6 key. See the DIS paragraph for details.

```
DIS WINDOW
```

It may then be reenabled by using the ENABLE command or press F6 again. See the ENA paragraph for details.

```
ENA WINDOW
```

You can move the windows and expand them using the F4 and F5 function keys.

On-screen Operations Inside the DATA or XDATA window you can use PgUp and PgDn to scroll information.

File Operations It is also possible to save set ups to a file using the SAW command. See the SAW paragraph for details.

```
SAW A_SETUP
```

Set ups can the be loaded back using the LOW command. See the LOW paragraph for details.

```
LOW A_SETUP
```

By using many set up files you can switch the static window to reflect different parts of the memory. By using macros to do that you could switch windows using only three key strokes!

More Windows Ctrl + A toggles CODE Window ON/OFF
Ctrl + R toggles REGS Window ON/OFF
Ctrl + W toggles WATCH Window ON/OFF
Ctrl + D toggles DATA Windows ON/OFF
(F6 has the same function)

2 INSTALLATION

Dear 8051 User: 2-3
Materials and Supplies 2-3
Power Requirements 2-3
Inspection of Jumpers 2-4
Emulator Board 2-6
Standard Trace Board 2-10
Advanced Trace Board 2-13
Addressing the Emulator and Trace Boards 2-14
Installing the Emulator and Trace Boards 2-16
Mapping 2-17
System Requirements 2-18
Getting Started 2-18
Files Provided 2-26
EMUL51-PC/BOX 2-27
BANK SWITCHING 2-33

2 INSTALLATION

Dear 8051 User:

Thank you for choosing the EMUL51-PC. We hope it will be valuable to you in your 8051 projects.

Materials and Supplies

Check that the following items are present:

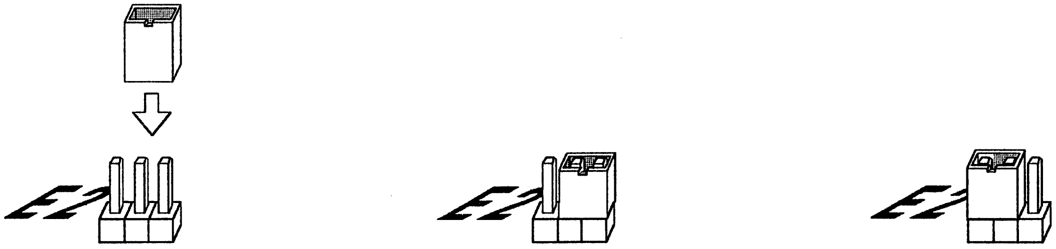
a. One pod board
b. One emulator board
c. One ribbon cable
d. Floppy disks
e. One trace board (optional)
f. Other optional items

Power Requirements

The emulator board requires about 1.7A from the PC's 5V power supply, and the trace board typically requires 1.3A. Before proceeding, check that the PC's 5V power supply is sufficient to deliver the necessary current. If it can't, a larger power supply will have to be installed. Your computer dealer can give you information on where to purchase one.

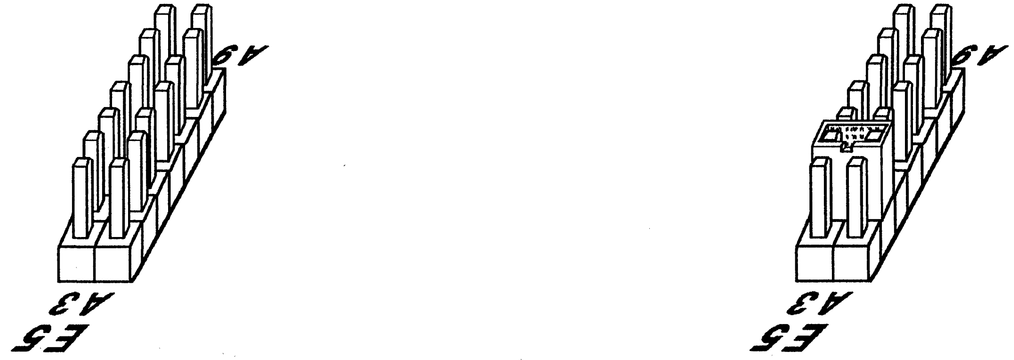
Inspection of Jumpers

Several sets of jumper pins are provided on the boards for configuring of on-board functions. The connection method is to slide a jumper block over two adjacent pins, (see Figures 1 and 2 for details).



a. Pins not connected b. Block connecting pins at right c. Block connecting pins at left

Figure 1. Typical Set of Three Jumper Pins



a. No pins connected b. Two pins connected, ("A4"pair)

Figure 2. Typical Set of Paired Jumper Pins

Jumpers for the emulator and trace boards are described in this section of the manual. For information on other POD boards, refer to the PODs chapter.

Board jumpers have been preset at the factory prior to shipment. On the boards you receive it is important to compare the jumper configurations against the configurations described in this manual. If a jumper is installed other than shown, refer to the "Jumper Descriptions" information before changing its position. Jumpers may be installed for operating characteristics required at the time of order.

Emulator Board

On the emulator board the locations of jumper pins are designated on the board artwork as E1, E2, E3, E4 and E5, (see specified rectangles in Figure 3).

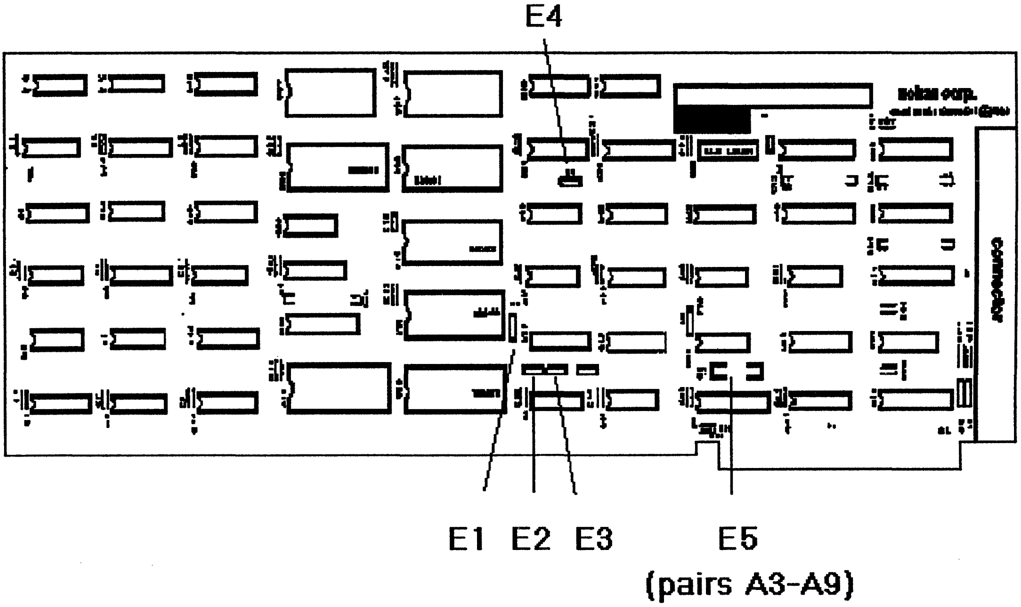


Figure 3. Locations of Jumpers on Emulator Board

32K Emulator When shipped from the factory, the emulator board is normally configured for 32K RAM and with I/O Port Address set for 110H. Figures 4A and 4B show details of the jumper connections for this configuration.

The 32K emulator board can be manufactured in either of two ways, depending on memory availability at the time of manufacture. The variations are: to have either one large memory installed in the bottom socket, or else to have four small memories installed in the four sockets.

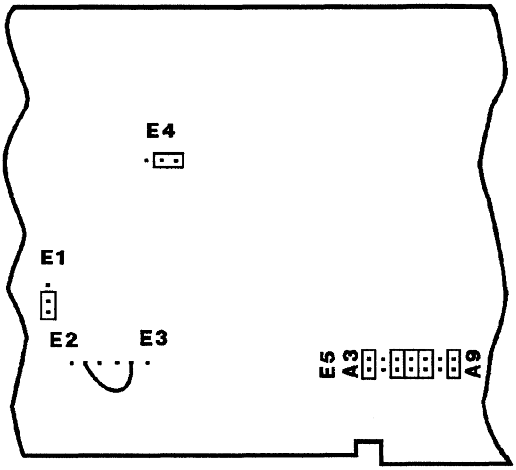


Figure 4A. Emulator Board with ONE CHIP 32KB installed, Address 110H. Code and XDATA (if mapped to emulator) always overlaid.

ONE-MEMORY E32 BOARD Figure 4A shows the jumper configuration if there is a single memory in the bottom socket. Note the wire wrap connection between E2 and E3. The memory is a 32k by 8 static RAM, of the type 62256, 43256, 55256, or similar number.

FOUR-MEMORY E32 BOARD Figure 4B shows the jumper configuration if there are four small memories in the sockets. The memories are 8k by 8 of the type 6264 or similar. (If there are four large memories in the sockets, the board is not a 32k emulator board.)

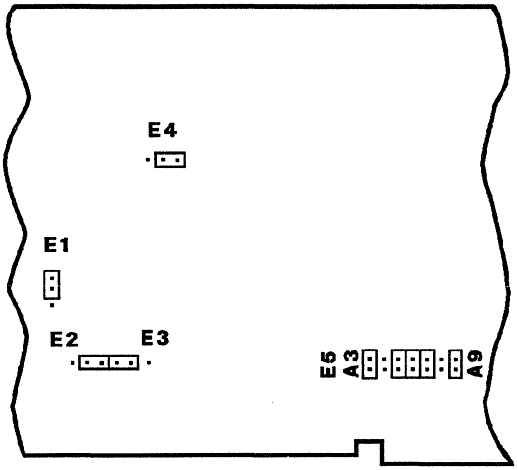


Figure 4B. Emulator Board with FOUR CHIP 32KB RAM, Address 110H. Code and XDATA (if mapped to emulator) always overlaid.

128k Configurations If four 32k by 8 RAM chips are inserted in sockets U25, U26, U27 and U28, then the emulator board can be configured either as shown in Figure 5 or as in Figure 6. The differences between these configurations depend on whether the 64k code and 64k XDATA are in separate areas or are overlaid. (Note: For Bank Switch configurations, see "Bank Switching" Section at the end of this chapter.)

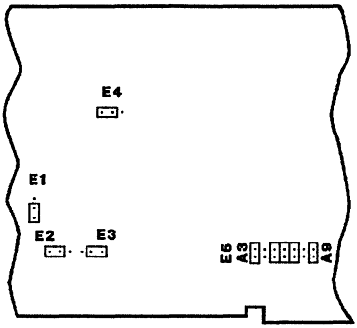


Figure 5. Emulator Board with 128KB installed, 64k Code and 64k XDATA in SEPARATE area Address 110H.

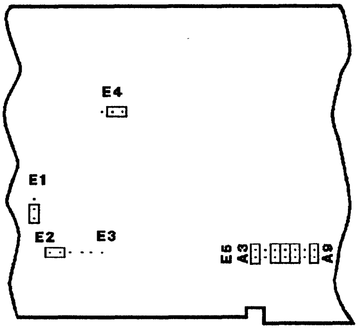


Figure 6. Emulator Board with 128KB installed, 64k Code and 64k XDATA in OVERLAID areas, Address 110H, MUST BE INVOKED AS A 32k EMULATOR!

Standard Trace Board

On the trace board the locations of jumper pins are designated on the board artwork as W1, W2, W3, W4 and W5, (see specified rectangles in Figure 7).

STANDARD TRACE BOARD MEMORY CONFIGURATION Both the TR4 and TR16 trace boards are shipped in one of two memory configurations. There is either one large memory soldered into the bottom row at position U42, or else there are four small memories soldered into the bottom row.

The one memory version has the lower jumper, W2, connected. The four-memory version has the upper jumper, W1, connected. Both variations are functionally the same. Since the W1/W2 jumper depends on the memories soldered at the time of manufacture, it does not get changed even if the trace frame size is changed.

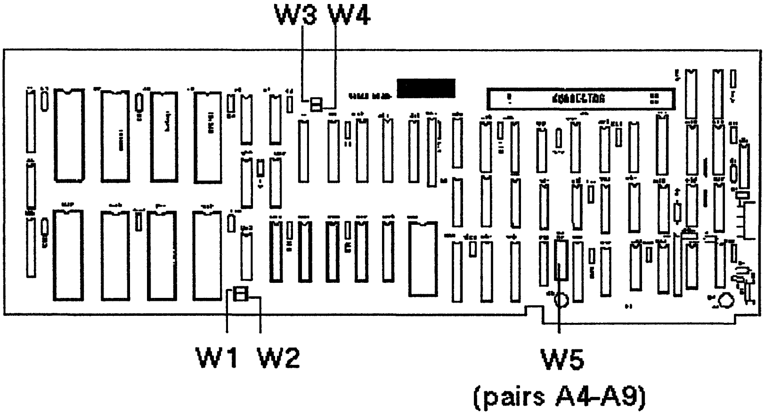


Figure 7. Locations of Jumpers on Trace Board

W1/W2 setting	U39-U41	U42
W1 W2	8k x 8 empty	8k x 8 32k x 8

These two factory build configurations both have the same function and size.

4K Trace Here is the trace board configured for 4k, (with 8k by 8 RAM chips in sockets U3, U4, and U5), and the I/O Port Address is set for 100H, (see Figure 8).

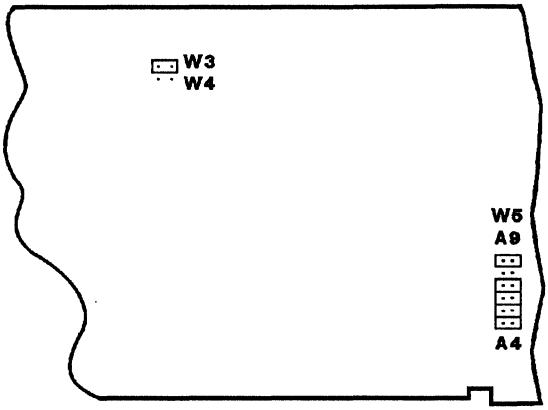


Figure 8. Trace Board with 4K Installed, Address 100H

16K Trace For a 16K trace board, 32K by 8 RAM chips have to be inserted in sockets U3, U4 and U5. Figure 9 shows the jumper configuration, again with I/O Port Address set for 100H.

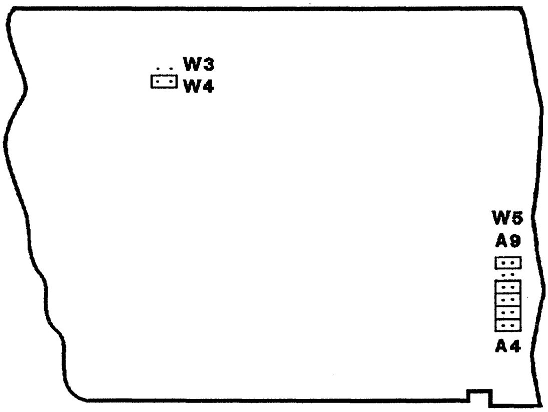
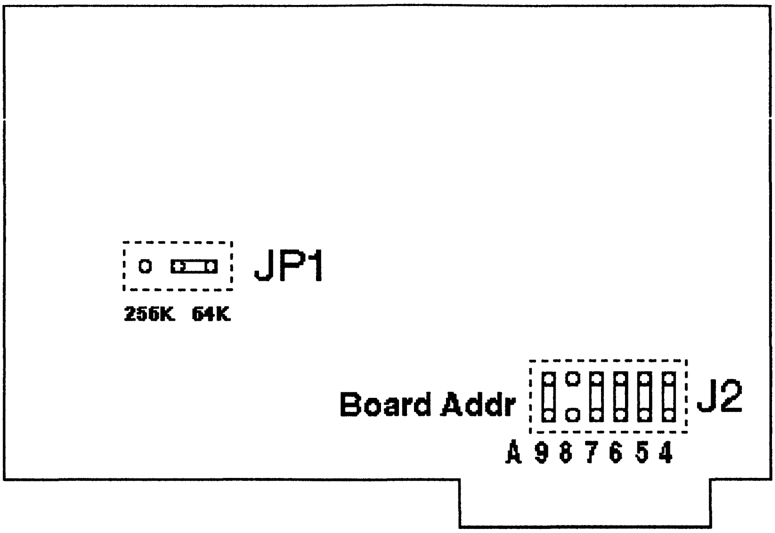


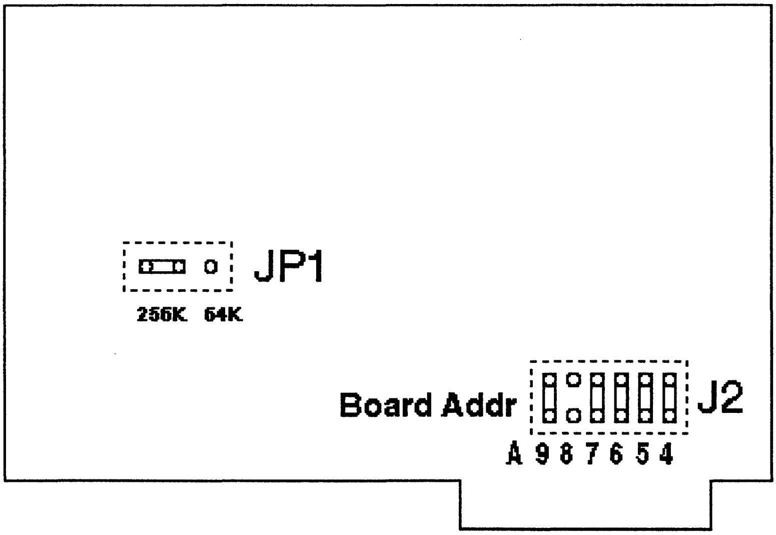
Figure 9. Trace Board with 16K installed, Address 100H

Advanced Trace Board The ATR64 trace board has a memory configuration with the jumper top on the right.



64K Advanced Trace Board Jumpers and Headers

The ATR256 trace board is has a memory configuration with the jumper top on the left.



256K Advanced Trace Board Jumpers and Headers

Addressing the Emulator and Trace Boards

The emulator address jumpers have been factory preset to 110 (HEX) and the trace board jumpers have been preset to 100 (HEX). These address settings are set for a typical system. Table 1 shows how a typical system uses its address locations. If your system is presently using locations 110 and 100 (HEX), alternate address locations will need to be found and the appropriate changes made to the jumpers and software.

Table 1. Standard I/O Addresses

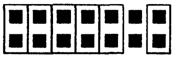
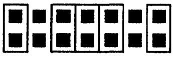
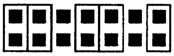
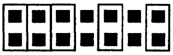
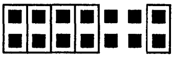
Hex Location	*Use*
000 - 0FF	Used by system
1F0 - 1F8	Fixed Disk
200 - 207	Game Adaptor
210 - 213	Expansion Unit
278 - 27F	Parallel Printer Port 2
2F8 - 2FF	Secondary Asynchronous Printer Adaptor
300 - 31F	Prototype Card
320 - 323	Fixed Disk Controller
360 - 36F	Reserved
378 - 37A	Printer Adaptor
380 - 38F	Alternate Binary Synchronous Communications Adapter, SDLC Adaptor
3A0 - 3AF	Primary Binary Synchronous Communications Adaptor
3B0 - 3BF	Monochrome Display and Printer Adaptor
3C0 - 3CF	Reserved
3D0 - 3DF	Color/Graphics Monitor Adaptor
3F0 - 3F7	Floppy Disk Controller
3F8 - 3FF	Primary Asynchronous Printer Adaptor

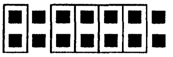
To change address jumpers, free address space must be found between 000 and 3FF (HEX) for the emulator and trace I/O addresses. The emulator board requires 8 consecutive addresses and the trace board requires 16 consecutive addresses. There are 6 batch files that have been included on the disk for your convenience. If there is a change to the addresses and/or memory, the batch files will have to be changed accordingly. (Batch files are discussed later in this chapter under Getting Started.)

Addressing Examples

The tables below give examples of addressing on the emulator and trace boards. Table entries are arranged in the same order that the jumpers are physically located on the boards.

Table 2. Emulator Addressing Example

3 4 5 6 7 8 9	Hex Adr.
	100 - 107
	110 - 117
	120 - 127
	140 - 147
	180 - 187

	210 - 217

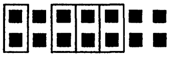
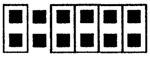
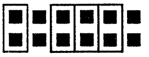
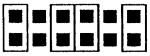
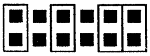
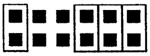
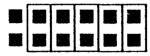
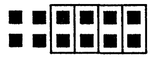
	310 - 317

Table 3. Trace Addressing Example

9 8 7 6 5 4	Hex Adr.
	100 - 10F
	110 - 11F
	120 - 12F
	140 - 14F
	180 - 18F
	200 - 20F
	300 - 30F

Installing the Emulator and Trace Boards

With the address jumpers in place and after you have inspected the boards for any damage, it is time to install the board(s) in your PC. Perform the following steps in the order shown.

1. Turn power off.

   POWER MUST ALWAYS BE OFF WHEN YOU PLUG IN THE BOARDS, AND WHEN YOU CONNECT THE RIBBON CABLE AND POD BOARD.

2. If you have both the trace and emulator boards, you should first insert the trace board with the short ribbon cable connected.

3. Connect the ribbon cable to the emulator board before inserting the board.

4. Make sure the connector fingers of the board(s) are secured into the female connector of the PC's motherboard.

5. Don't forget to put the screws back on the brackets.

6. Now connect the long ribbon cable to connector on the emulator board located at the back of the PC. The connector has a notch on it, so you can't insert it the wrong way.

7. Attach the other end of the ribbon cable to the POD board. Close the locks on the connector over the cable. The connectors of the ribbon cable are identical so it does not matter which end is connected to the POD or the emulator board.

8. Before you plug the pod into the target system, always first connect the "pod cable" to the target system ground. This prevents ground currents from flowing through any of the signal pins of the microprocessor chip, possibly destroying it.

Mapping

Code memory and external data memory can be mapped in 4K blocks to either the emulator or the target (your board). By default the code memory is mapped to the emulator, and the external data memory is mapped to the target. For details on how to change the mapping, please refer to the MAP paragraph in the COMMANDS section of the manual.

If you have a 32K emulator and you map both code and external data to the emulator, code and external data will be overlaid. In other words, for this work you must have your code and your data occupying different address areas.

If you have a 128K emulator and you map both code and external data to the emulator, code and external data will reside in physically different areas, so they are NOT overlaid. If, however, you want code and external data to be overlaid, (all 64K), you must change the jumpers as shown in Figure 6. AND YOU MUST INVOKE THE EMULATOR AS A 32K EMULATOR for this to work!!

System Requirements

Make sure that DOS 3.0 or a later version is being used. A minimum memory size of 640K is recommended.

Getting Started

1. Verify that the power supply jumper on the POD board is in the correct position.
2. Turn on the computer.
3. If you have configured the power jumper for target, power to target must be turned on.
4. If your installation disk has the file INSTALL.EXE, use it to install the EMUL51 files onto your hard disk. It can optionally create a new subdirectory to install the files to.
5. Invoke the program by typing EMUL51 followed by a number of parameters. For details on parameters see Step 8 below.
6. You can use an editor to create a startup batch file. If the program CONFIG.EXE is installed you can use it to generate E.BAT.
7. Create the batch file that corresponds to your hardware. If the emulator address had changed to, say, 240H, you would revise the contents of the batch file to look like this: EMUL51 -p8031 -e240 -m32

   This example applies when only the emulator board is used. Again, if any changes are made to the trace board address and/or memory, you will have to revise the contents of the batch file accordingly. Note that the appropriate *.SYM file should be present in the current directory or as specified by the environment variable EMUL31, (see note below).

Table 5. Typical Batch Files for Invoking EMUL51

File Name	*Contents*
E32.BAT	EMUL51 -p8031 -e110 -m32 For emulator only, 32K.
E128.BAT	EMUL51 -p8031 -e110 -m128 For emulator only. 128K (64K emulation memory plus 64K external data memory)
E32T4.BAT	EMUL51 -p8031 -e110 -m32 -t100 -f4 32K emulator and 4K trace buffer
E32T16.BAT	EMUL51 -p8031 -e110 -m32 -t100 -f16 32K emulator and 16K trace buffer
E128T4.BAT	EMUL51 -p8031 -e110 -m128 -t100 -f4 128K emulator and 4K trace buffer
E128T16.BAT	EMUL51 -p8031 -e110 -m128 -t100 -f16 128K emulator and 16K trace buffer

8. The parameters may appear in any order. The following options are available:

Table 6. Parameter Options When Invoking EMUL51

-p	<processor> Depends on POD and invocation. See -p Table 6-A. No "C" is used for CMOS, and no speed is specified. Example: -p80451 for POD-C451-PGA-16. If the emulator is DMA modified for 'C152 or 'C452, use a D suffix on all types, regardless of whether that part can do DMA. (Example: -p8032D for a DMA modified emulator board.)
-e	<emulator address>
-m	<number of K installed on the emulator (32 or 128)>
-t	<trace address>
-f	<number of K frames installed on the trace board (4 or 16)>
-s	<ii> or <io> or <oi> or <oo>. Used to specify usage of P3.6 and P3.7 as input (i) or output (o) for POD-51. Also used for POD-31-S or any "-S" type pod when configured to have P3.6 & P3.7 as I/O.
-d	<number>. Used to increase time in time-out loops; useful if you run at very low frequencies.
-c	<use if you drive a mono monitor with a color card>
-v	<use if you have problems with certain EGA/VGA cards>
-a	<serial channel (if you use the "BOX") (1, 2, 3 or 4)>
-i	<name of include file to be executed at invocation>
-b	<baud rate for serial channel. (30, 60, 12, 24, 96, 19, 38, 115)>
-x	<crystal frequency in MHz> Used by the advanced trace board for timestamp scaling.
-q	Maximum number of modules. Default is 100. 29 bytes are allocated for each module, independently of symbol storage. Increase to accommodate more modules. Reduce to decrease memory usage.
-n	<telephone number for modem including modifying characters>
	Modifying characters for telephone number:
P	Instructs the modem to use pulse-dialing.
T	Instructs the modem to use tone-dialing. (Default).
W	Directs the modem to wait until a dial tone is detected. If a dial tone is not detected within 30 seconds, the modem hangs up and displays the message NO DIALTONE.
0-9	Give the digits of a telephone number to be dialed. When using tone-dialing, the modem can also transmit the symbols * and # from a 12-button dialing keypad.
,	(comma) Directs the modem to pause for 2 seconds before processing the next symbol in the command.
!	Causes a hook flash of 1/2 second.
@	Instructs the modem to wait 30 seconds for one or more rings followed by 5 seconds of silence before processing the next symbol. This command can be used to call a system that does not have a dial tone.
/	Causes a pause of 1/8 second in the dialing sequence.

9. To invoke the EMUL51 it is easiest to use a batch file as described above. If you don't want to use a batch file, you can, of course, type *EMUL51* followed by the appropriate set of parameters as described above in Step 7.

10. To start the program, insert the disk in the default drive and execute the batch file that you have prepared.

11. If there is a problem with the addresses, you will receive error messages like "Can't find emulator at address xxx" or "Can't find trace at address xxx". Please check that the POD is connected to the emulator via the ribbon cable. If you still have problems, you may have to find another address space. (Also refer to the Troubleshooting chapter.)

    DON'T FORGET TO TURN POWER OFF BEFORE YOU PLUG IN, OR UNPLUG BOARDS, RIBBON CABLES, OR POD BOARD!

TABLE 6-A: "-p" Parameters for Invocation
Parameters are the same for any frequency suffix.

EXTERNAL MODE PODS	
-p8031	POD-31, POD-C31, POD-31-S, POD-C31-S
-p8031d	DMA emulator & POD-31, POD-C31, POD-31-S, POD-C31-S
-p8032	POD-32, POD-C32, POD-C32-S
-p8032d	DMA emulator & POD-32, POD-C32, POD-C32-S
-p8344	POD-44, POD-44-S
-p8344d	DMA emulator & POD-44, POD-44-S
-p80154	POD-C154, POD-C154-S
-p80154d	DMA emulator & POD-C154, POD-C154-S
-p80321	POD-321, POD-321-S
-p80321d	DMA emulator & POD-321, POD-321-S
-p8051fa	POD-C252/C51FA, POD-C252/C51FA-S
-p8051fad	DMA emulator & POD-C252/C51FA, POD-C252/C51FA-S

	EXTERNAL MODE PODS (Continued)
-p8051gb	POD-C51GB-PGA
-p8051gbd	DMA emulator & POD-C51GB-PGA
-p8051sl	POD-C51SL
-p80152	POD-C152-DIP, POD-C152-PGA
-p80152d	DMA emulator & POD-C152-DIP, POD-C152-PGA
-p407	POD-407
-p80451	POD-C451-DIP, POD-C451-PGA
-p80451d	DMA emulator & POD-C451-DIP, POD-C451-PGA
-p80452	POD-C452-PGA
-p80452d	DMA emulator & POD-C452-PGA
-p80515A	POD-C515A-PGA
-p80517A	POD-C517A-PGA (Previously called POD-C537A-PGA)
-p80535	POD-532-PGA, POD-535-PGA, POD-C535-PGA
-p80535d	DMA emulator & POD-532-PGA, POD-535-PGA, POD-C535-PGA
-p80537	POD-C537-PGA
-p80537d	DMA emulator & POD-C537-PGA
-p80552	POD-C552-PGA
-p80552d	DMA emulator & POD-C552-PGA
-p80652	POD-C652
-p80652d	DMA emulator & POD-C652

BONDOUT PODS	
-p5001	POD-5001
-p8051b	POD-C51B with C51 or C851 configuration
-p8051bd	DMA emulator & POD-C51B with C51 or C851 configuration
-p80652b	POD-C51B with 652, 654 or 662 configuration
-p80652bd	DMA emulator & POD-C51B with 652, 654 or 662 configuration
-p83410	POD-CL410
-p83410d	DMA emulator & POD-CL410
-p80451b	POD-C451B-PGA
-p80451bd	DMA emulator & POD-C451B-PGA
-p80515	POD-C517B-PGA with 515 configuration
-p80515d	DMA emulator & POD-C517B-PGA with 515 configuration
-p80515ab	POD-C517AB-PGA with 8XC515A configuration
-p80517	POD-C517B-PGA with 517 configuration
-p80517d	DMA emulator & POD-C517B-PGA with 517 configuration
-p80517ab	POD-C517AB-PGA with 8XC517A configuration
-p80552b	POD-C552B-PGA
-p80552bd	DMA emulator & POD-C552B-PGA
-p83410	POD-CL580
-p80751	POD-C751
-p80751d	DMA emulator & POD-C751
-p83782	POD-CL782

HOOKS MODE PODS	
-p8751FC	POD-C51FX
-p8752	POD-C52
-p8752d	DMA emulator & POD-C52
-p87054	POD-C054
-p80528	POD-C528
-p80528d	DMA emulator & POD-C528
-p80550	POD-C550-PGA
-p80550d	DMA emulator & POD-C550-PGA
-p83558	POD-C558
-p80575	POD-C575
-p80592	POD-C592-PGA
-p80592d	DMA emulator & POD-C592-PGA
-p80752	POD-C752
-p80752d	DMA emulator & POD-C752

PORT SUBSTITUTION (Internal-Mode-Only) PODS	
-p8051	POD-51, POD-C51
-p8051d	DMA emulator & POD-51, POD-C51
-p8052	POD-51, POD-C51 to use 256 bytes of RAM (but not Timer 2)
-p8052d	DMA emulator & POD-51, POD-C51 to use 256 bytes of RAM (but not Timer 2)

NOTE: Some of the DMA "d" options may not be implemented yet. Contact Nohau if you have a problem.

Files Provided

The following files are provided with EMUL51-PC:

Table 7. Some Files Provided With EMUL51-PC (A Partial List)

Command/File	*Function*
EMUL51.EXE	Emulator program.
EMUL31.HLP	Documentation file. It is used from within EMUL51 via an indexing system. This file should be in the current directory or in a directory specified by SET EMUL31.
EMUL31.NDX	Index file for EMUL31.HLP.
8031.STR	Typical 8051 object files are automatically loaded into the EMUL51 at invocation. These files should be in the current directory, or in a directory specified by SET EMUL31. (See below.)
Examples:	
8032.STR	Used for 8032 type processors.
8051.STR	Used for 8051.
80452.STR	Used for 80C452 procesors.
	(and many more)
TEST.SRC	A short 8051 assembly program (source list file).
TEST.A03	Object file created from TEST.SRC. This file can be loaded into the EMUL51 via the LOAD command. The disassembler can be used to view it.
TEST	Object file created from TEST.C and TEST1.C. This file can be loaded into EMUL51. It can demonstrate source-level debugging.
8031.SYM	Symbol file for 8031. Symbol files for other processors are also available.

Note that it is not necessary to have *.STR, EMUL31.HLP, EMUL31.NDX and the *.SYM in the current directory. We have used the "environment" feature in MS-DOS to implement this. Here is how it works.

The program will start looking for 8031.STR, EMUL31.HLP, etc., in the current directory. If they are not found, it will look for the environment parameter EMUL31. If it is found, the string defined by EMUL31 will precede the file names to form a path to the directory in which the files are. We recommend adding the SET command to your autoexec.bat file so that it will automatically be included when your system is booted.

SET EMUL31=\usr\nohau\ is an example of using the SET command for an environment string. The files 8031.STR, EMUL31.HLP, etc., should then be available in the subdirectory \usr\nohau.

EMUL51-PC/BOX

The Box option accommodates users having PCs without spare card slots or who may need to move the emulator frequently, as well as PS/2 users. The Box has its own power supply and accepts the same emulator and trace boards as for the PC.

The Box-S version communicates with the PC over a standard serial port. The Box-M version has a serial channel and an internal modem; this allows the PC to be at a location remote from the Box and system under test.

The Box-P version (not available in all areas) communicates with the PC over a dedicated parallel port which extends the I/O bus of the PC, and requires a "non-standard" parallel card (provided by Nohau) to be installed in the PC.

Preliminary Steps If the back of your box chassis is labeled 115/260 volt or similarly, it has an automatically switched power supply and no selection is required. If your chassis is not so labeled, remove the top cover from the Box and confirm that the power supply is correctly configured for your line voltage (115 - 130 VAC or 220 - 240 VAC). Then plug the power cord into the Box receptacle at the rear of the box. After that, check the board (or boards) supplied with the Box, as follows.

Box-S: One serial card that plugs into any slot in the Box. (Also supplied is a serial communication cable.)

Box-P: One parallel transmitter card that plugs into any slot in the Box. (Also supplied are a parallel communication cable and a separate parallel receiver card for plugging into your PC.)

Box-M: One serial card that plugs into any slot in the Box, and one modem card (Everex EV-940A) that plugs into any slot in the Box. (Also supplied is a modular telephone cable for connecting the Box to the telephone line.)

Emulator and Trace Boards If your emulator board, and trace board if you have one, are not already installed in the Box, install them in the same manner as you would install these boards into your PC. Don't forget to secure the boards with the bracket screws.

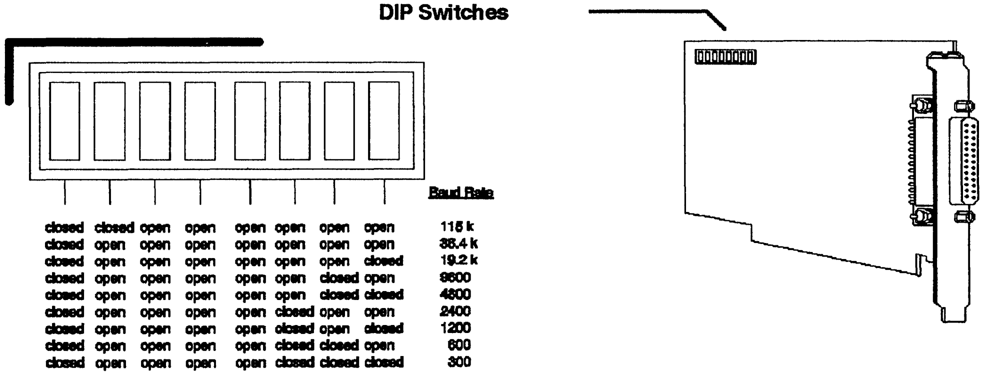
Box-P (Not available in all areas) Because Box-P is like an extension of the PC bus, all considerations of I/O addresses for the emulator board and the trace board as described earlier in this manual still apply. Electrically this option is equivalent to having the boards inside the PC. The emulator board comes with the I/O address preset to 110H, and the trace board's I/O address is preset to 100H.

1. Replace the top cover on the Box.
2. Plug one end of the twisted-pair ribbon cable into the emulator board; make sure that the tang on the cable lines up with the notch in the emulator board. Plug the other end of the cable into the POD board in the same way. (The cable is symmetrical, so it doesn't matter which end is plugged into the emulator.)
3. Plug the parallel receiver board into an empty slot in the PC.
4. Mate one end of the communication cable to the connector on the parallel card in the Box, and mate the other end to the parallel card in your PC.
5. Plug the Box power cord into the AC power outlet and turn on the Box.
6. Turn on your PC.
7. Install the EMUL51-PC software on your PC and create a batch file, as described in Table 5 and Table 6 earlier in this chapter under the heading "Getting Started."

Box-S The serial card inside the box has an 8-position DIP switch in the upper rear corner. Figure 12 shows the switch location and gives a list of the switch settings. For Baud rate a typical setting is 9600.

1. Confirm that the DIP switch settings are properly selected for your application, then replace the top cover on the Box.

2. Plug one end of the twisted-pair ribbon cable into the emulator board; make sure that the tang on the cable lines up with the notch in the emulator board. Plug the other end of the cable into the POD board in the same way. (The cable is symmetrical, so it doesn't matter which end is plugged into the emulator.)



								Baud Rate
closed	closed	open	open	open	open	open	open	115 k
closed	open	open	open	open	open	open	open	38.4 k
closed	open	open	open	open	open	open	closed	19.2 k
closed	open	open	open	open	open	closed	open	9600
closed	open	open	open	open	open	closed	closed	4800
closed	open	open	open	open	closed	open	open	2400
closed	open	open	open	open	closed	open	closed	1200
closed	open	open	open	open	closed	closed	open	600
closed	open	open	open	open	closed	closed	closed	300

To use the 115 k baud option, you must have:

a. EPROM for the Box Board (Rev B2 or "B1 + 115KB" or "2.1 + 115KB" or later)
b. Version 5.6F EMUL51-PC software or later

Figure 12. DIP Switches and Settings on Serial Transmitter Board

3. Mate one end of the serial communication cable to the connector on the Box serial board, then mate the other end of the cable to the serial port connector on your PC.

4. Turn on your PC.

5. Plug the Box power cord into the AC power outlet and turn on the Box.

6. Install the EMUL51-PC software on your PC and create a batch file, as described in Table 5 and Table 6 earlier in this chapter under the

heading "Getting Started." Here is an example of a batch file:

```
EMUL51 -p8031 -e110 -m128 -t100 -f16 -a1
```

The "-a1" indicates that you are using serial port 1. The following COM ports are supported: COM1 on address 3F8. COM2 on address 2F8. COM3 on address 3E8. COM4 on address 2E8. They are specified as -a1, -a2, -a3 or -a4. Specifying -a1 or -a2 causes interrupts to be used for received data.

The address of the COM board can also be specified after the -a option. Example: -a180 for serial port on address 180. If you do not want interrupts for COM1, specify -a3F8.

The I/O address range is from 100 to FFFF.

PS/2 computers may only work using interrupts and may be limited to using either COM1 or COM2.

7. Now execute the batch file. If the program comes up, you are now ready to start working. If not, you may get one of the error messages described below.

The message *send error (01)* three times it is an indication that the serial communication is not working. Check the serial card in your PC and verify that you have indicated the right port number in the batch file. Also check the physical connections. Then make sure that the switches on the serial card in the box are in the proper positions: closed, open, open, open, open, open, closed, open. (See Figure 12.)

If you get error messages like *unsuccessful (10)* three times, it is an indication that the serial communication is working, but the emulator is unable to communicate with the serial board in the box. Probably the processor on the pod board is not running. Check the power jumper and crystal jumper. Also confirm that the pod board is hooked up to the emulator board!

BOX-S Communication Error Codes

"Send Error (XX)": Error messages from COM port.

01	Timeout.
02	UART Overrun Error.
04	UART Parity Error.
08	UART Framing Error.
80	Lost Carrier (No DCD from Modem).

Combinations of bits are possible. For example:

"Send Error 03":	Both timeout and overrun errors occurred.
"Data Error":	Unexpected data was received.
"CRC Checksum Error":	There was a bad CRC in the received message.
"Unsuccessful (XX)":	Error messages back from the box.

01	Lost Carrier (Modem).
02	Wrong start character was received (not STX).
03	Checksum error.
04	Wrong message length.
05	Bad command code.
10	"em_adr" failed (timeout).
11	"em_sdat" failed (timeout).
12	"em_gdat" failed (timeout).
13	"em_go" failed (timeout).

Box-M The program both in the PC and in the box use standard Hayes commands, so any Hayes-compatible modems should work. However, we can only guarantee that it will work if you use the Everex EV-940A 2400 baud modem, which is the type of modem installed in the EMUL51-PC/BOX-M. Given below are some of the features listed in the EV-940 manual:

- Compatibility with the Hayes Smartmodem command set and with the major types of modems in current use in the United States, Europe, Asia, Australia, and South America.
- Transmission speeds of 300, 1200, and 2400 bits per second.
- Automatic adjustment of speed to match the speed of an incoming call.
- Automatic adjustment of speed when dialing out at 2400 bps, (i.e., "downshifting").
- Compatibility with Bell 212A, CCITT V.22bis, and CCITT V.22.
- A half-sized board that can be installed in any expansion slot in an IBM PC, XT, or AT, and in most compatible computers.

1. Confirm that the DIP switch settings on the serial board are properly selected for your application, (see Figure 12). For 2400 baud the switch settings should be:

   closed, open, open, open, closed, closed, open, open.

2. Replace the top cover on the Box.
3. Plug one end of the twisted-pair ribbon cable into the emulator board; make sure that the tang on the cable lines up with the notch in the emulator board. Plug the other end of the cable into the POD board in the same way. (The cable is symmetrical, so it doesn't matter which end is plugged into the emulator.)
4. Plug one end of the modular telephone cable into the jack on the modem board, and plug the other end into the telephone line.
5. Turn on your PC.
6. Plug the Box power cord into the AC power outlet and turn on the Box.
7. Install the EMUL51-PC software on your PC and create a batch file, as described in Table 5 and Table 6 earlier in this chapter under the heading "Getting Started." Here is an example of a batch file for 2400 baud and Nohau's phone number:

```
EMUL51 -p8031 -e110 -m128 -t100 -f16 -a1 -b24 -n4088661820
```

The "-a1" indicates that you are using serial port 1. (Ports 2, 3, and 4 are also supported.)

BANK SWITCHING

Many users today are breaking through the 64k barrier of the 8051 family chips. The problem is that every user is utilizing a different scheme to implement the bankswitching hardware. Our present solution is to modify a standard 128k emulator board to mimic the bankswitching schemes of individual target systems.

The operation of bankswitching memory in the EMUL51-PC requires certain information from the user. This information needs to be very specific about how the memory operation will be handled.

If the emulator is used without bankswitching there will be 64K of overlaid memory. (CODE and XDATA). In this case the "bankswitch cables" should be left grounded.

Currently Nohau has two different types of emulators that we feel will meet most users' needs. If none of the configurations will satisfy your requirements, please contact us.

This section covers the following topics :

- Software examples with IAR / Archimedes and Keil / Franklin compilers
- Hardware examples
- EMUL51-PC bankswitch commands and examples on setup
- Description of the Nohau EMUL51-PC/E128-BSW bankswitch emulator
- Description of the Nohau EMUL51-PC/E256-BSW bankswitch emulator
- Hints concerning bankswitching the Nohau way

Software Examples Using Bankswitching and the IAR / Archimedes Compiler

Description of the IAR / Archimedes Banked Memory Model:

The banked memory model is normally supported by the IAR / Archimedes compiler and does not require any additional software. However, you must use version 4.00 or later of the compiler / linker package. The compiler supports up to 256 banks of code. The IAR / Archimedes compiler supports common area and banks but has (unlike the Keil / Franklin) the restriction that the common area must be at least 16kbyte. Like the Keil / Franklin, the compiler works very well with the Nohau bankswitch emulators. Writing code for bankswitching with the IAR / Archimedes does not differ from normal use. However, you should keep the module size as small as possible, or at least check the module size versus the bank size. This will enable the linker to fit all banks together without losing too much code space.

Parameters for the IAR / Archimedes compiler / linker in banked memory model:

IAR / Archimedes ICC8051 compiler options:

-mb This parameter determines that banked memory model should be used. The library file used with the banked memory model is called CL8051B.LIB

The default port assignment for bankswitching is port P1. If any other assignment is required the assembler file L18.S03 needs to be modified. After the modification, assemble and put the module in the CL8051B.R03 library using the XLIB librarian replace-module command (see the XLIB section in the IAR / Archimedes manual for further reference).

IAR / Archimedes XLINK linker options:

-Z(segment type)segment=address The -Z option is the standard option for attaching addresses to segments. When using bankswitching this switch is used for the common areas and the -b switch, described below, is used for the bankswitched areas.

```
-b(segment type)segment=bank+start address, bank length,
bankincrement+offset
```

The bank determines what the first bank number should be. The start address determines the start address of the banked area. The bank length sets the size of each bank in the banked area.

The bank increment consists of two 16 bit values. The first two bytes determine the number of bank increments to be performed when the previous bank is filled. The last parameter offset is an offset from the local address to be able to create asymmetrical bank arrangements. This value is normally set to zero.

Example 1:

```
-b(CODE)CODE, MOD1, MOD2, DISPLAY = 00006000, 2000, 00010000
```

The first two bytes tell the linker that the first bank number is zero and the next two bytes determine that the bank area starts at 6000H. The next parameter sets the bank size to 2000H (8kbyte). The last parameter determines that each new bank number should be incremented by one and that no local offset should be used. Then the linker automatically fills the different 8kbyte banks with code. In this example no common (root) area is defined. However, this example is not suitable for the Nohau bankswitch emulators.

A more useful example may be as follows:

Example 2:

```
-Z(CODE)MOD_INT, KEY_POLL=0
-b(CODE)MAIN, KEYBOARD, DISPLAY, PWM_UNIT = 00008000, 8000,
00010000
```

This configuration defines the first bank number to be zero, every bank starts at 8000H, each bank is 32k of size, and the bank number increment is one with no local address offset. The standard -Z option defines the common (root) area. In this case the interrupt and a keyboard poll routine are put in the root area.

This configuration is supported by all of Nohau's different bankswitch emulators and is also quite easy to implement in hardware (see following hardware examples). It is also software wise, a good generic solution that allows up to 32k of common space for interrupt vectors, string constants, and other frequently used routines. When using the Nohau emulator there is the possibility of having from one to seven 32k banks in addition to the root bank. This will cover most needs of large code size applications.

In general the IAR banked memory model lets you use many -b declarations to add multiple modules to one bank definition.

Example 3:

```
-b(CODE)CODE0, MAIN, IN_OUT, PWM_UNIT, MAG_READ = 00004000,
4000, 00010000
-b(CODE)SERIAL1, I2C_COM, DISK_IO
```

This will define a 16k bank system, with the modules both on line 1 and 2 included in the bank definition. This is easily done by just leaving out the special bank parameters in line 2.

The example above can also be written like this:

Example 4:

```
-b(CODE)CODE0, MAIN, IN_OUT, PWM_UNIT, MAG_READ \
SERIAL1, I2C_COM, DISK_IO = 00004000, 4000, 00010000
```

For further information on bankswitching with the IAR / Archimedes compiler, please refer to the "The Banked Memory Model" section in the IAR / Archimedes manual.

Using Bankswitching With the Keil / Franklin BL51 Banked Linker

Description of the Keil / Franklin BL51 Bank Linker

The Keil / Franklin BL51 Bank Linker is an extension of the standard L51 Linker. The BL51 Linker is able to handle up to 16 64kbyte banks of code which will increase the 8051 code area to 1Mbyte instead of the normal 64kbyte. Writing code for a bankswitch application does not require any modifications of existing code. This applies to the IAR / Archimedes as well. The only actual difference to the IAR / Archimedes is the number of banks and the fact that no root bank is required. However, most applications require a root bank. The BL51 Linker supports all the different modes of the Nohau bankswitch emulators.

Keil / Franklin BL51 Linker Options

BANKAREA (start, end)

Determines the size of the banked code.

Example 1:

BANK AREA (0000H, FFFFH)

This is for a solution using all of the total 64kbyte as banked area.

Example 2:

BL51 BANKAREA (8000H, FFFFH)

This is for an application with a physical root bank from 0000H to 7FFFH and the banked area from 8000h to FFFFH.

COMMON (saddr | seg)

Locates segments in the common area. The common area is available to all banks and does not require any bankswitching when accessed.

Example 3:

BL51 COMMON{MOD_INT, KEY_POLL, MOD_UART}, BANK0{MOD_1, MOD2, MOD3 }, BANK1 {MOD3}

This puts the interrupt module MOD_INT, the keyboard poll routine KEY_POLL, and the serial interface module MOD_UART in the common area.

BANK0 (saddr | seg) / BANKn (saddr | seg)

Locates segments in the bank 0..n. Up to 16 (0..15) banks are supported.

Example 4:

```
BL51 COMMON {MOD_INT.OBJ, MOD_ROOT.OBJ} BANK0
{MOD_1.OBJ, MOD_2.OBJ} BANK1 {MOD_DISP.OBJ, MOD_4.OBJ}
BANKAREA ( 8000H TO 1FFFFH)
```

**** NOTE ****

When using an application with no physical common (root) area, the Keil BL51 does not duplicate the areas defined as common. By using the Object Converter OC51 you will get the code of the different banks in separate files. The Nohau EMUL51 supports this duplication automatically from version 5.7R. Please contact your Nohau distributor for updates if required.

For further information on parameters and the BL51 Linker please refer to the Keil / Franklin BL51 Linker / Locater manual and the L51 Linker / Locater manual.

Hardware Examples on Bankswitching With the 8051

The following pages will show some examples on how to handle 8051 hardware bankswitching. These are only examples, so please use them as suggestions on how to deal with bankswitching when designing your hardware.

When using the Nohau bankswitch emulators, all different types of hardware designs can be supported. However, some cases may require a specially configured emulator. If this manual and the different bankswitch schemes do not seem to meet your needs, please contact Nohau to see if we can work out a solution that will suit your requirements.

When designing bankswitch hardware which is to be used with emulators, try to simplify the accessibility of the bankswitch signals since these must always be connected to the pod board (see the emulator descriptions further on in this manual). This is especially important if you use xdata writes to switch banks. In these cases the bankswitching signals are only available in your target.

In general, one thing to consider when designing hardware to be used with emulators is to prepare for connecting a pod. Make sure the PCB layout does not put you in a corner. Try to have space around your CPU socket. If you have a surface mounted target, mount some of your prototype boards with sockets. Surface mounted PLCC sockets are available today from most socket and connector manufacturers.

EMUL51 BANKSWITCH COMMANDS

The following list of commands, except SCOPE, refer only to bankswitching :

BANKBYTE [address] [memtype] [mask]

This command defines which byte controls the bankswitching and holds the current bank number. Both register bytes and XDATA bytes are defined as bankbyte with this command. The emulator needs the bankbyte information to be able to read the current bank number when the BANK command is issued (see BANK command further down).

Example 1:

BANKBYTE 90H RB 1CH

The address 90H refers to port P1, RB refers to a register byte, and the value 1CH is the mask used for "filtering" the port P1 value so that the correct bits are used. In this example the bits P1.2, P1.3, and P1.4 are used. The mask bits must be consecutive and not inverted.

Example 2:

BANKBYTE 3FFFH XB 3H

In this case, XDATA address 3FFFH is used. The two first bits in this byte are used.

BANK [bank number]

This command displays or sets the current bank. Actually it contains the value of BANKBYTE since that is the reference to the bankswitch byte.

Example 3:

BANK 2

This sets the current bank to number 2. When using BANK for switching to a bank, it executes the BANKSWITCH macro defined before loading your code (see the "Examples on different ways to set up the EMUL51 for bankswitching" section further on).

BANKADDRESS [address1] TO [address2]

This command is for setting up the "banked area". This command must be invoked before loading the code. The emulator needs this information to handle the code correctly when loading.

The default values are 8000H TO FFFFH, which can be handled by all of the Nohau bankswitch emulators.

Example 4:

BANKADDRESS 0 TO FFFF

This will set up the emulator for handling a full 64k (0H to FFFFH) bank area.

BANKSEGMENT [bank number] [segment name]

This command is for "attaching" a certain segment name to a bank number. Many segments can be attached with one bank but they must be entered one by one. The number of segment symbols and segments in each bank must coincide. The address boundaries of the segments and the different banks must also coincide. This command must be used when the object code contains segment information which needs to be attached to their respective bank.

Example 5:

```
BANKSEGMENT 1 SEGMENT1
BANKSEGMENT 1 SEGMENT2
BANKSEGMENT 1 SEGMENT3
BANKSEGMENT 2 SEGMENT4
BANKSEGMENT 2 SEGMENT 5
```

This will attach segments 1 - 3 to bank 1 and the rest, 4 - 5 to bank 2

BANKMODULE [banknumber module]

This command is for attaching a module to a certain bank. This requires that the module name information is present in the object file.

Example 6:

BANKMODULE 1 INT_VEC
This will attach the module INT_VEC to bank 1.

SCOPE [module] This command displays the scope of the code which was loaded for each module. Scope is the address area a certain module resides in. If the code is in a bank as defined by BANKSEG it is indicated.

Example 7:

SCOPE This displays the code scope of all modules.

SCOPE CMD_INT This displays the code scope of module CMD_INT.

All of the above commands except SCOPE initiate the emulator to handle bankswitching. There is not any way to "reset" bank handling, except by exiting the EMUL51 and re-enter.

Examples of Different Ways to Set Up the EMUL51 For Bankswitching

Example 1:

```
BANKBYTE 38 DB             ;Internal data address 38H is used as " bankbyte"
BANKADDRESS 0  TO FFFF     ;All 64k is bankswitched. Bank size is 64k
BANK 1                     ;Symbols in FILE1 belong to bank 1
:SWITCH 1                  ;Invoke macro to switch hardware to bank 1
LOAD FILE1                 ;Load FILE1
:SWITCH 2                  ;Switch hardware to bank 2
LOAD FILE2                 ;Load FILE2
```

The object files in this example do not contain any segment information.

The macro SWITCH could for example be defined as follows :

```
DEFINE : SWITCH
P1 = %0
EM
```

The advantage of using macros is that you can make your own "commands" with parameters. This will save you typing time and when using bankswitching, macros are required (see macro BANKSWITCH in examples further on).

Example 2:

```
BANKBYTE E023 XB                  ;External data address E023 is used as
                                  bankbyte
BANKADDRESS 8000 TO FFFF          ;Upper 32k is bankswitched
BANKSEG 1 SEGMENT1                ;Specify banks in different segments
BANKSEG 1 SEGMENT2
BANKSEG 1 SEGMENT3
BANKSEG 1 SEGMENT4
BANKSEG 2 SEGM1
BANKSEG 2 SEGM2
BANKSEG 2 SEGM3
LOAD BASEBANK.OBJ                 ;Load the common (root) bank.
                                  Addressed up to 8000 (32k)
:SWITCH 1                         ;Invoke macro to switch to bank 1
LOAD BANK1.OBJ                    ;Load file BANK1.OBJ
:SWITCH 2                         ;Switch to bank 2
LOAD BANK2.OBJ                    ;Load file BANK2.OBJ
:SWITCH 3                         ;Switch to bank 3
LOAD BANK3.OBJ                    ;Load the code to the last bank
```

In this example the segments were defined before loading the code so that all symbols ended up in the right bank.

Another example of the macro SWITCH could be as follows :

Example 3:

```
DEF : SWITCH                      ;Define macro with name SWITCH
RBY .P1 = %0                      ;Sets the bits in P1 according to the input
                                  bank number
EM                                ;Ends macro definition
```

In this macro definition P1 is used as bankbyte. The %0 refers to the first parameter used with this macro. In this case, it is the bank number.

As mentioned above, bankswitching can be performed while the code is loaded. At present this works for INTEL OMF with SEGMENT INFORMATION as described above. It also work with IAR / Archimedes. When the loader detects that bankswitching needs to be done, it checks to see if a user defined macro called BANKSWITCH is present. If so, it calls the macro with one parameter, namely the bank number.

Example 4:

```
DEF :BANKSWITCH          ;Define macro BANKSWITCH
IF %0 EQ 1               ;If bank number equals 1
RBi .P1+0 = 0            ;Set P1.0 to "0"
ENDIF                    ;End of IF statement
IF %0 EQ 2               ;If bank number equals 2
RBI .P1+0 = 1            ;Set P1.0 to "1"
ENDIF                    ;End of IF statement
EM                       ;End macro definition
```

In this example P1 bit 0 is used to switch banks, and only 2 banks are used. The BANKSWITCH macro must be defined according to the hardware switching scheme in the target system.

Tracing During Bankswitching

When bankswitching is activated the trace will use E0 and E1 to trace the active bank. E0 and E1 are available on the POD board as "wire wrap posts". When nothing is connected a "1" will be traced. The following combinations on E0 and E1 indicates current bank:

E1	E0	BANK
0	0	bank 0
0	1	bank 1
1	0	bank 2
1	1	bank 3

The result will be that the correct symbols will be shown according to the bank indicated by the values traced at E0 and E1.

Breakpoint Handling During Bankswitching

On the Nohau bankswitch emulators there are no direct ways to specify in which bank a break should occur. This could result in breaking at the correct address but in the wrong bank. If you have a trace board, however, it is possible to make the break occur in the correct bank as the following examples will show. This example uses a the external signals E0 and E1. The current software only supports E0 and E1 for tracing. That is, only four banks are supported with the trace and this macro.

```
DEFINE :GO              ;Define macro GO
RES BR                  ;Reset current breakpoints
TBR = ON                ;Enable trace breakpoints
TR = TRACE ALL & TRIG A & ITRAC ALL
                        ;Set up trace conditions
IF %1 EQ 0              ;If bank number equals 0
QRA0 = ADR %0 & P3 00XXXXXY
                        ;Set up QRA0 for bank 0
GO                      ;Start emulation
ORIF %1 EQ 1            ;If bank number equals 1
QRA0 = ADR %0 & P3 01XXXXXY
                        ;Set up QRA0 for bank 1
GO                      ;Start emulation
ORIF %1 EQ 2            ;If bank number equals 2
QRA0 = ADR %0 & P3 10XXXXXY
                        ;Set up QRA0 for bank 2
GO                      ;Start emulation
ORIF %1 EQ 3            ;If bank number equals 3
QRA0 = ADR %0 & P3 11XXXXXY
                        ;Set up QRA0 for bank 3
GO                      ;Start emulation
ELSE
BR = OFF                ;Disable trace breakpoints
TR = TRIG NOTRIG        ;Set up trace for default values
QRA0 = ADR XXXX & P3 XX
                        ;Reset QRA0 to default values
DIS QRA0                ;Disable qualifier QRA0
GO TILL %0              ;Go till address parameter
ENDIF                   ;End IF statement
EM                      ;End macro definition
```

Usage:

Type ":go [Break Address], [Bank Number #]"

[Break Address]The breakpoint address

[Bank Number #]The bank number in which you want the break to occur in

0 equals bank number 0
1 equals bank number 1
2 equals bank number 2
3 equals bank number 3
x equals common (root) bank

This will work when a common (root) bank is used. If the Break Address is 0000 - 7FFF then the emulator will always break, regardless of the Bank Number. If the address is 8000 - FFFF the Bank Number # must be specified.

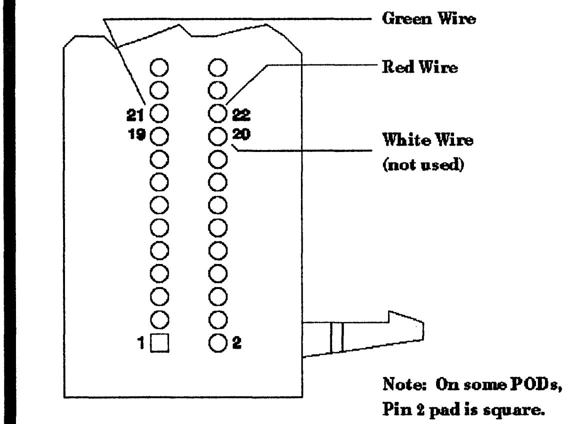
In this case two EZ-hooks must be connected between the bankswitch signals in your target system and E0 and E1on the POD board. Use E0 as the LSB and E1 as MSB.

The trace can be set up for any type of bankswitching supported by Nohau emulators, in addition to a bankswitch application with a configuration other than that where the above example is used. The possibility to set up with the trace breakpoints based either on specified port values or xdata writes with a specific value will let you break in the bank you desire.

For further information on bankswitching and commands for the EMUL51, please refer to the bankswitching and commands section.

Description of the Nohau EMUL51-PC/E128-BSW Bankswitch Emulator

The 128k bankswitch emulator has two different modes of operation which will be discussed later. This emulator cannot emulate XDATA memory. Any POD board can be modified to work as a bankswitching unit. For most operations, one or two wires will be used to control the bank selection. These wires must always be connected to the bankswitch logic. See Figure 10 for the modifications required for the POD board.

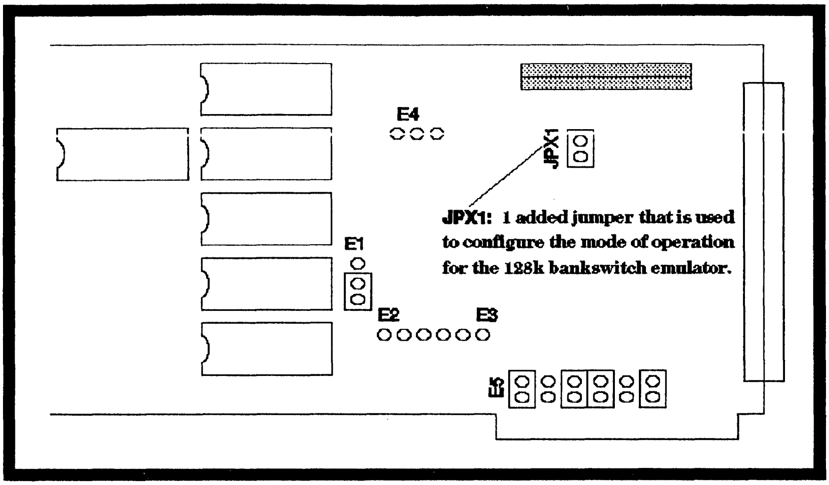


Note: On some PODs, Pin 2 pad is square.

POD BOARD, SOLDER SIDE

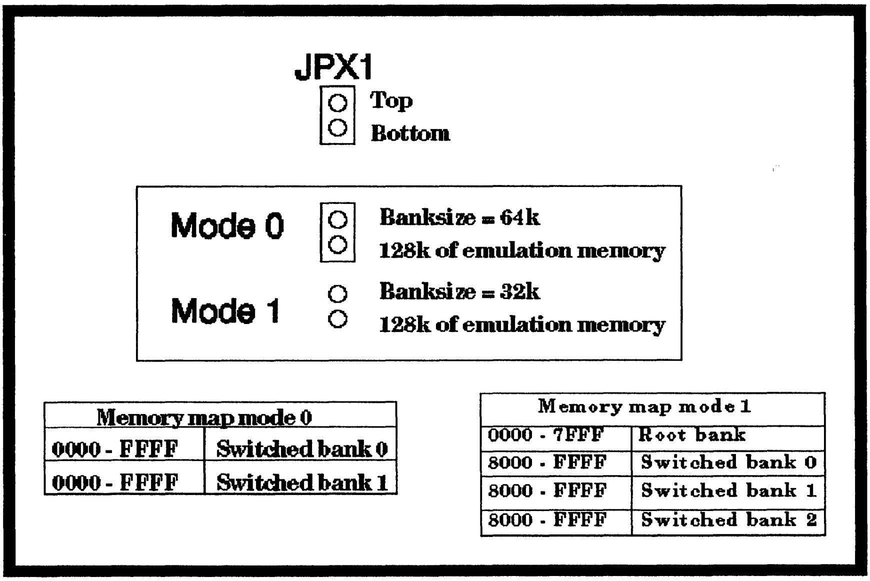
Figure 10. POD Board Modification for EMUL51-PC/E128-BSW

The EMUL51-PC/E128-BSW must be invoked as a 32k emulator (-m32). **BE AWARE!!** Do not modify the switch settings shown in Figure 11 below. This may cause improper function and damage to the emulator.



JPX1: 1 added jumper that is used to configure the mode of operation for the 128k bankswitch emulator.

Figure 11. Layout of EMUL51-PC/E128-BSW Bankswitch Emulator



Mode 0 — Banksize = 64k, 128k of emulation memory

Mode 1 — Banksize = 32k, 128k of emulation memory

Memory map mode 0	
0000 - FFFF	Switched bank 0
0000 - FFFF	Switched bank 1

Memory map mode 1	
0000 - 7FFF	Root bank
8000 - FFFF	Switched bank 0
8000 - FFFF	Switched bank 1
8000 - FFFF	Switched bank 2

Figure 12. Jumper Description and Memory Map for the EMUL51-PC/E128-BSW

Mode 0		
Green	Red	Bank
n/c	0	0
n/c	1	1

Mode 1		
Green	Red	Bank
0	0	0
0	1	1
1	0	2

Figure 13. Control Lines for the EMUL51-PC/E128-BSW Bankswitch Emulator

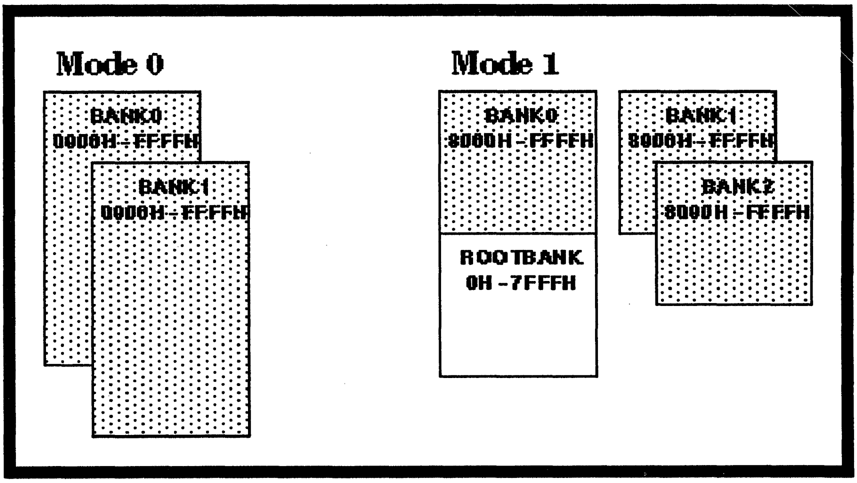
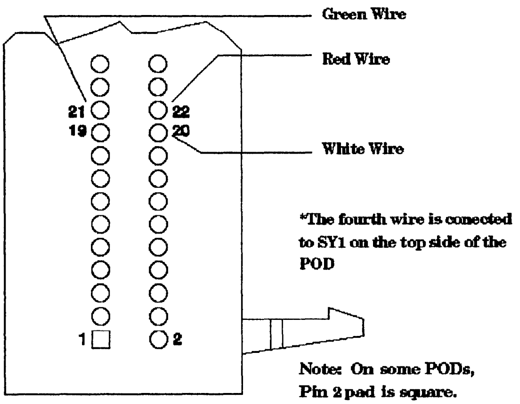


Figure 14. Bankswitch Modes EMUL51-PC/E128-BSW Graphic Illustration

Description of the Nohau EMUL51-PC/E256-BSW Bankswitch Emulator

The 256k bankswitch emulator has four different modes of operation which will be discussed later. This emulator can only emulate XDATA memory in mode 3. Any pod board can be modified to work as a bankswitching unit. For most operations, two or three wires will be used to control the bank selection. These wires must always be connected to the bankswitch logic. There is a special mode of operation which is customer specific mode and it requires that four wires are used to control its operation. See Figure 15 below for modifications for the POD board.



*The fourth wire is conected to SY1 on the top side of the POD

Note: On some PODs, Pin 2 pad is square.

POD BOARD, SOLDER SIDE

Figure 15. POD Board Modification for the EMUL51-PC/E256-BSW

The EMUL51-PC/E256-BSW must be invoked as a 32k emulator (-m32). **BE AWARE!!** Do not modify the switch settings shown in Figure 16 below. This may cause improper function and damage to the emulator.

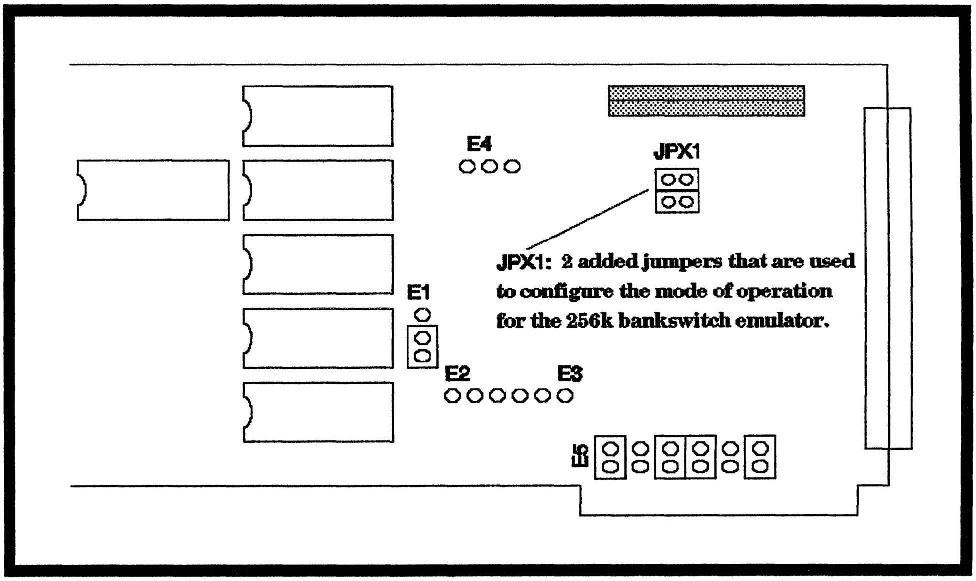
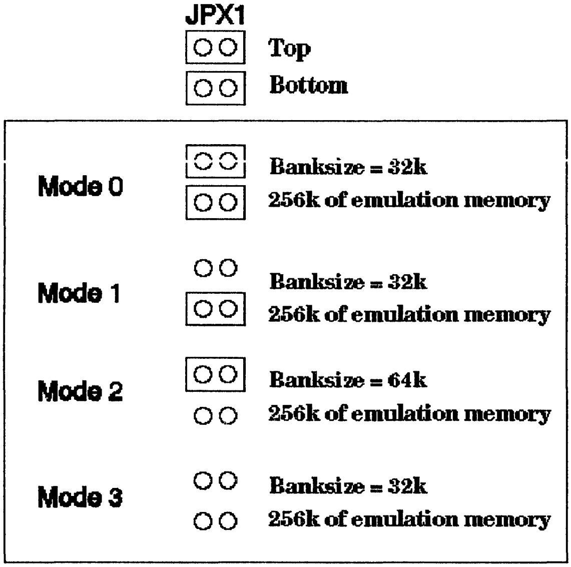

E4
JPX1
JPX1: 2 added jumpers that are used to configure the mode of operation for the 256k bankswitch emulator.
E1
E2
E3
E5

Figure 16. The EMUL51-PC/E256-BSW Bankswitch Emulator



Mode 0	
0000 - 7FFF	Root bank
8000 - FFFF	Switched bank 0
8000 - FFFF	Switched bank 1
8000 - FFFF	Switched bank 2
8000 - FFFF	Switched bank 3
8000 - FFFF	Switched bank 4
8000 - FFFF	Switched bank 5
8000 - FFFF	Switched bank 6

Mode 1	
0000 - 7FFF	Root bank
8000 - FFFF	Switched bank 0
8000 - FFFF	Switched bank 1
8000 - FFFF	Switched bank 2
8000 - FFFF	Switched bank 3
8000 - FFFF	Switched bank 4
8000 - FFFF	Switched bank 5
8000 - FFFF	Switched bank 6

Mode 2	
0000 - FFFF	Switched bank 0
0000 - FFFF	Switched bank 1
0000 - FFFF	Switched bank 2
0000 - FFFF	Switched bank 3

Code	Mode 3 (special)		Xdata
0000 - 7FFF	Switched bank 0	8000 - FFFF	Switched bank 2
0000 - 7FFF	Switched bank 1	8000 - FFFF	Switched bank 3
		8000 - FFFF	Switched bank 4
		8000 - FFFF	Switched bank 5

Figure 17. Jumper Description and Memory Map for the EMUL51-PC/E256-BSW

Mode 0				
SY1	White	Green	Red	Bank
n/c	0	0	0	0
n/c	0	0	1	1
n/c	0	1	0	2
n/c	0	1	1	3
n/c	1	0	0	4
n/c	1	0	1	5
n/c	1	1	0	6
n/c	1	1	1	root

Mode 1				
SY1	White	Green	Red	Bank
n/c	0	0	0	root
n/c	0	0	1	1
n/c	0	1	0	2
n/c	0	1	1	3
n/c	1	0	0	4
n/c	1	0	1	5
n/c	1	1	0	6
n/c	1	1	1	7

Mode 2				
SY1	White	Green	Red	Bank
n/c	n/c	0	0	0
n/c	n/c	0	1	1
n/c	n/c	1	0	2
n/c	n/c	1	1	3

Mode 3				Code	Xdata
SY1	White	Green	Red	Bank	Bank
X	0	X	0	0	0
X	1	X	0	1	1
0	0	1	1	0	2
1	0	0	1	0	3
0	0	0	1	0	4
1	0	1	1	0	5
0	1	1	1	1	2
1	1	0	1	1	3
0	1	0	1	1	4
1	1	1	1	1	5

Figure 18. Control Lines for the
EMUL51-PC/E256-BSW Bankswitch Emulator

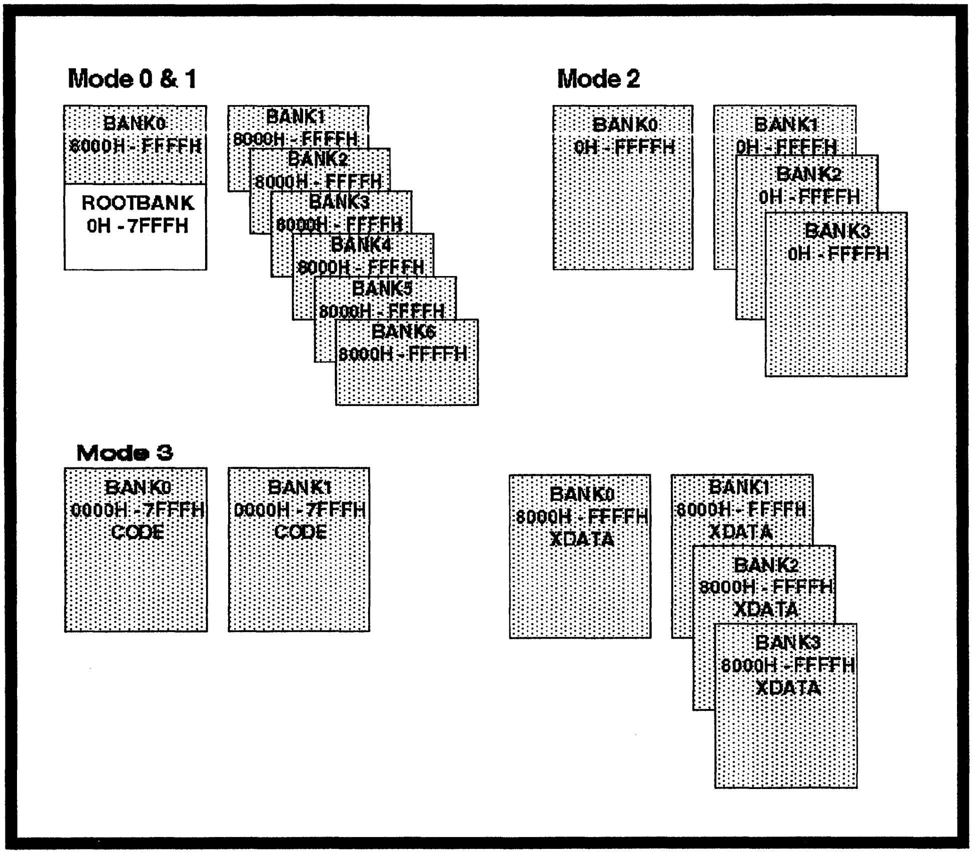


Figure 19. Bankswitch Modes of EMUL51-PC/E256-BSW, Graphic Illustration

3 COMMANDS

AASM	Assembly mode	3-5
ABR	Automatic Breakpoint Remove	3-5
ACC	Accum. display and set	3-7
ADJUST	Windows	3-7
ASM	Assemble one line	3-7
AV	View Auto Variable	3-8
B	*(see ACc)*	
BANK	Bank	3-8
BANKADDRESS	Bankaddress	3-9
BANKBYTE	Bankbyte	3-9
BANKMODULE	Define or display banked modules	3-9
BANKSEGMENT	Banksegment	3-10
BANKSY1	Set or show bank SY1 signal	3-10
BB	Break on internal bit	3-10
BIC	Break on Internal Contents	3-11
BR	Breakpoint registers	3-14
BRM	Breakpoint Mode Register	3-14
BRS	Breakpoint in Go Slow mode	3-15
BUFFERSIZE	Buffersize	3-15
BYB	Break on internal byte	3-16
CALCULATE	Calculate checksum	3-17
CB	Code memory	3-18
CLB	Clear Breakpoint RAM	3-19
COUNT	Execution of block	3-20
CST	*(see OST)*	
CW	Code Window On/Off	3-20
DASM	Disassemble	3-21
DBYTE	*(see CByte)*	
DEFINE	Define macro symbol	3-22
DIR	Macro directory	3-23
DISABLE	Disable an EMUL51 function	3-23
DOM	Module domain	3-26
DP	Disassemble from PC	3-26
DPTR	Data Ptr Reg display and set	3-26
ENABLE	Enable an EMUL51 function	3-27
END	End	3-29
EVALUATE	Evaluate an expression	3-29

3 COMMANDS

EVENT	(Simulator only) Specify a simulation event	3-30
EXIT	Exit to DOS	3-31
FILL	Fill memory with value	3-31
Format	Format Commands	3-32
GB	Go till Breakpoint	3-33
GI	Go till contents match	3-33
GO	Start emulation	3-34
GR	Go Register	3-36
GS	Go Slow	3-36
HEAP	Heap	3-37
HELP	Help	3-38
IB	Internal Break on pattern	3-38
IF	Conditional	3-38
INCLUDE	Load command file	3-39
INTERRUPT	Display interrupts	3-40
J	*(see SN)*	
LC	Loop Counter	3-40
LIN	Line step	3-41
LIST	Open log file	3-41
LOAD	Load program/symbols	3-42
LOB	Load Binary	3-44
LOP	Load Ppa setup command	3-44
LOVER	Line OVER	3-44
LOW	Load Window setup	3-44
MACRO	Display MACros	3-45
MACDELAY	(Simulator only) Macro after delay	3-45
MAPC	Map Code	3-46
MAPX	Map eXternal data	3-46
MEM	Mem	3-46
MOD	Module	3-47
N	*(see SN)*	
NOHIT	Nohit	3-47
NOSNOW	Remove screen snow	3-47
OST	Open Slow Trace buf	3-47
P1	*(see ACc)*	
P3	*(see ACc)*	

3 COMMANDS

PBYTE	*(see CByte)*	
PC	PC Reg display and set	3-48
PPA	Analyzer	3-48
PUT	Save macros in DOS file	3-49
QRA	Trace Qualifier Register	3-49
QUIET	Quiet	3-51
RBIT	Bit memory	3-51
RBS	Bank select	3-52
RBYTE	(see CByte)	
RCS	Relative Computer Speed	3-52
REG	Display regs.	3-52
REMOVE	Remove symbols or macros	3-53
REPEAT	Repeat block	3-54
RESET	Reset to initial status	3-55
R0	Display and set working reg.	3-55
R1-R7	*(see R0)*	
S	*(see Step)*	
SAP	Save Ppa setup command	3-56
SAVE	Save program or symbols	3-56
SAW	Save Window set up	3-57
SCOPE	SCOPE Command	3-57
SEB	Set Breakpoints	3-57
SECONDS	Display emulation cycles	3-58
SERIAL	(Simulator only) Map Serial Port	3-58
SHELL	Shell	3-60
SL	SL	3-60
SN	Step Next	3-60
SNAP	SNAP Command	3-62
SO	Source level debug	3-63
SP	*(see ACc)*	
SST	Single Step Trace	3-63
STACK	Display contents of stack	3-64
STEP	Execute one instruction and stop	3-64
SUFFIX	Display or set radix	3-65
SWD	Setup data window	3-65
SWR	Setup register window	3-65

3 COMMANDS

SWX	Setup ext. data window	3-66
SY0A	SY0a Reg. display and set	3-67
SYMBOLS	Display symbols	3-67
SYMLOAD	Load symbols from a converted assembly listing	3-67
SYSTEM	Execute DOS Command	3-69
TABS	Tab	3-69
TB	Trace Begin	3-69
TBR	Break on Trace trig	3-70
TC	Trig Counter	3-70
TD	Trace dissassemble	3-71
TDF	Frame display	3-71
TDL	TD in List mode	3-72
TE	Trace End	3-73
TM0	*(see DPTr)*	
TM1	*(see DPTr)*	
TR	Trace Register	3-73
TS	Trace setup	3-74
TSGET	Load Trace Setup	3-74
TSPUT	Store Trace Setup	3-74
UNTIL	*(see COunt)*	
VERSION	Version Number	3-75
WAIT	Wait for keystroke	3-75
WATCH	? and w[x]? - To Display C Variables	3-75
WHILE	*(see COunt)*	
WRITE	Display express. or string	3-76
XBYTE	*(see CByte)*	
XTAL	XTAL	3-76
:NAME	*(see MACro)*	
25, 43, 50	Screen Mode Commands	3-77
?	*(see WATCH)*	

3

COMMANDS

AASM

Automatic in-line ASseMbler

abbreviation **AA**

format AASM

description AASM enters a mode where you can only input instructions or ORG. To quit this mode press ENTER at the prompt. (AASM works like the ASM command except that you don't have to enter ASM (or A) in front of each instruction.)

Also see the ASM command.

examples

```
AASM
```

ABR

Automatic Breakpoint Remove

abbreviation **AB**

format ABR <start address> TO <stop address>

description The ABR command is always used in conjunction with the SEB command (SEt Breakpoints) and the GB command (Go till Breakpoint). First the SEB command is issued to set the complete 64K bit breakpoint RAM to all 1's. Using the GB command at this point would break immediately at the first instruction. The ABR command, however, should first be used to remove breakpoints from all opcode addresses which are within the legal program. Normally, legal code resides in a number of different blocks within the 64K address space of the 8051. Breakpoints must be removed from each of these blocks separately.

The ABR will remove only the breakpoint of the first byte in each instruction. Second and third bytes will still have breakpoints. Data tables within the code must be taken into consideration. If you include a data table within a block of legal instructions, the ABR will treat the data table as if it were instructions. In such a case, ABR will then remove breakpoints from some bytes of the data table and then possibly on the wrong bytes of the rest of the block, (because the breakpoint removing program will get out of sync).

It is therefore extremely important that you remove breakpoints from all legal program blocks, but it is equally important that they are only removed from legal program blocks. Legal program blocks include the RESET vector at address 0 as well as any active interrupt vectors. The GB command should then be used. Now if code is executed outside of the legal blocks that you programmed, a breakpoint will occur. By looking back in the trace you should be able to figure out what caused it to happen. Without the trace option it may be more difficult to see exactly what actually happened. (Do NOT use the GO command, because it will erase all breakpoints and put in new breakpoints according to the BR registers.)

examples

```
SEB              ;set all breakpoints
AB 0 TO 0        ;remove breakpoint on reset vector
AB 13 TO 13      ;remove breakpoint on external int 1 vector
AB 100 TO 45F    ;remove breakpoints from legal program block

                 100H to 45FH
AB 470 TO 1134   ;remove breakpoints from legal program block
                 470 to 1134.  Maybe there was a data table
                 between 460 and 470.
RES CHI          ;start from scratch (not neccesary)
GB               ; Go till Breakpoint.
```

ACC

Display and Set Commands for special registers

format

ACC [= <value>]
B [= <value>]
P1 [= <value>]
P3 [= <value>]
SP [= <value>]

description

These commands provide a way to display and set the Accumulator and B, P1, P3 and SP Registers.

examples

```
ACC         ;Display contents in the Accumulator.
ACC = 68    ;Set Acc to 68H.
B           ;Display contents in the B register.
B = 54      ;Set the B register to 54H.
SP          ;Display contents in the SP register.
SP = 30     ;Set SP to 30H
P1 = FA     ;Set Port P1 to FAH.
P3 = C0     ;Set Port P3 to C0H.  (Always keep P3.6 and
            P3.7 high, except for bondout PODs and
            POD-51.)
```

3

ADJ

Adjust windows command

abbreviation

AD

format

ADJUST

description

The ADJUST command will make the ADJUST window appear on the screen. Here you can choose which window to adjust. You can also use the function keys F4 and F5 to do the same thing.

examples

```
AD
```

ASM

The one line assembler

abbreviation

A

format

ASM [ORG address]
[instruction]

description

ASM will either display or set the EMUL51-PC's assembly pointer, or assemble one instruction into code memory at the assembly pointer location. It displays the updated assembly pointer after assembling any instruction.

Addresses of JUMP and BRANCH instructions are written as absolute addresses. The one line assembler will automatically calculate the offset in the case of a relative jump.

Bits are denoted using a "+" and not a "."
See also the AASM command.

examples

```
A JNB  .P1+4,.LOOP
A CLR  .ACC+2
A SETB .B+5
```

[ORG address] sets the assembly pointer to the specified address

[instruction] can be any valid instruction mnemonic except CALL and JMP, (which are "generic instructions").

examples

```
A                   ;displays assembly pointer
A ORG 200H          ;sets assembly pointer to 200H
A CJNE A,
.DAT,.LOOP          ;assembles instruction
A JB .P1+5,
1000                ;An offset to address 1000H will be from the
                    current position.
```

AV

View Auto Variable

format AV [C stackvariable] [type (i, c, u, s, f, l)]

description The ? and w? can give you better results than AV. For a detailed discussion of this command and source level debugging in general, refer to the Source Level Debugging section in Chapter 1.

BANK

Bank

format BANK [bank]

description Displays or sets current bank. Actually displays or sets the contents of
BANKBYTE.

BANKADDRESS Bankaddress

format BANKADDRESS [address1] TO [address2]

description Where [address1] TO [address2] is the "banked address area". This command MUST be issued BEFORE the user file is loaded. This is because the loader uses this information to "mark" the bank number.

The default values are 8000 to FFFF.

BANKBYTE Bankbyte

format BANKBYTE [address] [memtype] [mask]

description Where [address] is a dedicated address in internal data memory, DB, register memory, RB, or in external data memory, XB. The mask determines which bits in the byte are used. Bits must not be inverted and must be in the correct order of significance. If the type is RB and the address is a port, the bankbyte can be used by both the emulator and program as both status and control of the current bank. The contents of this address must "shadow" current bank. In other words: the user's bank switching code must write to this address when bank switching is done.

BANKMODULE Define or Display banked modules

format BANKMODULE [banknumber modulename]

description This command can set or show the association between a module and bank number. [module name] is the module name in the symbol table that your assembler or compiler must produce. Only Intel's OMF is supported. Many modules can be associated with one bank, but you must enter them one by one. All symbols in a bank should be covered by the correct number of modules present in that bank. Module and bank boundaries should coincide.

This command should be issued BEFORE the user file is loaded if you use the "module method" to indicate bankswitch while loading symbols. This is because the loader uses this information to "mark" the bank number.

BANKSEGMENT Banksegment

format BANKSEGMENT [number] [segment name]

description This command can set or show the association between a segment and a bank number. [segment name] is the segment name in the symbol table that your assembler or compiler must produce. Only Intel's OMF is supported. Many segments can be associated with one bank, but you must enter them one by one (see example in Chapter 2). All symbols in a bank should be covered by the correct number of segments present in that bank. Segment and bank boundaries should coincide.

This command should be issued BEFORE the user file is loaded if you use the "segment method" to indicate bankswitch while loading symbols. This is because the loader uses this information to "mark" the bank number.

BANKSY1 Set or show bank SY1 signal

format BANKSY1 [=OFF | ON]

description This command is for tracing banks with more than two bank select wires. This SY1 line is in addition to the E0 and E1 lines for bankswitched tracing.

BB Break on write to Bit address

format BB <bit address>

description The *<bit address>* is between 0 and FFH. This command will go through the entire code memory and look for the following instructions:

```
CLR bit
CPL bit
JBC bit, rel
MOV bit, C
SETB bit
```

When any of them is found, a corresponding breakpoint is activated in the breakpoint RAM. Then when the GB command is used, these breakpoints will break the program when a bit is written to the specified bit address. NOTE that only DIRECT bit writes will work. You can combine with the BYB command. (See the BRK paragraph.)

examples

```
BB 4     ;set up breakpoints on bit writes to bit address 4
BB .EA   ;set up breakpoints on bit writes to the EA bit address. (In
          the IE register.)
```

BIC Set Breakpoints on Internal Contents

abbreviation BI

format BIC <address>, <low value>, <high value>, <low code address>, <high code address>, <type>

where:
<address> is the byte address of either an internal data byte or a (DBYTE) Special Function Register (SFR).
<low value> is the lower limit of values on which to break.
<high value> is the higher limit of values on which to break.
<low code address> start address for "code scan".
<high code address> stop address plus one for "code scan".
<type> must be DB (DBY, DBYT, DBYTE) or RB (RBY, RBYT, RBYTE)

description BIC will scan the 64K code area for all instructions, that if executed, can change the contents of the specified address. On all such instructions a breakpoint will be set at its corresponding address. These breakpoints will be treated as regular breakpoints if you use the GB command to execute the code. In other words, this would work as BB, BYB and IB. When using the GI command, however, the breakpoint will be taken, but if the contents of the address specified in BIC is not between (and including) <low value> and <high value>, the program execution will resume. This check takes approximately 290 cycles, which is equivalent to 290 microseconds if a 12 MHz crystal is used. (For more details, refer to the GI command.)

All previous breakpoints are automatically cleared from the breakpoint RAM before BIC starts to put in new breakpoints. (This does not affect the BRx registers, which are used for GO.)

Different categories of instructions can change the contents of an internal address. When BIC is executed, messages will be written on the screen that tell which category is presently scanned for. The following messages can be seen:

Scanning code (Direct byte). Takes time. Please wait...
Scanning code (Direct bit). Takes time. Please wait...
Scanning code (Indirect byte). Takes time. Please wait...
Scanning code (Register). Takes time. Please wait...
Scanning code (DPL, DPH). Takes time. Please wait...
Scanning code (PUSH). Takes time. Please wait...
Scanning code (CALL). Takes time. Please wait...
Scanning code (B-REG). Takes time. Please wait...
Direct byte: Direct write to a byte address, like MOV addr,A. See BYB listing of all "direct write byte" instructions.

Direct bit: Direct write to a bit address,like CLR bit. See BB for a lislting of all "direct write bit" instructions.

Indirect byte: Only in the DBYTE area. Scanning occurs for the following instructions:
INC @R0; INC @R1; DEC @R0; DEC @R1; MOV @R0,#data;
MOV @R1,#data; MOV @R0,data; MOV @R1,data; XCH A,@R0;
XCH A,@R1; XCHD A,@R0; XCHD A,@R1; MOV @R0,A; MOV @R1;

Register: Only in the DBYTE area on addresses under 20H. Scanning occurs for the following instructions:
INC Rn; DEC Rn; MOV Rn,#data; MOV Rn,data; XCH A,Rn; DJNZ Rn,addr; MOV Rn,A; where n is 0 - 7.

DPH, DPL: Only in the RBYTE area. Direct byte writes already covered by "Direct byte" above. The following two instructions can however also influence DPH and DPL:
MOV DPTR, #data; INC DPTR;

PUSH: Only in the DBYTE area and for SP (81H) in the RBYTE area. Following instruction: PUSH data;

B-REG: Only for the B-register (F0H) in the RBYTE area. Direct writes to B are covered by "Direct byte" above. The following two instructions can however also influence B: DIV AB; MUL AB;

The following SFR:s are not fully covered:

ACC (E0H)

PSW (D0H).
All timer registers (TH0, TL0, TH1, TL1, TH2, TL2)
Port pins are covered, but not Port latches. (See Read-Modify-Write in 8051 data books.)
All SFR:s where hardware directly changes bits, (like TCON, SCON and SBUF).

Note 1. When using GI, if a breakpoint is not reached and you press ESC to manually break, you will get WRITE TIMEOUT and READ TIMEOUT errors. This happens because the code which is handling breakpoints is modified to check if the contents of the specified address is within range. If this occurs YOU MUST ISSUE THE COMMAND "* RES EMUL" which reloads the 8051 overlay code.

Note 2. When you look back in the trace buffer after a break has occured after using the GI command, you will see the instruction:

```
JBC .IE+7(EA),addr
```

or, if you use the TDF command, a number of frames starting with the following data:

```
10 AF 01 00
```

This happens because the trace will always sample the first instruction in the breakpoint handling routine (which is JBC .EA, addr) before it turns off. This is normally taken care of in software so you will not see it. In this case we decided to let it show for a number of reasons: You get an indication where "check breaks" were taken. It is also difficult to remove part of the trace without inadvertently removing "valid" trace frames.

examples

```
BIC 30 40 50 0 200 DB
            ;prepare breakpoints for all instructions between address 0
            and 200H which may change the content of address 30H.
            Break when contents of 30H is between (and including)
            40H and 50H.
BIC .TJ F0 20 0 200 DB
            ;as previous example. ".TJ" must be previously defined via
            DEF or via symbol table. Range is now F0, F1...FF, 00,
            01...20
BIC .SP 23 23 0 200 RB
            ;Break when SP (stackpointer 81H) becomes 23H if
            between addresses 0 and 200H
```

BR

Breakpoint Registers display and set

format

```
BR
BRx  (where x is any number from 0 to 9)
BRx = address
BRx = address1 TO address2
BRx = address LENGTH number
BRx = wild card expression
```

description

There are 10 breakpoint registers. BR2 - BR9 must be programmed directly with the BR command. Only BR0, BR1, and BR (BR0 or BR1) can be set inside the GO command. See "Breakpoints" in Chapter 1. Breakpoints are enabled by the GR (Go Register), or directly within the GO command. Breakpoints can also be set up directly within the GO command. (See GR and GO.) Breakpoints can be disarmed by setting the GR to FORever. Breakpoints can be reset with the REset command: for example RES BR2 or RES BR.

examples

```
BR        ;displays all 10 breakpoint registers.
BR0       ;displays breakpoint register 0.
BR9       ;displays breakpoint register 9.
BR2 = 124A TO 125F
          ;sets breakpoint register 2 to addresses between 124AH
          and 125FH
BR6 = 96E L 10
          ;sets breakpoint register 6 to addresses starting at 96EH
          and 16 addresses ahead to 97EH.
BR1 = XXX
          ;X means don't care, in this case all addresses with first 4
          bits = 0000.
BR1 = XXXY
          ;Y denotes binary, in this case all addresses starting with
          0000 0000 0000 0
BR1 = 0101XXX010XXY
          ;Any combination is allowed.
```

BRM

BReakpoint Mode register

format

```
BRM
BRM = [OP] [WR] [RD]     (any combination)
```

description

The BRM command triggers breakpoints on OP (opcode addresses), WR (external write addresses), RD (external read addresses) or any combination of these addresses.

examples

```
BRM             ;displays current status of BRM
BRM = RD        ;breaks only on external read addresses
BRM = WR        ;breaks only on external write addresses
BRM = OP        ;breaks only on opcode addresses. (default).
BRM = RD WR     ;breaks on external read or write addresses
BRM = OP RD WR
                ;breaks on opcode, external read, or external write.
```

BRS

Breakpoint Registers for "Slow" breakpoints

format

BRS
BRSx (where x is any number from 0 to 9)
BRSx = <partition> <type> [address]

description

Displays or sets BRSx registers. These breakpoints are "software breakpoints" so they only work when you use the GS (Go Slow) command.

<partition> is any range or wildcards, normally used only for addresses. Here they stand for "memory contents" unless <type> is ADDR.

<type> specifies one of the following memory types or its abbreviations: CBYTE, DBYTE, PBYTE, RBYTE, XBYTE or ADDR.

[address] is a unique address for which the "content" should be tested for a match with <partition>. The address has no meaning when <type> is ADDR, in which case <partition> gives the address or range of addresses.

examples

```
BRS      ;displays all BRS registers.
BRS0     ;displays the BRS0 register.
BRS1 = 10 TO 20 DB 30
          ;sets BRS1 to break when the contents of internal databyte at
          address 30H becomes a value between 10H and 20H.
```

BUFFERSIZE

Buffersize

format

BUF

description

Displays the size of the trace buffer, the location of the first frame, and the location of the last frame. Display will not work when the trace is running.

example

```
BUF
Stored frames:      9014
First frame:        -2045
Last frame:         -6969
```

BYB

Break on direct write to Byte address

format

BYB <byte address>

description

The <byte address> is between 0 and FFH. This command will go through the entire code memory and look for the following instructions:

```
ANL dir,A
ANL dir,#data
DEC dir
DJNZ dir,rel
INC dir
MOV dir,A
MOV dir,R0
MOV dir,R1
MOV dir,R2
MOV dir,R3
MOV dir,R4
MOV dir,R5
MOV dir,R6
MOV dir,R7
MOV dir, dir
MOV dir,@R0
MOV dir,@R1
MOV dir,#data
ORL dir,A
ORL dir,#data
XCH A,dir
XRL dir,A
XRL dir,#data
```

When any of these instructions is found, a corresponding breakpoint is activated in the breakpoint RAM. Then when the GB command is used, these breakpoints will break the program when a byte is written to the specified byte address. NOTE: only DIRECT byte writes. You can combine with the BB or IB commands. (See the BRK paragraph.)

examples

```
BYB 4          ;set up breakpoints on byte writes to bit address 4.
BYB .IE        ;set up breakpoints on byte writes to the IE register).
```

CAL

Calculate checksum

abbreviation **CA**

format CALC <start address> TO <stop address>

description Calculates a 16-bit checksum by adding all bytes in code memory between and including <start address> and <stop address>. On a regular PC a calculation of 64K bytes takes approximately 1.5 minutes.

examples

```
CALC 0 TO 1FFF        ;calculates checksum of all bytes between 0 and
                      1FFF.
```

CB Display and change memory contents

abbreviation See Description

format See Description

description The following types of memory space can be displayed or changed via EMUL51-PC:

Type of Memory	*Command*	*Range*
On-chip data memory: Data Byte	DB, DBY, DBYT, DBYTE	0 - 7FH, (plus 80H - FFH if available)
Register memory: Register Byte	RB, RBY, RBYT, RBYTE	80H - FFH
Bit addressable memory: Register Bit	RBI, RBIT	0 - FFH
External data memory, (with read back check): eXternal Byte	XB, XBY, XBYT, XBYTE	0 - FFFFH
External data memory, (without read back check): Put Byte	PB, PBY, PBYT, PBYTE	0 - FFFFH
Code memory: Code Byte	CB, CBY, CBYT, CBYTE	0 - FFFFH

Commands to display and change memory contents can be used in many ways. The examples that follow demonstrate each possibility. In place of CB in these examples you can substitute any of the other commands listed above; in place of the numbers you can substitute any other numbers that fall within corresponding valid range listed above. Symbols defined in the internal or external symbol table may also be used. See also FILL.

examples

```
CB 23               ;Displays data byte on address 23.
CB 23 TO 35         ;Displays data bytes on addresses 23 through 35.
CB 23 L 10          ;Displays data bytes on address 23 + another 10
                    bytes. L means length and can also be
                    expressed as LEN or LENGTH.
CB 23 = 10          ;Changes the contents of address 23 to 10.
CB 23 = 10, 14, 15
                    ;Same as previous example, but contents of
                    addresses following the first address are also
                    changed to the data separated by commas.
CB 23 TO 70 = 12
                    ;All addresses from 23 through 70 are filled with
                    12.
CB 23 L 10 = 12
                    ;Addresses from 23 + another 10 addresses are
                    filled with 12.
CB 23 TO 45 = 12,13,14
                    ;Addresses from 23 through 45 are filled with a
                    repeating pattern of 12,13,14.
CB 23 L 10 = 12,34,56
                    ;Same as previous example.
CB 100 = XB 400 TO 40F
                    ;Specifies addresses from other memory areas on
                    the right side of = sign.
```

The above two examples can be combined in a number of ways:

```
CB .LOOP L 100 = 12,34,'ABC',XBYT 1000 TO 1010
```

'ABC' is evaluated as three bytes with ASCII equivalents of A, B and C, (41H, 42H, 43H).

NOTE: If used to the right of the = sign, CB and DB must be written as CBY and DBY, (because otherwise they would be interpreted as the hexadecimal numbers CBH and DBH).

CLB

CLear Breakpoint RAM

format CLB

description This command clears the breakpoint RAM. The BB, BYB, BIC and IB commands can be used to set patterns of breakpoints to the breakpoint RAM; they will overlay patterns in the RAM without destroying earlier patterns. The GO command will also clear the breakpoint RAM before putting in active breakpoints as defined by the BR registers. After the GO command these breakpoints are still active and can thus be combined with more patterns created with the above mentioned commands. This special command is different than simply resetting the breakpoint registers: see the RESet and BR commands.

COU Count

abbreviation CO

format

```
COUNT decimal-expression<cr>
        [command<cr>]
        [WHILE boolean-expression<cr>]
        [UNTIL boolean-expression<cr>]
ENDCOUNT
```

description This command sets up loops. To end the command before the full number of loops are done, just press the ESC key. It may take a couple of loops before the break occurs.

COUNT begins a block of commands to be executed a specific number of times.

decimal-expression is a dec. number or expression in which default radix is decimal.

<cr> is an intermediate carriage return. In a block command such as Count, all carriage returns are intermediate, (terminating the input line but not terminating the command entry), until the ENDCOUNT keyword is entered.

WHILE and *UNTIL* are keywords.

boolean-expression is the least significant bit in an expression. Condition is TRUE if the least significant bit is 1, otherwise it is FALSE.

ENDCOUNT terminates the block of commands; it can be abbreviated to END.

CW Code Window On and Off

format CW

description This command toggles the Code Window on and off. (You can also open the Code Window by using ENA CODE, and close it using DIS CODE.) Two modes of display, "assembly" and "source," are available when the Code Window is open. (You use the F2 Function Key to toggle between Assembly and Source.)

In Assembly Mode, which is enabled by default, the disassembler is used to generate the text in the Code Window. Source Mode is usable only when you are working with either the Archimedes or Franklin/Keil C-compiler, or with a PL/M-51 compiler. For more details about Assembly and Source Mode, see Code Window in Chapter 1.

DAS

Disassemble

abbreviation D

format

DASM
DASM partition

description

This command disassembles the contents of user program memory into assembler mnemonic and operands.

If DASM without partition is used, 10H bytes starting from the address currently pointed to by the disassembly pointer are disassembled. The disassembly pointer will be automatically incremented, so that if you issue the D command again disassembly will continue from where the previous disassembly command ended.

In the above format, the partition is a single address, or a range of addresses expressed as address TO address, or address LENGTH address. An instruction is displayed if its first byte is within the partition, even if subsequent bytes are outside.

The D command with a partition, and the DP command, will automatically change the disassembly pointer, so that it will be available for the next D command. See also the DP command.

examples

```
D .LOOP          ;disassembles one instruction starting on the label .LOOP
D 100 TO 200     ;disassembles from 100 and 200 addresses
D .START L 20    ;disassembles from .START and 20H addresses
D .#120 L 10     ;disassemble from PL/M line 120.  (Domain must be set.
                 See Domain command.)
D                ;disassembles 10H bytes starting from disassembly pointer.
```

DBYTE

Data Byte See CB

DEF

Define macro, Define symbol

abbreviation **DEF**

Define Macro

format

```
DEFINE :macroname<cr>
      [command<cr>]   ....
EM
```

description

This command allows you to enter a user defined macro name and block of commands into the macro definition table.

:macroname is the name of the macro block preceded by a colon.
<cr> stands for Carriage Return
EM Terminates the macro definition.
See also Macros in Ch. 1.

examples

```
*DEF :G
.*GO FROM .START TILL .END
.*R
.*EM
*
```

Define symbol

format DEFINE [..module-name].symbol-name = expression [type]

description

This command also allows you to define a symbol.

After the keyword *DEFINE* you type the desired user-defined symbol and corresponding numeric value into the user symbol table.
..module-name is the name of an existing module to contain the symbol being defined. If no module name is specified,

the symbol is placed in the unnamed module. Unnamed modules occurs before named modules in the symbols table. *.symbol-name* is the name of the symbol preceded by a period.
EXPRESSION is a number, reference or formula that equates to the number expressed.
type can be CODE, XDATA, DATA, IDATA, BIT, NUMB, LINE, or UNDEF.

examples

```
* DEFINE .MOTOR = 210T   ;add new symbol to unnamed module.
* DEF .FLAG = 34 BIT     ;new symbol gets BIT type.
```

DIR

Macro Directory

format DIR

description Displays a directory of the names of all macros. In the macro definition table, the names are listed in the order that they were entered.

DIS

Disable Command

abbreviation **DIS**

format

```
DISABLE   [CASE]
          [CODE]
          [EXPANSION]
          [GLOBAL]
          [HELP]
          [LABEL]
          [LSYMB]
          [NSYMB]
          [NUMB]
          [PLUS]
          [REG]
          [SIG]
          [SOURCE]
          [SYMBOLIC]
          [SYNTAX]
          [WATCH]
          [WINDOW]
          [XDATA]
```

description

This command cancels the effect of its object on the EMUL51-PC.

CASE By Default the characters in symbols are not case sensitive. ENA CASE will make them case sensitive while DIS CASE makes them non-sensitive.

CODE Disables the CODE window if it was active. By default it is not enabled.

EXPANSION When disabled, macro expansion is not displayed. When enabled, the macro expansion is displayed on the console before execution.

GLOBAL When enabled, ALL symbols will be searched. If disabled ONLY the current module will be searched. Disabling GLOBAL will speed up disassembly, trace disassembly and CODE WINDOW update. Disabled by default.

HELP When enabled, the small HELP window at the lower right of the screen is visible. Can also be toggled on and off with Alt + F10. Enabled by default.

LABEL By default labels are automatically shown during disassembly. DIS LABEL suppresses labels in disassembly listings.

LSYMB When disabled, no symbols are shown when listings generated by DBYTE, CBYTE, XBYTE, PBYTE, RBYTE, or RBIT are used. Disabled is default.

NSYMB When enabled, symbols typed as NUMB will be used in place of all other types. Useful only with Archimedes / IAR C-51, if you have used the XOBJ program to generate an OBJ file. Disabled is default.

NUMB When enabled, the symbol table will search through all variables defined as "NUMB". Normally "EQU'S". Disabling NUMB may significantly speed up disassembly, trace disassembly and code window update. Enabled by default.

PLUS See ENABLE comand.

REG Disables the REGS window.

SIG See ENABLE command.

SOURCE Enables the Source only in the CODE window.

SYMBOLIC When disabled, addresses are displayed in hexadecimal radix. When enabled, addresses are displayed in symbolic format. Symbols are initially enabled.

SYNTAX When enabled, context sensitive help line will be seen at the bottom of the screen when typing new commands. Enabled by default.

These commands, (which may be combined with the ENABLE command), can be used in an INCLUDE file to automatically initialize a desired setup. (See Chapter 2 under the *-i* option for the invocation line.)

WATCH Disables the WATCH window if it was active. It is disabled by default.

WINDOW When enabled, the static windows reflecting different memory locations are shown. Enabled by default.
XDATA Disables the EXTERNAL DATA window. When the EXTERNAL DATA window is on REA^D signals (/RD) will be emitted from the chip to either the target or emulator depending on how the mapping is set. That may create a problem in the target. By using DIS XDATA no /RD signals are emitted and the EXTERNAL DATA window will not be seen.

examples

```
DISABLE SYMBOLIC
DIS EXPANSION
DIS LSY
DIS NSY
DIS WI
DIS H
DIS SYN
DIS NUMB
DIS GLO
DIS SO
DIS PLUS
DIS XDATA
```

DOM — Domain Command

abbreviation **DO**

format DOMAIN [..module-name]

description This command displays or sets the default module-name. The default is used for all line number references not specified with a module-name. Symbol references are not affected by this command.

..module-name is an existing module-name that becomes the default module for referencing line numbers.

examples

```
DOMAIN ..ETON1
```

DP — Disassemble from PC

format DP

description Disassembles 16 bytes, starting from where PC (Program Counter) points. This command will change the disassembly pointer, (see DASM).

DPTR — Display or set Data Pointer Register, Timer 0, Timer 1

abbreviation **DPT**

format

```
DPTR = [expression]
TM0 = [expression]
TM1 = [expression]
```

description DPTR Displays or sets the 16-bit data pointer register.
TM0, TM1 Displays or sets the 16-bit value of Timer 0 or Timer 1.

EXPRESSION is a number, reference or formula that equates to the number expressed; 16 bits are assumed.

For DPTR the low 8 bits are stored in DPL (address 82H in Register memory). The high 8 bits are stored in DPH (address 83H in Register memory).
For TM0 the low 8 bits are stored in TL0 (address 8AH in Register memory). The high 8 bits are stored in TH0 (address 8CH in Register memory).
For TM1 the low 8 bits are stored in TL1 (address 8BH in Register memory). The high 8 bits are stored in TH1

(address 8DH in Register memory).

examples

```
DPTR            ;display data pointer
DPTR = .TABLE   ;sets to 16-bit value.
TM0             ;displays TM0
TM1 = 1A5D      ;sets TM1 to 1A5DH
```

ENA Enable

abbreviation EN

format

```
ENABLE      [CASE]
            [CODE]
            [EXPANSION]
            [GLOBAL]
            [HELP]
            [LABEL]
            [LSYMB]
            [NSYMB]
            [NUMB]
            [PLUS]
            [REG]
            [SIG]
            [SOURCE]
            [SYMBOLIC]
            [SYNTAX]
            [WATCH]
            [WINDOW]
            [XDATA]
```

description The Enable command causes its object to affect the EMUL51-PC.

CASE By default the characters in symbols are not case sensitive. ENA CASE will make them case sensitive.

CODE Enables the CODE window. By deault it is not enabled.

EXPANSION When enabled, the macro expansion is displayed on the console before execution.

GLOBAL When enabled, ALL symbols will be searched. If disabled ONLY the current module will be searched. Disabling GLOBAL will speed up disassembly, trace disassembly and CODE WINDOW update. Disabled by default.

HELP When enabled, the small HELP window at the lower right of the screen is visible. Can also be toggled on and off with Alt+F10. Enabled by default.

LABEL By default labels are automatically shown during disassembly. DIS LABEL suppresses labels in disassembly listings. ENA LABEL makes labels appear.

LSYMB When enabled, appropriate symbols will interleave listings generated with CBYTE, DBYTE, XBYTE, PBYTE, RBYTE, or RBIT commands. Default is disabled.

NSYMB When enabled, symbols typed as NUMB will be used in place of all other types. Useful only with Archimedes / IAR C-51, if you have used the XOBJ program to generate an OBJ file. Disabled is default.

NUMB When enabled, the symbol table will search through all variables defined as "NUMB". Normally "EQU'S". Disabling NUMB may significantly speed up disassembly, trace disassembly and CODE WINDOW update. Enabled by default.

PLUS Enables the disassembler to display numbers as symbols followed by " + hex number " if a number not exactly matching a symbol was found. Default is disabled. For example, MOV DPTR, #4005 could be MOV DPTR, #LOOP+5 if PLUS was enabled.

REG Disables the REGS window.

SIG Enables the output from the Advanced Trace Board. The POD signal "S4" is called ANB in the Advanced Trace Setup.

SOURCE Enables the Source only in the CODE window.

SYMBOLIC When enabled, causes display of addresses in symbolic format. Symbols are initially enabled.

SYNTAX When enabled, context sensitive help line will be seen at the bottom of the screen when typing new commands. Enabled by default.

WATCH Enables the WATCH window. It is disabled by default.

WINDOW When enabled, shows static windows reflecting different memory locations. Enabled by default.

XDATA Enables the EXTERNAL DATA window. Normally the XDATA WINDOW is on by default except when "bondout" pods or some of the "hooks pods" are used. When the EXTERNAL DATA window is on READ signals (/RD) will be emitted from the chip to either the target or emulator depending on how the mapping is set. That may create a problem in the target. By using DIS XDATA no /RD signals are emitted and the EXTERNAL DATA window will not be seen.

These commands, (which may be combined with the DISABLE command), can be used in an INCLUDE file to automatically initialize a desired setup. (See Chapter 2 under the *-i* option for the invocation line.)

examples

```
ENABLE SYMBOLIC
EN SYM
EN EXP
EN LSY
EN WI
EN H
EN SYN
EN NUMB
EN GLO
EN REG
EN SO
EN PLUS
EN XDATA
```

END

END Command

The END command is used to end a REPEAT or COUNT loop. Please see these commands for further information.

EVA

Evaluate

abbreviation **E, EVA, EVAL**

format EVALUATE expression

description The Evaluate command performs the specified mathematical operations to convert the expression to a 16-bit numeric result. The result is displayed in binary, octal, decimal, and hexadecimal, as a pair of ASCII characters (if value is between 20H and 7EH), and as an address. The address displayed is the name of the user symbol or a line number.

EXPRESSION A number, reference, or a formula that can be equated to a numeric result.

examples

```
EVAL 100Q + 13T
E CBYTE .VAL + 100
```

EVENT

(Simulator Option Only)

Specify a simulation event.

abbreviation EVE

format EVENT {R,W,*} <addr[range]>
{CB,XB,DB,RBY,RBI,PC} [lo hi value]]
[:mac]

description This is a simulator command which specifies events to be intercepted. The default action is 'break'. However, the user can attach a MACRO that will be executed automatically without stopping.

{ R,W,*} Read/Write access or both
<addr[range]> address
address LEN count
address TO address

{CB,XB,DB,
RBY,RBI,PC} memory type or program counter
[lo [hi value]] optional contents of address is included in the event specification. A single value or a range.

[:mac] an optional macro
It will be called with 3 operands:

:MACRONAME ADDRESS COUNT CYCLES

where

ADDRESS is the trapped address,
COUNT is an individual event count,
CYCLES is a machine cycle counter, all modulo 64K.

Note: events are only sensitive to instruction execution, NOT to memory accesses from the command line (to avoid the risk of infinite nesting). However, macros can install or remove events!

EVENT without arguments will list the installed events, showing their ID numbers and counts.

see REMOVE EVENT number
REMOVE EVENTS
RESET EVENTCOUNT number
RESET EVENTCOUNTS

examples

```
EVENT W .p0 RBY :setP0
EVENT * 8000 TO FFFF PC
EVENT R 0 TO FFFF XB
```

EXI

EXIT and QUIT Commands

abbreviation **EX QUI**

format

EXIT
QUIT

description

EXIT; Terminates EMUL51-PC and returns to DOS.
QUIT; Returns to DOS without terminating the emulator if it was running.

FIL

FILL

abbreviation **F**

format FILL <range> <value> <type>

description

The Fill command writes a specified value to a range of addresses of a specified type. You can do the same using CB, DB, XB, etc., but the FILL command is much faster.

examples

```
F 0 TO FFFF 55 XB       ;writes 55H to the external memory.
F 30 L 20 0 DB          ;clears internal data memory from address 30H
                        and the following 20H bytes.
```

FORMAT

Format Commands (In support of C-Compilers)

abbreviations

DI, DU, DF, DS, XI, XU, XF, XS, CI, CU, CF, CS

format

```
DINT     address [range]
DUINT    address [range]
DFLOAT   address [range]
DSTRING  address [range]

XINT     address [range]
XUINT    address [range]
XFLOAT   address [range]
XSTRING  address [range]

CINT     address [range]
CUINT    address [range]
CFLOAT   address [range]
CSTRING  address [range]
```

Where:

[range] is either of the form
[<address> TO <address>] or
[<address L <number>]

x in the descriptions below is either D, X, or C

description

xINT will display the contents of <address> and <address+1> as a signed integer. If a range is specified, signed integers will be displayed until the range is exhausted.

xUINT will display the contents of <address> and <address+1> as an unsigned integer. If a range is specified, unsigned integers will be displayed until the range is exhausted.

xFLOAT will display the contents of <address>, <address+1>, <address+2> and <address+3> as a float. If a range is specified, floats will be displayed until the range is exhausted. NAN and -NAN (Non Applicable Number) means that the four bytes could not be translated to a float.

xSTRING will display the contents of <address> as an ASCII character or a ".", if the byte was not translatable as a printable ASCII number. If a range is specified, ASCII characters will be displayed until the range is exhausted or a 0 is encountered. If no range is specified ASCII characters will be displayed till a 0 is encountered or maximum 16 characters have been printed. The ASCII string will be surrounded by quotes ("). The ending quote will only be seen if the ending 0 was found within the range (or 16 bytes if no range was specified).

examples

```
CI 0             ;Display CODE memory address 0 and address 1
                 ;as a signed integer
DI 30 L 10       ;Display DATA memory from address 30H to address
                 ;40H as signed integers
XU 3000 TO
3020             ;Display XDATA memory from address 3000H to
                 ;address 3020H as unsigned integers
XF 4000 L 40     ;Display XDATA memory from address 4000H to
                 ;address 4040H as floats
DS 0             ;Display DATA memory adress 0 as an ASCII
                 ;character. Up to 16 ASCII characters will
                 ;be displayed unless a 0 is encountered
XS 1000 L 100    ;Display XDATA memory starting from address 1000H
                 ;as ASCII characters. Display will stop if a 0 is
                 ;encountered or when the range is exhausted
```

GB

Go until Break

format

GB [FROM <address>]

description

The GB command starts execution without first erasing the breakpoint RAM, and is useful after breakpoint RAM has been initiated by BB, BYB, IB, SLM and/or BIC.

GI

Go till Internal contents match

format

GI [SY0]
GI [FROM addr] [SY0]

description

This command is used together with the BIC command which presets breakpoints for the desired address, and also specifies which range of contents to break on. (See BIC for all details.)

Note 1: If, when you use GI, a breakpoint is not reached and you use ESC to manually break, you will get WRITE

TIMEOUT and READ TIMEOUT errors. This is because the code which is handling breakpoints is modified to check if the contents of the specified address is within range. If this occurs, YOU MUST ISSUE THE COMMAND "* RES EMUL" which reloads the 8051 overlay code.

Note 2: When you look back in the trace buffer after a break has occured after using the GI command you will see the instruction:

```
        JBC .IE+7(EA),addr
```

Or, if you use the TDF command, you will see a number of frames starting with the following data:

```
        10 AF 01 00
```

This is because the trace will always sample the first instruction in the breakpoint handling routine, (which is JBC .EA, addr), before it turns off. This is normally taken care of in software so you will not see it. In this case we decided to let it show, for a number of reasons: You get an indication where "check breaks" were taken. It is also difficult to remove part of the trace without inadvertently removing "valid" trace frames.

THIS COMMAND SHOULD ONLY BE USED AFTER "BIC" HAS FIRST SET UP BREAKPOINTS AND SPECIFIED ADDRESS AND RANGE OF CONTENTS!

examples

```
GI              ;Go from present PC (Program Counter)
GI SY0          ;As example above but qualified with SY0
GI FROM 0       ;Go from address 0
GI F 100        ;Go from address 100H
GI F 100 SY0    ;As example above but qualified with SY0
```

GO

GO

format

```
GO [FROM address] [FOREVER]
                  [TILL SY0]
                  [TILL BR0]
                  [TILL BR0 WITH SY0]
                  [TILL BR0 OR SY0]
                  [TILL BR1]
                  [TILL BR1 WITH SY0]
                  [TILL BR1 OR SY0]
                  [TILL BR]
                  [TILL BR WITH SY0]
```

[TILL BR OR SY0]
[TILL address [OR address]
[TILL address [OR address] WITH SY0]
[TILL address [OR address] OR SY0]

description

This command begins the real time emulation.

FROM specifies the start address for emulation and changes the program counter before the start of emulation. FROM also resets the emulation timer (cycle counter). If the FROM is omitted, the current PC is used.

FOREVER disables all breakpoints except the ESC key and TBR = ON.

TILL SY0 halts emulation when the external signal SY0 goes low (or high, depending on the SY0A register).

TILL BR0, TILL BR1, TILL BR Halt real-time emulation when an address matches BR0, BR1 or BR (BR0 or BR1).

Breakpoint registers BR2 - BR9 may also be set up to break emulation, but not with the GO command. Instead they must be set up separately with the BR command.

WITH SY0 Halts emulation when a breakpoint condition and SY0 are simultaneously true. (SY0 true can be high or low, depending on the SY0A register.)

OR SY0 Halts emulation when either a breakpoint condition is true, or SY0 is true (high or low, depending on the SY0A register).

examples

G	;starts emulation at present PC value. All breakpoint factors remain unchanged
G F .START	;starts emulation at .START. All breakpoints factors remain unchanged.
G F .START T .LOOP	;starts emulation at .START. BR0 is set to .LOOP. Other breakpoints remain unchanged.
G F .START T .LOOP OR .STOP	;starts emulation at .START. BR0 is set to .LOOP and BR1 is set to .STOP. Other breakpoints remain unchanged.
G F .START TILL BR	;starts emulation at .START and will end when BR0 or BR1 is encountered, (or any of BR2 - BR9, if they are active).
G F 0 T BR W SY0	;SY0 is part of the break condition.
G F 0 T BR0 OR SY0	;BR0 or SY0 true breaks emulation.
G T SY0	;SY0 true breaks emulation.

GR

Go Register

format

```
GR
GR =  [FOREVER]
      [TILL SY0]
      [TILL BR0]
      [TILL BR0 WITH SY0]
      [TILL BR0 OR SY0]
      [TILL BR1]
      [TILL BR1 WITH SY0]
      [TILL BR1 OR SY0]
      [TILL BR]
      [TILL BR WITH SY0]
      [TILL BR OR SY0]
      [TILL address [OR address]
      [TILL address [OR address] WITH SY0]
      [TILL address [OR address] OR SY0]
```

description

This command displays or sets halt conditions for real-time emulation without starting emulation. (See GO for other details.)

GS

Go Slow

format

GS [FROM <address>]

description

This command will make the emulator single step (without showing anything on the screen). Between each step the BRS conditions will be checked for matches. When a match occurs single stepping will break and registers will be shown on the screen. This break check is edge sensitive, which means that it will NOT break continuously on a constant match of memory contents. Only if a match first fails will it break on the next match. (See OST command for generating trace buffer.)

examples

```
GS              ;Go Slow from current PC address
GS F 10A8       ;Go Slow from 10A8H
```

HEAP Heap

abbreviation **HEA**

format

```
HEAP
HEAP = [STD] [EXT] [EMS] [XMS] [DISK]
```

The HEAP comand is used to "expand the heap" to memory located outside the standard 640K of the PC. This is useful if you have a very large symbol table that does not fit in the standard memory. The symbol table will be loaded to the specified memory location instead of to the standard memory. Up to 64K of the standard memory will be allocated in some cases to handle the "outside memory". Use the MEM command to see how much memory is left after the emulator software has been loaded, after your file has been loaded, etc.

STD = standard memory (640K)
EXT = extended memory
EMS = expanded memory (LIM)
XMS = extended memory specification (DOS 5.0)
DISK = file in current directory
The HEAP command can only be used once and only BEFORE any file has been loaded into the EMUL51. Observe that symbol handling when doing disassembly and updating the CODE window will be much slower when this option is used. It is possible to disable searching for specific symbols to speed up the search:

DIS LABEL disables search for CODE symbols.
DIS SYMBOL disables search all symbols.
ENA LABEL enables search for LABELS only.

examples

```
HEA          ;display which "HEAP" memory is being used
HEA = DISK   ;use a diskfile to store symbols
HEA = EXT    ;use extended memory to store symbols
HEA = EMS    ;use expanded memory to store symbols
```

HELP

The HELP Command.

This command takes you back up to the top menu and "pulls down" the HELP menu. Choose any item and press RETURN.

IB

Internal Break on pattern

format

IB <byte1>, <byte2 or x>, <byte3 or x>

(where *<byte1>* should normally be an instruction code and *x* means "don't care")

description

This command will look for the specified pattern throughout the entire code memory and progam a breakpoint wherever a match is found. Breakpoints will only be executed if the corresponding address is executed as the first byte in the instruction. (The emulator-generated Valid Fetch signal takes care of this.) Also see the CLB , BB, and BYB commands. Further details are available under BREAKPOINTS in Chapter 1.

IF

IF conditional

format

```
IF boolean-expression [THEN] <cr>
        [command <cr>]
[ORIF boolean-expression <cr>]
        [command <cr>]
[ELSE <cr>]
        [command <cr>]
```

description

You can use this command to set up the conditions to be met in order for a block of commands to be executed.

IF begins a block of commands to be executed when the IF condition is TRUE.

BOOLEAN EXPRESSION is any of the forms of expression. The result is TRUE when the least significant bit of the result is 1; otherwise it is FALSE. When the boolean expression after IF or ORIF is TRUE, the commands in that block are executed, to the exclusion of all other blocks in that IF command.

THEN is an optional addition to the IF command.

<cr> is an intermediate carriage return that ends each input line of the IF command. It does not terminate the entry of the command.

COMMAND is any EMUL51-PC command except DEFINE Macro and REMOVE Macro.

ORIF introduces a secondary block of commands. When the condition after the IF is FALSE, the first ORIF condition found to be TRUE executes the ORIF block, to the exclusion of all other command blocks.

ELSE introduces a block of commands to be executed when no IF or ORIF condition is TRUE.

ENDIF terminates the entire IF command.

examples

```
*IF .VAL EQ 200H
*.GO FROM .ST TILL .END
*ORIF .VAL EQ 100H
*.GO FORE
*ELSE
*.WRITE 'DONE'
*ENDIF
```

INC Include

abbreviation INC

format INCLUDE [drive] filename

description This command reads a sequence of emulator commands from the specified drive and file, and is used to load macro definitions from disk.

examples

```
INCLUDE GO.MAC          ;load the macro GO.MAC from the
                        default drive.
```

INT

Interrupt

abbreviation **INT**

format INTERRUPT

description For processors except Siemens 80535: This command displays the Interrupt level 0 (IIP0), Interrupt level 1 (IIP1), Interrupt Enable register (IE), and Interrupt Priority register (IP). IIP0 and IIP1 are displayed as YES or NO, to indicate whether an interrupt on that level is active or not.

To obtain information on the cause of interrupt, compare the program counter (PC) with the code listing. If an emulation session is started, and the PC was changed since the last session without resetting all interrupts, a warning will be issued: "WARNING ACTIVE INTERRUPTS". To reset the interrupt in progress registers internal to the microprocessor, one of the following actions must take place:

A RETI is executed.
A RESET CHIP command is issued.
The Reset push button on the pod is pressed.
A reset from the target.

Siemens 80535: Command works essentially as above, except that IEN0, IEN1, IP0, and IP1 are displayed. Four levels of Interrupt In Progress are indicated as YES or NO.

J

See SN (Step Next)

LC

Loop Counter

format

```
LC
LC = decimal value
```

description This command displays or sets the Loop Counter and is used in some trace conditions. (See TR command).

examples

```
LC      ;display the loop counter value.
LC=2000 ;sets the loop counter to 2000 decimal.
```

LIN **Linestep**

abbreviation L

format L [FROM <address>]

description This command goes through the symbol table and locates all symbols with the type LINE, and sets a breakpoint on the corresponding addresses. Then the code will be executed until a breakpoint is encountered. Pressing the ESC key will also break execution. This is usable only when working with C51(Archimedes / IAR, BSO/Tasking, Franklin/Keil or Intermetrics/Whitesmiths/COSMIC) and PL/M-51 (Intel or BSO/Tasking). Using the GO or SN commands will erase all LINE number breakpoints, but the STEP command will not. The L command erases all previously activated breakpoints in the breakpoint RAM before it is executed.

LIS **List**

format

```
LIST filename
        [STOP]
```

description With this command a copy of everything printed on the "scroll" screen except trace displays, (see TDL, TDFL, etc.), will be sent to the specified file. Substitution of the word *STOP* instead of a file name will close the file. Choosing a new file name will cause old file to be closed and the new file to be opened.

examples

```
LIST LPT1          ;list on parallel printer.  May be
                   buffered.  Flushed at LIST STOP.
LIST
EMSESS.EMU         ;save on file
LIST A:
EMUDATA.DAT        ;save on file
LIST STOP          ;close existing file
```

LOA Load

abbreviation LOA

format

```
LOAD [d:]
filename        [NOCODE]
                [NOSYMBOLS]
                [RELMS]
                [?M <modulname>]
```

description

With this command you can load the user program and symbol table from the specified drive and file name into code memory on the emulator. Code goes to emulator memory regardless of the MAPC setting. Code is loaded without any readback check. If the file contains a symbol table, it will be loaded without a check for duplicate symbols. This means that multiple loads of the same file will result in multiple versions of the same user symbols. You can avoid having multiple copies of the symbol table by using a REM SYM command before loading the file. If the symbol table contains references to system symbols, they will not be loaded.

NOCODE suppresses loading of the code from the file, but will load the symbol table if present.

NOSYMBOLS suppresses loading of the symbol table from the file, but will load the code if present.

RELMS loads a RELMS obj file, (8080 object format).

?M <modulname> loads symbols only from the specified module and may be used if you run out of memory with big symbol files. This command may be repeated any number of times for different modules to load only the modules currently needed for debugging. If you need to load many modules but not all, you should set up a macro for loading. This works only for INTEL OBJ (OMF) formats. (See example below).

MOTO Used when loading Motorola "S1" files with extended "SA" symbol records.

The following file formats are supported:

```
INTEL HEX       ;Only code
INTEL OBJ
(OMF, AOMF)     ;Code, modules, typed symbols.  Object
```

	Module Format, Archimedes Object Module Format.
AVOCET SYM	;Code is same as Intel HEX. Separate symbol file, typed symbols
IAR SYMBOLIC	;Code and untyped symbols
HMI SYM	;Intel HEX. Separate symbol file, Untyped symbols.
INTEL SYM	;Can appear together with INTEL HEX
RELMS OBJ	;Equivalent with Intel's 8080 obj format, Untyped symbols
AMERICAN AUTOMATION	;Separate symbol file, untyped symbols
2500 AD	;Separate symbol file, untyped symbols. Microtek symbols supported.
ObjectExtended	;For Franklin/Keil
UBROFF	;Universal Binary Relocatable Object File Format (IAR/Archimedes, others).
IEEE 695	;BSO/Tasking
COSMIC	;Intermetrics/Whitesmiths
Miscellaneous	;various other assemblers

examples

```
LOAD TEST.A03       ;load an Archimedes file
LOA COUNT.OBJ
NOCODE              ;load only symbols from COUNT.OBJ
LOA TME
NOSYMBOLS           ;load only code from TME
LOA SDER.OBJ
?M VIR              ;load code and symbols from SDER.OBJ, but symbols
                    only from the module VIR
DEF :LOAD           ;macro to load symbols from three modules out of
                    PLM.OBJ
 LOA PLM.OBJ ?M MAIN
 LOA PLM.OBJ ?M TINT
 LOA PLM.OBJ ?M FINISH
EM
```

To load code to writable target code memory, first load code to the emulator. Then use a command such as XBYTE 0 = CBYTE 0 TO 1FFFF. See CBYTE.

For loading macros, see INC.
For loading binary, see LOB.
For loading PPA setups, see LOP.
For loading window setups, see LOW.
For loading trace setups, see TSGET.

LOB

Load Binary

format

LOB <[d:]filename> <start address in hex>

description

This command loads the given user program in raw binary form from the specified drive and file name into code memory on the emulator.

examples

```
LOB CHK.BIN 1000   ;Load the binary file CHK.BIN to emulator code memory.
                    Load to address 1000H and up.
```

LOP

LOad Ppa setup command

format

LOP <[d:]filename>

description

Load PPA setup data previously saved using the SAP command.

example

```
LOP PPA.SAV          ;Load data from PPA.SAV to PPA setup window.
```

LOVER

Line OVER

format

LOVer

description

This command steps to the next source line without taking CALL instructions. It generates a Go to the next source line (line number reference).

LOW

Load Window Setup

format

LOW [d:] filename

description

Loads a window setup file which was previously saved using the SAW command.

examples

```
LOW WIN.SUP     ;loads win.sup from current directory.
LOW A:XDAT.SUP ;loads XDAT.SUP from the a: drive.
```

MAC

Macro

abbreviation MAC

format MACRO [:macroname]

description Displays the definition of a macro defined by *:macroname.* If MACRO is used without a name specification, all macros will be displayed.

:macroname must be defined previously. See DEFine command. See Macros in Chapter 1.

examples

```
MACRO           ;display all macros
MAC :EXEC       ;display the macro :EXEC
```

MACDELAY

(Simulator option only)

Execution of a macro after a time delay.

abbreviation MACD

format MACDELAY count [msec | sec] :macro [operand ..]

description This is a simulator command which is intended to be combined with a macro in order to implement time delays. The specified macro will be executed after the specified time or cycles.

Arguments:

count [msec | sec]

Time delay. The default time unit is 'machine cycles', but can be altered to milliseconds or seconds.

:macro [operand ..]

The macro to be scheduled for later execution. The optional arguments are frozen at the time of the 'MACDELAY' command.

examples

```
MACDELAY 100 msec :startmotor
```

```
define :event_P1
macdelay 850 :printer_ready %0 %1 %2  ; inherit
operands
em
```

MAP

Map

format

```
MAPX
MAPC
MAPX = E [T] E [T] .....
MAPC = E [T] E [T] .....
MAPC = USER
MAPX = USER
MAPC = EMUL
MAPX = EMUL
```

description

MAPX and MAPC display mapping of external memory (X)and code memory (C). The maps are set in 16 portions of 4K bytes each. Each 4K segment may be mapped either to the emulator (E) (default for CODE), or to the target system (T) (default for XDATA). Start with the lowest 4K (0000H to 0FFFH) section and end with the highest 4k (F000H - FFFFH) section.

MAPC = USER maps all code memory to the target PROM.

MAPX = USER maps all external data memory to the target.
MAPC = EMUL maps all code memory to the emulator.
MAPX = EMUL maps all external data memory to the emulator.

examples

```
MAPX                ;displays mapping of external memory
MAPC = E E T T T T T
T T T T T T T T T   ;maps first 8K of code memory to the emulator and the
                    remaining 56K to the target system.
MAPX = USER         ;maps all xdata memory to the target.
MAPC = EMUL         ;maps all code memory to the emulator.
```

MEM

MEM Command

description

Will report how much memory is available for symbols, SYSTEM, SHELL, etc. If you seem to have too little memory left you may be able to use the HEAP command to take advantage of EXT or EMS memory. You may also be able to remove resident programs or reduce "buffers" in CONFIG.SYS.

MOD

MODULE Command

description This command shows the current module.

N

See SN (Step Next)

NOHIT

NOHIT Command

format NOHIT [0] [1]

description This command is used in connection with the SYSTEM command. NOHIT 1 will suppress the "Hit any key to continue.." message when returning to the emulator after a SYSTEM command. NOHIT 0 will return to normal operation where you must press a key before returning to the emulator.

NOS

Nosnow

format NOSNOW

description Use this command if you get "snow" on the screen. The window update will then be slightly slower, because the output is only done when the video is "retracing". There is no command to go back to "snow."

examples

```
NOSNOW              ;prevents snow on screen
```

OST

Open (Close) Step Trace file

format OST
CST

```
OST
CST
```

description This command will open a file "STEPTRAC.TRC" in the current directory, where it will store trace data from each single step. Data from STEP will be stored in this file in a circular fashion, up to a maximum 1000 instructions. Then the oldest frame will be overwritten. While this file is open, the TDS and TDFS commands can be used to view the contents of the trace file. The CST command is used to close the file. After the CST command is used, the file is lost and can not be viewed.

PBYTE

Put Byte See CB

PC

Display or Set PC

format

PC = [expression]

description

This command allows you to display or set the 16-bit Program Counter.

EXPRESSION can be a number, reference, or formula that equates to the number expressed.

examples

```
PC                  ;Display program counter
PC = .BEGIN + 7     ;Set PC to new address
```

PPA

Program Performance Analysis

format

PPA

description

This command works only if you have the trace board.

Executing the command takes you into the PPA window, where it is possible to monitor where the program spends its time. This is done by using the trace board to sample where the program is. The PPA program updates one counter for each programmed "bin." The result can then be displayed dynamically, either in numbers and percentages or with bar graphs. Usage is self explanatory; a short description is given below.

Commands are activated by typing the upper case letter associated with each command. You can also move the cursor to the desired command, then press the return key.

Edit takes you into the edit mode. Use the arrow keys to move to the desired positions to enable or disable any of the "bins" or ranges. Then program the "bin" itself, by putting in either a single address or a range of addresses.
Examples:

```
1000
320 TO 330
200 L 10
```

Run/Hold If Run is visible and activated, the program and the PPA will begin from where the PC is. If Hold is visible and activated, the PPA will stop until Run is activated again. (The program will not stop, however.)

Bargraph On / Bargraph off toggles between showing numbers and bar graphs.

Clear count resets all "bin" counters and immediately starts counting again.

ESC breaks program execution if it was not active when PPA was entered, and exits PPA.

PUT

Put

format

PUT [drive:]filename :macro-name, :macro-name ......MACRO

description

This command stores macro definitions in a DOS file.

drive:filename designates the drive and filename of the file that will receive the macro definitions. If the file already exists on that drive, the old file will be overwritten. If *drive:* is omitted, the default drive is assumed.

:macro-name is the name of a macro preceded by a colon.

MACRO stores all macro definitions currently in the macro table.

examples

```
*PUT C:SAVMAC :TSTLOOP :G
*PUT MACRO
```

QRA

TRACE QUALIFIER register commands (The new TS command is recommended)

format

QRAx
QRBx
QRAx = [ADR spec1] & [DAT spec2] & [P1 spec3] & [P3 spec3]
QRBx = [ADR spec1] & [DAT spec2] & [P1 spec3] & [P3 spec3]

x is 0 - 9

spec1 is address
address TO address
address LENGTH number
wildcard (hex)
wildcard (binary)
spec2 is VF + WR + RD + SY1 + SY0 + I2 + I1 + I0 + 8 bit DATA can be specified as in spec1 above. Normally only wild cards would be used.
spec3 is Port 1. 8 bits can be specified as in spec1 above but normally only wild cards would be used.

VF = Valid Fetch ;active high
WR = WRite to
external memory ;active high (not low!!!)
RD = ReaD from
external memory ;active high (not low!!!)
SY0 = external signal from pod
SY1 = external signal from pod
I2, I1, I0 indicates
interrupt level
from 0 to 7. ;I2 most significant

The QRA and QRB registers are used to set up trig conditions together with the TR register. It can also be used as trace conditions with the TR register.

The A and B in QRAx and QRBx stand for A-condition and B- condition respectively.

The four fields that make up a condition (ADR, DAT, P1, and P3) are ANDed together, meaning that all four fields have to match to produce a true condition. The fields are, however, independent of each other. If you for example set up QRA0 to trace all WRITEs to address 4000H and QRA1 to trace all READs from address 2000H, you will also trace all WRITEs to 2000H and all READs from address 4000H.

If any of the four components of the QR register are left out when QRAx (QRBx) is set up, it will default to all wild cards.

For an overview of how the trace works see the TRA paragraph.

examples

QRA0	;display QRA0
QRA1 = A 0 TO 7FF & D xx80	;set up QRA1 so that it will match frames only with addresses between 0H and 7FFH when data is 80H. VF, WR, RD, SY1, SY0, I2, I1, I0, Port 1 and Port 3 will all be set to wild cards (don't cares).
QRB9 = P1 12	;set up QRB9 to match frames only when Port 1 is 12H
QRB7 = D 00XXXXXX10000000Y	;matches only when 80H is read from external memory. Observe the "Y" indicating binary.

QUIET

QUIET Command

format

QUIET

description

This command is used to supress printing registers etc. to the screen after breakpoints or single steps. Instead of printing several lines only a dot, '.', is printed. Normally used to speed up executions of MACROs. For this to make a difference the CODE WINDOW should be off. Each time the command is used it toggles supression ON or OFF.

RBIT

Register BIt

abbreviation

RBI

format

RBIT partition [= boolean expr., boolean expr.], ..

description

This command displays or sets the contents of one or more addresses in the bit addressable memory.

PARTITION is a single address, or a range of addresses expressed as "address TO address" or "address LENGTH address."

BOOLEAN EXPRESSION is a number, reference, or expression. The result is truncated to its least significant bit. Also see CB.

examples

RBIT .CY	;display carry bit
RBI .ACC+4	;displays bit 4 in the accumulator
RBI .P1+2	;displays bit 2 in Port 1
RBI 0 to 7F = 1	;set all bit locations in RAM to 1

RBS

Bank Select

format

RBS [= expression]

description

This command displays or sets the register bank select (bits 3 and 4 of PSW). RBS governs the location of working registers R0 through R7 in internal data memory.

EXPRESSION is a number, reference, or formula that equates to the number expressed. For RBS the result must be 0, 1, 2, or 3. Otherwise an error will result.

examples

```
RBS     ;displays current bank number
RBS=3   ;Set RBS to 3
```

RBYTE

Register Byte See CB

RCS

RCS Command. Relative Computer Speed

description

RCS will print a number indicating the speed of your computer. Here are some numbers:

4.7 MHz XT	268
10 MHz XT	531
20 MHz '286	2302

If you run a fast machine and your crystal frequenty is low, you may want to use the -d option when you invoke the emulator. (See the INSTALLATION Chapter in the manual.)

REG

Display Registers

abbreviation

R, REG

format

REGISTERS

description

This command displays contents of the program counter (PC), accumulator (ACC), multiplication register (B), stack pointer (SP), data pointer (DPTR), working registers R0 and R1 in the bank currently selected, and program status register (PSW). PSW is displayed in binary radix.

Register names are displayed instead of "PSW."

Abbreviations have the following meaning:
C = Carry flag (CY)
A = Auxiliary carry flag (AC)
F = Flag 0 (F0)
RS = Register bank select control bits (RS1, RS0)
O = Overflow flag (OV)
- = reserved
P = Parity flag (P)

The other registers are displayed in hexadecimal.

After the registers, the next instruction is displayed in disassembled form.

REM Remove

abbreviation **REM**

format

```
REMOVE SYMBOLS
    ..module-name.symbol-name
    MACROS
    :macro-name
```

description This command will delete user defined symbols or macros from the symbol table or macro definition table.

SYMBOLS removes all user defined symbols. See the SYM paragraph.

..module-name is the name of the module containing the symbol to be removed.

.symbol-name is the name of a user defined symbol preceded by a period. If two or more symbols exist with the same name, only the first one is removed.

MACROS removes all macro definitions.

:macro-name is the name of a macro preceded by a colon (:).

examples

```
REMOVE SYMBOLS ;remove all modules, user symbols, and line numbers.
REMOVE .VALUE  ;remove the first occurrence of the user symbol .VALUE.
REM ..ETON4.LOOP ;remove the user symbol from one module.
REM MACROS     ;remove macros.
REM :REPCYC    ;remove a macro.
```

REP Repeat

abbreviation REP

format

```
REPEAT <cr>
   [command <cr>]
   [WHILE boolean-expression] <cr>]
   [UNTIL boolean expression <cr>]
ENDREPEAT
```

description This command introduces a block of commands to be repeated.

<cr> is an intermediate carriage return. It ends each input line but does not terminate the command entry. The carriage return after the END keyword causes termination of the command.

COMMAND is any EMUL51-PC command except DEFINE macro and REMOVE macro.

WHILE introduces a halt condition for the command block. The loop halts when the WHILE condition is tested and found to be FALSE.

UNTIL introduces a halt condition for the command block. The loop halts when the UNTIL condition is tested and found to be TRUE.

BOOLEAN EXPRESSION is any number, reference, or expression. The result is TRUE when the least significant bit of the result is 1; otherwise it is FALSE.

ENDREPEAT terminates entry of the command block. It may be abbreviated to END.

To end a REPEAT command before a condition ends it, merely press the ESC key. It may take a couple of loops before the break occurs.

examples

```
*REP
.*STEP
.*REG
.*UNTIL PC EQ .DONE
.*ENDR
```

RES Reset

abbreviation RES

format

```
RESET  BRx
  BR
  EM
  CHIP
```

description This command restores its object to the initial status of EMUL51-PC.

BRx, (where *x* is a number from 0 through 9), resets breakpoint register x to no breakpoints.

BR resets breakpoint registers 0 and 1 to no breakpoints.

EM resets the microprocessor and all registers to their initial state. This is like restarting the EMUL51-PC.

CHIP resets the microprocessor to its initial state. The same effect is obtained if the reset push button is pressed, (if jumper JB1 is closed).

examples

```
RES EM          ;resets hardware and all registers
RES CHI         ;resets only hardware
```

R0 Display and set working register R0 - R7

format

Rn = [expression]
(where *n* = 0, 1, 2, 3, 4 ,5 ,6, or 7)

description You can use this command to display or set contents of working registers in the currently selected register bank. (See the RBS command.)

EXPRESSION is a number, reference, or formula that equates to the number expressed. For R0 - R7, the result is truncated to the low byte.

examples

```
R0
R4=20H
```

SAP

SAve Ppa setup command.

format

SAP <[d:]filename>

description

Save PPA setup window to a file. Use LOP command to reload.

examples

```
SAP PPA.SAV          ;Save the PPA window setup to PPA.SAV
```

SAV

Save

abbreviation

SAV

format

```
SAVE [d:]
filename        [NOCODE]
                [NOSYMBOLS]
                [HEXCODE] [partition]
                [partition]
```

description

This command copies the user program and symbol table to the specified drive and filename. If NOCODE or "partition" are NOT used, the partition of code affected by the most recent LOAD or SAVE command will be saved. At start up the default partition is set to 0. This means that no code will be saved if a SAVE (without partition) is used before LOAD.

NOCODE means no code will be saved, but the symbol table will be saved if present.

NOSYMBOLS means no symbols will be saved, but code from the default partition will be saved.

HEXCODE means the code will be saved in "INTEL HEX" format. This format is used by many PROM programmers. Partition may be used. No symbol table is saved.

examples

```
SAV A:MYPROG            ;save current program to a: drive.
SAV A:MYPROG NOCODE     ;save only symbols.
SAV MYPROG
NOSYMBOLS               ;save only code.
SAV MYPROG 0 TO FFF0    ;save code from 0 to FFF0H.
SAV MYPROG HEXCODE
0 TO 1000               ;save code from 0 to 1000H in Intel hex format.
```

For saving keyboard-console transactions, see LIST.
For saving macros, see PUT.
For saving PPA setups, see SAP.

For saving window setups, see SAW.
For saving trace setups, see TSPUT.

SAW

Save Window Setup

format

SAW [d:] filename

description

This command saves current window setup to the specified file. Use LOW to read file back into the emulator. This makes it possible to store any number of setups and switch between them using the LOW command. For more details, see Windows in Chapter 1.

examples

```
SAW WIN.SUP        ;save window setup to WIN.SUP in the current directory
SAW A:XDAT.SUP     ;save window setup to XDAT.SUP on drive a:
```

SCOPE

SCOPE Command

format

SCOPE [module]

description

This command displays the scope of the code that was loaded for each module. If the code is in a bank as defined by BANKSEG it is indicated.

examples

```
SCOPE   ;display code scope of all modules
SCOPE   ;display code scope of module CMD_INT
```

SEB

SEt Breakpoints

format

SEB

description

This command is used to set the complete 64K bit breakpoint RAM. This is useful in conjunction with the ABR command, (Automatic Breakpoint Remove). For details about automatic breakpoints, see the ABR command; for more about regular breakpoints, see Breakpoints in Chapter 1.

examples

```
SEB     ;set the 64K bit breakpoint RAM to all 1's.
```

SEC

Display emulation cycles

format SECONDS

description This command displays the number of cycles, (which in the case of a 12 MHz crystal is equal to microseconds), that elapsed during emulation. The timer is reset when the PC is changed, either with a PC command or when the FROM is used in the GO command. RESET EM also resets it.

SERIAL

(Simulator option only) **SERIAL SIMULATOR** - Map serial port to data source/destination

abbreviation SER

format

```
SERIAL In   {COM1|COM2} baudrate
         KEY  [hotkey (hex)]
         FILE filename1
         MACRO :macroname1
         STR  string
                Out  {COM1|COM2} baudrate
         WIN  upper row  lower row
         FILE filename2
         MACRO :macroname2
SERIAL  OFF
SERIAL
```

description This is a simulator command. When the program under simulation accesses SBUF, the simulator will intercept the processing and take an 'action'. The following options are available:

COM1
COM2 use the COM-port of PC. Its baudrate should be set to match the external hardware device and does not need to correspond to that of the simulated CPU.

KEY read input from the PC's keyboard. An optional 'hotkey', given in hex, will stop the simulation (default Esc).

WIN Direct output to a window on the screen.

FILE read/write data to the specified file, one byte at a time.

MACRO execute a MACRO.

STR a string is used as input. The string which follows the STR cannot contain any blanks. STR and "string" should be separated by a blank. The string will be used over again after it has been exhausted. See example below.

The timing of RI and TI is determined by the simulated baudrate. The "input device" is only polled at intervals corresponding to a "character time". Then, if a character is available, RI will be set. The RI and TI status bits will be simulated according to the following rules:

RI:	if more data is available, the 'RI' bit will be set. Conditions:
COM	character in buffer
KEY	keyboard hit
FILE	not end-of-file
MACRO	RI set by macro
TI:	set after a delay corresponding to the baudrate and number of bits.

examples

```
SER I KEY O WIN 15    ;input is keyboard, output is screen window at starting
                      at row 15 of the screen.

SER I FILE A:SERDATA
O FILE C:SEROUT       ;input is file a:serdata and output is c:serout.

SER I STR
HELLO_THERE
O WIN 15              ;input is the string HELLO_THERE (the string input
                      cannot have any blanks). The string will be used over
                      again when it has been exhausted.

SER I COM1
300 O FILE SAVE       ;COM1 300 baud is used as input.
```

SHELL

SHELL command

abbreviation **SH**

format SHELL [/d] [DOS command]

description SHELL swaps out the emulator program to EMS ro disk and loads a new DOS shell. Almost the entire memory will be available to run the new shell. The path to the shell must be defined by the environment variable COMSPEC. Typically you may put something like

```
SET COMSPEC=C:\DOS\COMMAND.COM
```

in your AUTOEXEC.BAT file.

EMS has priority, but disk can be forced as swapping media with the optional switch /D. Note that EMS is by an order of magnitude faster!

A full DOS command line can be given and executed with automatic return to the emulator. If the DOS command is left out, the SHELL will prompt you for a DOS command. The command "EXIT" brings you back to the emulator.

examples

SHELL ;Swaps the emulator to EMS, if available, otherwise to disk. Then it puts you in comand under the DOS coimmand processor. Terminate and return to the emulator with the command: EXIT

SHELL /D edit
myfile.c ;Swaps the emulator to disk (leaves the EMS available for the editor), then lets you edit the specified file. When you exit from 'edit', you will come directly back to the emulator.

SL

SL Command

description This command will set breakpoints on all line numbers. It can then be used with the GB command. Normally, you would simply use the L command which automatically sets breakpoints on all line numbers and starts emulation.

SN

Step Next

format SN [FROM <address>]
J
N

description The SN command is "single straight step." The regular Step command will always execute the next instruction, which means that it will go through all subroutine calls and interrupt service routines. The SN command, on the contrary, will skip over CALLs and interrupts. This means, of course, that if you don't come back after a subroutine or an interrupt, the program will continue to execute forever, unless you hit the ESC key or encounter another breakpoint.

Do not use this command when the Code Window is on.

The SN command uses it own "disassembly pointer" to calculate where the next instruction is located. This means that in order for this "disassembly pointer" to be initialized, you must first use SN FROM <address>. SN F PC will make it start from the present location. Thereafter SN with no parameters should be used.

SN will follow JUMP instructions and branch instructions. It will also step out of subroutines and interrupt service routines, but will not go into subroutines or interrupt service routines. After breakpoints and regular single steps, the next instruction in line to be executed will be shown together with some register values. In the "SN mode", however, the next instruction in "straight line" will be shown. Observe that this instruction may or may not be the next instruction in turn to be executed. An indefinite number of instructions in a subroutine or in an interrupt service routine may be executed before it. It will, however, be the last instruction to be executed before a break is taken. Also note that SN uses the BR0 breakpoint register. Any previous breakpoints held in this register will be destroyed.

N and J are used only to skip over LCALLs and ACALLs. No "disassembly pointer" is used with these two commands. Instead the PC (Program Counter) is used. If PC points to a LCALL or a ACALL, a breakpoint will be generated on the instruction following the CALL. Then with the "N" command a GO SLOW (GS) will be executed. The break will then be taken on the instruction following the CALL. The "J" command will instead execute a "GO" and consequently run through the subroutine at full speed. "J" will work as regular breakpoints, which means that the instruction following the CALL will be executed before the break is taken.

It is important to realize that J will execute the instruction following the CALL before the break is taken. If, for example, you had two LCALLs following each other, the first one could be skipped over using J. But the next LCALL instruction would be executed, with the result that you would wind up in the second sub-routine without being able to step over it. This would not happen with the SN and N commands.

examples

SN F 1000	;Set "disassembly pointer" at 1000 and step straight to next instruction.
SN	;Step straight to the next instruction starting from the "disassembly pointer".
J	;Step straight starting from the "PC" at full speed.
N	;Step straight starting from the "PC" at GO SLOW speed.

SNAP

SNAP command

format

SNAP <mode>, [<type>, <address>]

description

The SNAP command changes the use of breakpoints. Rather than actually breaking it uses the break only to update the windows (mode 1 and 2) or displaying the specified address (mode 3 and 4) before execution is resumed. Once SNAP is activated (mode 1 and 3) the only way to stop execution is to do: "SNAP 0". Otherwise each attempt to break only will result in updating windows or specified address. Mode 2 and 4 will automatically return to mode 0 after the first breakpoint was encountered and used to update.

MODE

0 Disable SNAP
1 Enable SNAP
2 Enable "one shot SNAP"
3 Enable "show one address only SNAP"
4 Enable "show one address only one shot SNAP".

TYPE (only used in modes 3 and 4 above)

D (internal data)
R (Special Function Register SFR)
X (External Data)

ADDRESS address of desired "SNAP address". (only used in modes 3 and 4 above)

examples

```
SNAP 1           ;Use breakpoints to update windows
SNAP 2           ;One shot update
SNAP 3 D 20      ;Display data address 20 at each breakpoint
SNAP 3 X 3000    ;Display external data address at each breakpoint
SNAP 4 R 80      ;Display SFR at address 80 one time
SNAP 4 R .SBUF   ;Display SBUF one shot.
```

SO

SOurce level debug

format

SO [=<type>,<path>,<extension>]

description

Displays or sets parameters used by the source level debugger.

TYPE Type of source file.

S stands for a regular source file where line numbers in the corresponding object file refer to the actual line numbers in the source file. (Unlike Intel's PL/M-51).

P stands for an Intel PL/M-51 file where line numbers in the object file correspond to information not in the source file, but in the list file.

PATH By default the debugger will look for the source file (or list file) in the current directory. Setting <path> to point to any other directory will cause the debugger to look there for the file.

EXTENSION By default the extension the debugger looks for is .C.

examples

```
SO                     ;display current parameters
SO=S,\PROJ\FILES\,SRC  ;regular source, source files are in \PROJ\FILES\
                       extension is .SRC
SO=P,,LST              ;Intel PL/M-51, current directory, extension .LST
```

SST

Single Step Trace

abbreviation

SST

format

SST ON (default)
SST OFF

description

When emulation starts (GO, STEP, GB etc.) the trace buffer is reset and all old information is lost. If SSE OFF is used, a STEP will not affect the trace buffer. Instead the trace buffer is left intact. Only the STEP comand is affected. GO, GB etc. will still reset the old trace buffer. The instructions executed by the STEP will NOT be appended to the trace buffer.

STA — Display contents of Stack

abbreviation STA

format STACK [expression]

description This command displays the contents of the EMUL51-PC processor stack area and stack pointer value.

EXPRESSION indicates the number of bytes at the top of the stack to be displayed. If the result exceeds the value of the stack pointer plus one (SP + 1), the display terminates at data location 0. If the expression is omitted, the display starts at the current stack top and stops at data location 0.

examples

```
*STACK 6        ;displays 6 bytes of the stack.
*STA            ;displays the stack pointer value and the contents of the
                 stack area.
```

STE — Step

abbreviation S

format STEP [FROM address]
STEP <number>

description This command executes one instruction, then stops and displays registers, and shows the next instruction in disassembled form.

FROM sets the PC to "address." If the FROM is omitted, the current PC is used.

<number> will single step specified number of times. ESC will abort stepping.

examples

```
STEP            ;step one step from current PC.
S F .LOOP       ;steps one step starting at .LOOP.
S 10            ;make 10 single steps.
```

SUF

Suffix

abbreviation SUF

format

```
SUFFIX =        [Y]
                [Q]
                [T]
                [H]
```

description This command displays or sets the default radix for numbers entered at the console.

Y indicates a Binary radix, *Q* indicates Octal, *T* indicates Decimal, and *H* indicates Hexadecimal. Hexadecimal is the initial default radix. See Evaluate.

examples

```
*SUFFIX          ;displays current radix.
*SUF = Y         ;sets to binary radix.
*SUF = H         ;sets to hexadecimal.
```

SWD

Setup Window Data address

format SWD <data address>

(where data address is in the range from 0 to E7H)

description This command sets up the first address to be displayed in the DATA window. The contents of this address and of the next 23 addresses will be automatically displayed in the DATA window. (For more details, see Static Windows in Chapter 1.)

examples

```
SWD 40  ;first address to be displayed will be 40H
SWD 80  ;only relevant in 8052 type CPU's.
```

SWR

Setup Register Window

format

```
SWR
SWR <position> <type> <symbol> <address>
```

description This command makes it possible to set up any of the 18 positions in the "REGISTER" window, to reflect any address in any of the different address spaces of the microprocessor. Using the SWR command without any parameters will show the setup. (For more details, see Static Windows in Chapter 1.)

<position> is any number betweeen 1 and 18 that designates the physical placement on the screen, as follows:

```
1.  4.  7.  10. 13. 16.
2.  5.  8.  11. 14. 17.
3.  6.  9.  12. 15. 18.
```

<type> is DB, RB, RBI, XB, CB or PB (or any equivalent notation).

<symbol> is any ASCII string having a maximum of 8 characters.

<address> is any address in the range from 0 to FFFFH.

examples

```
SWR             ;display current setup
SWR 10 XB
X4000 4000      ;make position 10 on the screen reflect the contents of
                external address 4000H.  The symbol X4000 is chosen to
                indicate eXternal memory 4000H, but the symbol may be
                any ASCII string with less than 9 characters.
SWR 2 RB
IRCON C0        ;Interrupt Request Control Register (used in Siemens
                80535), displayed in position 2.  RB indicates Register
                Byte.
```

SWX

Setup Window external data address

format

SWX <external data address>
(where external data address is in the range from 0 to FFE7H)

description

This command sets up the first external address to be displayed in the XDATA window. The contents of this address and of the next 23 addresses will be automatically displayed in the XDATA window. (For more details, see Static Windows in Chapter 1.)

examples

```
SWX 40          ;first address to be displayed will be 40H
SWX 1000        ;first address to be displayed is 1000H
```

SY0

Display and set SY0A register

format

SY0A
SY0A = H [L]

description

This command displays and sets the SY0A register, and determines whether SY0 will be active high (default) or active low.

examples

```
SY0A          ;displays SY0A
SY0A = L      ;sets SY0A to active low
SY0A = H      ;sets SY0A to active high.
```

SYM

Symbols

abbreviation

SYM

format

SYMBOLS [module name | ?] [symbol name | ?]
[data type | ?]
(where | means "or")

description

This command displays all user defined symbols and their corresponding values for all the modules (or as specified otherwise).

examples

```
SYM                ;displays all symbols.
SYM ESIN ? ?       ;displays all symbols in module ESIN
SYM ? SEW BIT      ;displays all BIT symbols with the name SEW
SYM ? ? CODESEG    ;displays all code segements loaded
SYM ? ? DATA       ;displays all DATA symbols
SYM ? ? MOD_BEG    ;displays all module names
SYM ? ? MOD        ;displays all MOD_BEG and MOD_END.  Useful to see
                            where modules begin and end!
SYM ? ? ? 2014     ;displays all symbols with value 2014 (hex)
SYM | more         ;stop listing after 5 lines.  Resume with keystroke.  Line
                   numbers (LINE) will not count in the "5 lines".
```

SYMLOAD

Load symbols from a converted assembly listing

abbreviation

SYML

format

SYMLOAD <filename>

description

<filename> is a file created by a utility program that extracts line number information from a list file created by an 8051 cross-assembler. The format of the contents of this file should be:

<filename.ext> <linenumber in decimal> <address in hex>.

example

```
51demo.lst 43 0000
51demo.lst 46 000B
51demo.lst 47 000E
51demo.lst 48 0010
```

SYMLOAD should be used in conjunction with the loadable file created by the cross assembler.

Nohau provides a utility, CONVERT.EXE, which works on some assembly list files. There is no guarantee that it will work on your assembly list file. Therefore we provide CONVERT.C which is the source file of CONVERT.EXE so that you can modify this to work on you list file. Below is a description of how CONVERT works:

CONVERT <list file name> <output line number file name>

E.g., CONVERT 51DEMO.LST 51DEMO.LSY

Restrictions are that the list files must have absolute addresses (linking is not necessary to resolve absolute addresses). Also 'org' is a keyword that must be followed by hex address.

This was tried on two assembler list files. They both had format: <hexaddress> <opcode> <linenumber> <source statement>. Changes will need to be made for other assemblers.

This is a typical sequence of commands that would be used in the emulator to debug in "assembly source":

```
SO = S,\FILES\ASSY\,ASM    ;see SO command for details
LOAD PROG.OBJ              ;Load object file created by
                           the assembler
SYML PROG.LSY              ;PROG.LSY contains line
                           symbols as described above
CW                         ;Turn on the CODE WINDOW
```

Now you should see a mix of the source code from

PROG.ASM and disassembly of the code from the emulator. To view only source press function key F2. F2 toggles between mixed mode and source mode. You may use the L or the S commands to step in your code. Use CTRL+R to see registers. Also see Source Level Debugging for more details.

SYS System

abbreviation SYS

format SYSTEM [DOS command]

description This command executes the specified DOS command and returns to the emulator. Enough memory must be available for this to work. Also COMMAND.COM must be available on the drive from which you loaded DOS.

examples

SYS AEDIT MYPROG.SRC	;executes AEDIT to edit the file MYPROG.SRC When you exit from AEDIT you return to EMUL51-PC.
SYS DIR \BIN*.OBJ	;Makes a directory of all files with the extension OBJ in subdirectory BIN, then returns to EMUL51-PC.

TABS TAB Command

format TAB
TAB <number>

description This command displays or changes the number of blanks that replaces TABs from a source file. Useful when the CODE WINDOW is used with source level debugging. Default number is 8.

examples

TAB 4	;use 4 blanks for TAB expansion
TAB	;display TAB expansion

TB Trace Begin

format TB

description This command begins a new trace; it can only be used when the EMUL51-PC is in emulation mode.

TBR

Trace Breakpoint

format

```
TBR =  [ON]
       [OFF]
```

description

This command displays, sets or resets the TBR register. If TBR = ON, a trig condition on the trace will unconditionally break emulation. Observe that this command, together with appropriate setting of the QR's, can be used to set new breakpoints even after the emulation has begun. (For details about how to set up trace conditions, see Trace in Chapter1. For more about how the TBR breakpoint works, see Breakpoints in Chapter 1.)

examples

TBR	;display status of the TBR register
TBR = ON	;activate TBR
TBR = OFF	;reset TBR

TC

Display and set Trig Counter

format

TC [= decimal number]

description

This command displays or sets the Trig Counter register, (which by default is set in the middle of the trace buffer). The TC register is used to count the number of frames to be traced after the trace has encountered a trig condition. A low number gives you more information about what happened before the trig point, while a larger number gives you more information about what happened after the trig condition. (Usually done in the TS screen.)

examples

TC	;Display current setting of TC
TC = 0	;trace stops immediately after the trig condition was encountered.
TC = 2000	;2000 frames after trig will be collected. With the 4K deep trace buffer you will also have about 2000 frames collected before trig.
TC = 4000	;You will get about 90 frames before trig and 4000 frames after the trig, (with the 4K deep trace buffer).

TD Trace Display

format

TD [frame]
TDS [frame]
TDP [frame]
TDD [frame]

description

The TD command displays frames collected in the trace buffer, in disassembled form. Special text lines like *** PROCESSING INTERRUPT ***, or *** MAIN *** clarify what has been collected. If TD is used without a frame number, frames around the trig point will be displayed. If trace was interrupted, either by a break condition or a TE command, the last frames in the trace buffer will be displayed. The trig frame is always number 0. All frames before the trig frame are minus frames, (see example below). If you have used the trace filler function you must be careful when you use the TD command, because the disassembler will not recognize if some frames are missing. For more details, see Trace in Chapter 1.

The TDS command is like the TD command, except that it displays from the "file buffer" instead of from the "trace buffer." For more details, see the OST command.

The TDP command is the same as the TD command, but displays ports and external data addresses in addition to what appears in the regular TD display.

The TDD command is the same as TD, but displays DMA info ('152 and '452), external data addresses and data in addition to the regular TD display. For more details see "How the Trace Works" in Chapter 1.

examples

```
TD        ;display frames around the trig point
TD -4000  ;display starting with frame number -4000, (i.e., 4000 frames before
           the trig).
TD 319    ;display starting with frame number 319 after the trig point.
```

TDF Trace Display Frames

format

TDF [frame]
TDFL start [TO stop] [-filename]
TDFL start [LEN number] [-filename]
TDFS [frame]

description

TDF displays frames collected in the trace buffer, in frame form. If TDF is used without a frame number, frames around the trig point will be displayed. If trace was interrupted, either by a break condition or a TE command, the last frames in the trace buffer will be displayed. The trig frame is always number 0. All frames before the trig frame are minus frames. For more details, see Trace in Chapter 1.

TDFL works like TDF, except that TDFL outputs continuous data without stopping after each page. If the optional -filename is used, the output is written to the specified file only, and not to the screen. In this case, LIST should not be active.

TDFS is like TDF, except that TDFS displays from the "file buffer." For details see the OST command.

examples

TDFS	;display frames around the trig point
TDFS -4000	;display starting with frame number -4000, (i.e., 4000 frames before the trig)
TDF 319	;display starting with frame number 319 after the trig point.
TDFL -1000	;displays all frames starting at -1000
TDFL -500 TO -400	;displays all frames between -500 and -400
TDFL -100 -FRAME.SAV	;saves all frames from -100 to the file FRAME.SAV in the current directory.
TDFL -1000 TO 1000 -\TRACE\SAMPLE1	;saves all frames between -1000 and 1000 to the file SAMPLE1 in the sub-directory \TRACE

TDL Trace Display List

format

```
TDL frame [TO frame] [-filename]
    [LEN number] [-filename]
TDSL frame [TO frame] [-filename]
    [LEN number] [-filename]
TDPL frame [TO frame] [-filename]
    [LEN number] [-filename]
```

description

The TDL command works like TD, except that TDL outputs continuous data without stopping after each page. If the optional -filename is used, the output is written to the specified file only, and not to the screen. In this case, LIST should not be active.

TDSL works like TDL, except that TDSL displays data from the "trace file."

TDPL also works like TDL, except that TDPL shows information with ports and external data addresses.

examples

TDL -4000	;display starting with frame number -4000, (i.e., 4000 frames before the trig)
TDL 319	;display starting with frame number 319 after the trig point
TDL -1000	;displays all frames starting at -1000
TDL -500 TO -400	;displays all frames between -500 and -400
TDL -100 -FRAME.SAV	;saves all frames from -100 to the file FRAME.SAV in the current directory.
TDL -1000 TO 1000 -\TRACE\SAMPLE1	;saves all frames between -1000 and 1000 to the file SAMPLE1 in the sub-directory \TRACE

TE

Trace End

format TE

description This command ends the trace and can only be used when EMUL51-PC is in emulation mode.

TR

Trace Register

format

```
TR
TR = TRACE  ALL    & TRIG       NOTRIG     &ITRACALL
            ACOND      ACOND               INT
            BCOND      BCOND               MAIN
           A & B       ACOND THEN BCOND
            A LOOP
            A LOOP THEN B
```

description The TS command is recommended in place of this command.

TR displays or sets the Trace Register. Any combination of the TRACE, TRIG and ITRACE conditions may be set. The examples below provide a number of possibilities.

examples

```
TR                  ;displays Trace Register.
TR = TRA A & B      ;sets trace condition to ACOND and BCOND.  Both A
                    and B must be true to trace.
TR  = TRI A LOOP    ;sets the trace to trig when the A condition(s) has
                    appeared the number of times set in LC (Loop
                    Counter).
TR  =  ITR MAIN     ;Only traces instructions in the main program; does not
                    trace instructions executed within interrupts.
TR  = TRA ALL
& TRI A             ;traces all.  Trigger on A condition(s).  Any TRACE,
                    TRIG and ITRAC not mentioned in a setup, remains
                    untouched.
```

TS

Trace Setup

format

TS

description

This command takes you to a "static screen" where all trace functions can be programmed and altered. TS may replace the following commands: TR, TC, and LC. Use of the TS screen is explained in detail under Trace in Chapter 1.

TSG

Trace Setup Get

abbreviation

TSG

format

TSGET [drive]<filename>

description

This command loads a previously-saved trace setup window (which was saved with the TSPUT command).

examples

```
TSG COPTRIG   ;loads COPTRIG to the TS window
```

TSP

Trace Setup PUT

abbreviation

TSP

format

TSPUT [drive]<filename>

description

This command saves the TS (Trace Setup) window (which you load back with the TSGET command).

examples

```
TSP COPTRIG   ;saves contents of the TS window to the file COPTRIG.
```

VER

Version

format

VER

description

This command displays the software version number (which is also displayed when EMUL51 is invoked). The command also shows the revision of the "COM" EPROM on the emulator board, shows which symbol file is being used for the internal chip symbols, and shows which STR file is in use.

WAI

Wait

format

WAIT
WAIT <number>

description

The WAIT command outputs the text message "Press any key to continue" to the screen, then waits for a key stroke before continuing. This command could be useful to halt execution of REPEAT commands, to enable reading the results of emulation.

WAIT <number> forces a delay of <number> seconds before returning to the prompt.

examples

```
WAIT
WAIT 10   ;wait 10 seconds before continuing
```

WATCH

? and W[x]? Commands. To display C variables.

format

?[*..]name[,[#][x | s | d]] [=value]
w[z]?[*..]name[,[#][x|s]] [=value]
(descriptions and example at the end of this paragraph)

description

If a symbol is multiply defined, the local variable is taken before a global variable. When the watch window is enabled (visible) you may define new watchpoints by typing 'w[x]' before the expression described above.

The Watch window is enabled with ENA WAT or Ctrl+W.

The w[z]? Command:
    Syntax: w[z]?[*..]name[,[#][x|s]] [=value]
where [z] is an optional number to specify in which location

in the Watch window the variable is to appear. If the [z] is omitted the current contents in the Watch window will scroll up.

Definition of locations in the watch window, from top to bottom:

1.
2.
3.
4.

example w2?tbl

WRI

Write

abbreviation **WRI**

format

```
WRITE [expression]
      [string]
```

description This command displays an (evaluated) expression or a string of characters for a sequence of such elements on your monitor. It may be used in macros.

EXPRESSION can be a number, reference, or formula that equates to the number expressed. For WRITE, the expression is evaluated and the result is displayed in hex.

STRING is a sequence of characters enclosed in apostrophes (').

examples

```
WRI 'START PUMP 1'       ;displays message.
WRITE .SEC,
'SECONDS PASSED'         ;expression and text.
```

XBYTE

eXternal Byte See CB

XTAL

XTAL

format XTAL [=expression]

description This command scales the timestamping for the Advanced Trace Board. The default is 12 MHz. This command will cause the timestamp resolution to change.

Alternatively, the value can be input on the startup command line with the -x switch.

example

```
XTAL = 11.0592
```

:name

(See MACRO)

25

25, 43, 50 Commands

format

```
25
43
50
```

description

Used to switch to 25, 43 or 50 lines on the display. These commands will only work if your hardware is set up to handle 43 or 50 lines. (EGA for 43 lines and VGA for 50 lines).

?

See WATCH.

4 PODS

Pod Selection 4-3
Connection and Setup 4-10
Adapters 4-10
"Generic" Board: 40-Pin External Mode POD,
POD-31, POD-C31, POD-32, POD-C32, POD-44,
POD-C154, POD-C252/C51FA, POD-C321, POD-C652 4-13
POD-31S 4-17
POD-51 / POD-C51 4-21
POD-C51B, POD-C652B 4-31
POD-C51FX 4-37
POD-C51GB-PGA 4-45
POD-C51SL-16 4-49
POD-C52 4-55
POD-C054 4-63
POD-C152-DIP 4-69
POD-C152-PGA 4-73
POD-407-14.75 4-79
POD-CL410 4-85
POD-C451-DIP 4-91
POD-C451-PGA 4-95
POD-C451B-PGA 4-99
POD-C452-PGA 4-105
SAB C502 Emulation 4-109
SAB C503 Emulation 4-111
POD-C515A-PGA-18 4-113
POD-C517A-PGA-18 4-123
POD-C517AB-PGA-18 4-125
POD-C517B-PGA 4-131
POD-C528 4-143
POD-532-PGA, POD-535-PGA, POD-C535-PGA . . . 4-151
POD-C537-PGA 4-161
POD-C550-PGA 4-167
POD-C552-PGA, POD-C562-PGA 4-173
POD-C552B-PGA 4-177

4

4 PODS

POD-C558-16 4-183
POD-C575 4-191
POD-CL580 4-197
POD-C592-PGA 4-203
POD-C598-KIT 4-209
POD-C751 4-211
POD-C752 4-215
POD-CL782 4-221
POD-5001-16 4-227

4

4

PODs

This section of the manual gives detailed information about the types of POD boards that are available with the EMUL51-PC series of emulators for the 8051 family of microcontrollers.

Pod Selection

Which POD you pick for your application depends on several things. What micro you are using, the frequency, the mode you are using it in, your target configuration, and price all play a part. The **EMUL51-PC ORDER INFORMATION** can help you choose among the POD types available from Nohau.

Nohau has four categories of PODs:
(1) External-mode PODs
(2) Bondout PODs
(3) "Hooks"-mode PODs, and
(4) Internal mode or Port Substitution PODs

External-Mode PODs External or microprocessor-mode PODs have several clear advantages. They are the cheapest type of POD. They have easily-replaceable standard microcontrollers. Your program runs on the real production part, with most of the lines directly connected to your target. You can use them where your microcontroller has external program or external read-write memory. With this type of POD, Port 2 is used only for the upper address bus and not as port input-output (I/O). Port 0 is used as a multiplexed address and data bus. In most of these PODs, P3.6 can only be used as the write line and P3.7 as only the read line. The **POD-31** is a typical external POD.

Your target system shares Port 0 and Port 2 with the emulator. The emulator uses Port 0 to run its monitor code when it is not executing your code in real-time. When you are not running real-time emulation, the POD holds the Program Store ENable line (PSEN/) high so that your external read-only memory (ROM) is not enabled. It also holds the ReaD/ line and WRite/ lines high so that none of your external read-write random access memory (RAM) or I/O is enabled.

But in the monitor (not emulating) mode, the address lines of both Port 2 and Port 0 are active and can output any address in the 64 kByte address range. It is up to your target system to prevent enabling any device unless the PSEN/ or the RD/ or the WR/ line goes low.

If you are emulating a ROM-less part, like the 8031, you can use the **POD-31** type of POD. You also might use this POD for some internal ROM applications. You could use it for designs that have all of Port 2 and Port 0 used as external buses. This is true even though the program can be executed from on-chip 8051 program memory in your final product. This is a case where your target schematic, rather than the device you are using lets you use this type of POD. So though the part may ultimately be an 8051 or 8751, you can use a **POD-31** because you are using the ports as buses.

If you are emulating the 80C31, 8032, 80C32, 8344, 80C51FA, 80C154, 80C321, or 80C652, you can get external-mode PODs for these microcontrollers. With the **POD-31**, you can emulate all those parts in external mode at up to 12 MHz. You can get the POD with the correct micro installed, such as a **POD-32**. Or you can change among the types by changing the microcontroller component in the POD yourself.

16 MHz, 20 MHz, 24 MHz, 30 MHz and 33 MHz PODs are available for 40-pin parts. If you use a **POD-C31-1** with a 16 MHz rated emulator, you can support such parts as the 80C31-1, 80C32-1, 80C51FA-1, 80C154-1 and 80C652-1. Any higher-frequency POD and emulator set can support all the lower frequencies.

For emulating the 80C51GB, 80C152JA/JB/JC/JD, 80C451, 80C452, 80C552,80532, 80535, 80C535, or 80C537, other external mode PODs are available. PODs for dual inline package (DIP) parts have a DIP plug coming out of the bottom of the POD. PODs for plastic leaded chip carrier (PLCC) parts have a pin-grid array (PGA) plug coming out from the bottom of the POD. To plug a PGA POD into a PLCC socket, you can get an optional adapter. The PGA68-PLCC68 is a typical adapter. If your target board has feed-through holes and you solder a PGA socket into it, you don't need an adapter. If your target board already has a PGA socket, you don't need an adapter.

POD-31-S: All the above external PODs use P3.6 as WR/ and P3.7 as RD/. But there is a 40-pin external mode POD version that lets you use these two lines as I/O. The basic board is the **POD-31-S**. This special -S option lets you select whether the two lines are to be WR/ and RD/ or else I/O. They can be unidirectional port lines (input-input, input-output, output-input, or output-output). You also can get this POD in the variations of 16 MHz and 20 MHz and any of the 40-pin part variations listed above.

If you are emulating the 80C152 or 80C452, you may want to know about direct memory access (DMA). For DMA, you can get the DMA option emulator. With it you can emulate all the DMA modes. Regarding 80C152 "E-Bus" (EPSEN/) execution, the emulator can execute program from Port 0, but not from the E-Bus Port 5.

POWER UP / POWER DOWN SEQUENCES FOR EXTERNAL-MODE PODS:

The first choice for power selection is "internal" position, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout PODs.) With the "internal" selection, the recommended power sequence is:

a. Power ON the PC (or box option).
b. Power ON the target.
c. Use emulator for session.
d. Power OFF the target.
e. Power OFF the PC (or box option).

Selecting external or "target" power is normally not necessary. If you want to determine how much current the microcontroller draws, you could use an actual part in the circuit instead of measuring using the POD. If you select "target" power for external-mode PODs, the following power-up power down sequence is then recommended:

a. Power ON the target.
b. Power ON the PC (or box option).
c. Use emulator for session.
d. Power OFF the PC (or box option).
e. Power OFF the target.

Another possibility is to have both the PC and the target system on the same power switch. That way the power to both will be turned on at the same time and off at the same time.

Bondout PODs Bondout PODs use special microcontrollers that the semiconductor manufacturers designed to perform all the functions of the part. They also have additional lines "bonded out" from the chip that allow control for emulation. PODs with these special chips cost more than PODs with widely available commercial chips, but give you a greater flexibility in how you use the ports. These PODs allow you to use any combination of internal or external mode of the chip, and respond to the state of the External Access (EA/) pin.

Nohau has bondout PODs for many different microcontrollers. You can emulate the 80/80C51, 87/87C51, 80/80C31, 80C/83C/87C652, 83C/87C654, 80C/83C662 and 80C/83C851 with the **POD-C51B**. 16 MHz and 24 MHz versions are available. To emulate the 83C451, 87C451 and 80C451 in the PLCC package, you can use the **POD-C451B-PGA**. See the previous discussion about PGAs. For emulating the 80C/83C/87C552 and 80C/83C/87C562 you can use the **POD-C552B-PGA**. It also is available in 16 MHz and 24 MHz versions.

For emulating the 83C751 and 87C751, use the **POD-C751**. It has a 24 pin DIP plug on the bottom of the POD. You also can get an adapter to plug into 28 pin PLCC sockets. A 16 MHz version is available.

The **POD-CL410** emulates the 83CL410, 83CL610, 80CL31 and 80CL51 at voltages down to 1.5 volts.

The **POD-CL782** emulates the 8XCL781 and 8XCL782.

The **POD-C517B-PGA** can emulate the 80C517, 80C537, 80515, 80C515, 80535, 80C535, 80512, and 80532. It has an 84-pin PGA plug. An adapter to plug into 68-pin PLCC sockets is included. If you are plugging into an 84-pin PLCC socket, you will need an optional adapter.

For the 8XC517A and 8XC515, Nohau has the **POD-C517AB-PGA-18.**

The **POD-5001-16** emulates the DS5000, 500FP, 5000T, 5001FP and 5002FP Adapters are available for both DIP and QFP.

You should be aware of one small problem that most 8051 bondout parts have with serial communication. Specifically, if your program has written to SBUF, but the TI flag has not been set, and you then break emulation, the TI flag will never be set after you resume emulation.

Bondout PODs give you the best of both worlds. They let you mix how you use the ports. So you can emulate the single-chip mode using the ports as I/O. Or you can emulate external bus mode. Or you can emulate a mix of both modes.

Limited Warranty on Bondout Chips

The POD-C51B (POD-C652B), POD-C552B-PGA, POD-C517B-PGA, and POD-C751 all use bondout parts especially designed to be used to emulate their standard counterparts. All bondout pods are delivered with chip power jumpered to the POD (coming from your PC). This makes it possible to run the pod without having it connected to a target system.

Bondout parts are fairly expensive (prices range from $150 to $500). For maximum protection of these chips please follow the instructions below:

When you connect the pod to your target you should change the power jumper so that the "bondout" gets its power from the target. If you do that it should not matter in which order you power up the PC and the target. However, if you leave the power jumper so that power is taken from the POD (PC), and plug it into the target, the PC must **ALWAYS** be on when the target is turned on.

POWER UP / POWER DOWN SEQUENCES FOR BONDOUT PODS:

When the power jumper is in the "PC position" you MUST sequence the power as follows:

a. Power ON the PC (or box option).
b. Power ON the target.
c. Use emulator for session.
d. Power OFF the target.
e. Power OFF the PC (or box option).

Steps b, c and d may be repeated any number of times as long as PC's power is on.

When the Power jumper is in the "target position", power can be turned on and off in any sequence.

We recommend that the PC and the target system's power be drawn from the same outlet to minimize ground currents. Also, be sure to always connect the black "hook wire" from the pod to a solid ground on the target before connecting the pod to the target.

WARRANTY WILL BE VOIDED IF THE POWER JUMPER IS IN "PC POSITION", THE PC IS OFF AND THE TARGET IS ON.

"Hooks"-Mode PODs Hooks mode PODs use a different technique for emulating. Certain chips can be put into a proprietary "hooks" mode that multiplexes the port information in and out of the part. The chip can still put out address and data. This group includes these PODs:

The **POD-C51FX** can emulate the 80C/83C/87C51FA/FB/FC. It can also emulate the parts that the POD-C52 can. It is available in a 16 MHz version.

The **POD-C52** can emulate the 80/80C/87/87C51, 80/80C/87/87C52, 80/80C31 and 80/80C32. You can use this POD when you need to emulate the 8052-type parts in single-chip mode when you are using Timer 2.

The **POD-C528** can emulate the 80/80C/87/87C528. A 16 MHz version is available.

The **POD-C550-PGA** can emulate the 80/80C/87/87C550. It has a 44-pin PGA plug. You also can get an adapter to plug into 44-pin PLCC sockets.

The **POD-C558-16** emulates the 83CE558 and 89CE558.

The **POD-C575** emulates the 8XC575.

The **POD-C592-PGA** can emulate the 80C/83C/87C592 "CAN" (Controller Area Network) bus microcontroller. It is available in a 16MHz version.

The **POD-C752** is for emulating the 83C752 and 87C752. It has a 28 pin DIP plug on the bottom of the POD. You also can get an adapter to plug into 28 pin PLCC sockets. A 16 MHz version is available.

The **POD-C054** can emulate the 8XC053 and 8XC054 "MTV" (Microcontroller for Television) chip.

With the "Hooks"-Mode PODs, you use jumpers to select whether the code is to be fetched from the emulator RAM or from external ROM. You also select whether the read-write memory is in the emulator or on your target board. The ports on these PODs may have slightly different electrical characteristics than the microcontroller. Specifically, some output signals can appear up to three clock cycles later that on the actual part. Also, some ports have greater current sink or sink and source capacity or slightly higher impedance than the actual part. For almost all applications, these differences will have no detrimental effects.

POWER UP / POWER DOWN SEQUENCES FOR HOOKS PODS:

Power selection should be from the "internal" position, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout PODs.) The recommended power sequence is:

a. Power ON the PC (or box option).
b. Power ON the target.
c. Use emulator for session.
d. Power OFF the target.
e. Power OFF the PC (or box option).

Selecting external or "target" power is NOT recommended. If you want to determine how much current the microcontorller draws, we suggest that you use an actual part in the circuit instead of measuring using the POD.

Powering on the target system while the computer (or box option) is not powered is NOT recommended.

Port Substitution or Internal-Mode-Only PODs This is an older POD design used before Nohau began manufacturing bondout and "hooks" mode PODs. The internal mode PODs are the **POD-51** and the **POD-C51**. These PODs use microcontrollers with two extra ports, the 80535 and 80C535. In these PODs, Port 4 on the '535 connects to the Port 0 lines on the plug on the bottom of the POD, and Port 5 connects to the Port 2 lines. This way, you have full use of all four 8051 ports as I/O.

There are some drawbacks to this method. One is that the ports of the internal-mode PODs can only be used as port I/O, and not as buses for external ROM or RAM. Another is that you must change addresses of three registers (P0, P2, and IP) in your source code in the version that you run on the emulator. Also, IP is only byte-addressable in these PODs.

These PODs cannot support Timer 2 of the 80/80C/87/87C52. The PODs can support 256 bytes of internal data RAM, and the emulation of 8 kBytes of internal program ROM.

POWER UP / POWER DOWN SEQUENCES FOR PORT SUBSTITUTION (Internal-Mode-Only) PODS:

The sequence for these PODs is the same as for external mode PODs. See the section "POWER UP / POWER DOWN SEQUENCES FOR EXTERNAL-MODE PODS" for recommendations.

You can use any size **EMUL51-PC** emulator and optional trace board with any POD you choose in Nohau's **EMUL51-PC** family. Choose an emulator and trace board with frequencies as fast or faster than the frequency of your fastest POD. If you need more information about which POD would best fit your application, please contact Nohau.

Connection and Setup

Connection to the emulator board is made with the ribbon cable supplied with the POD board. Connection to the empty microcontroller socket in the target system is made through pins provided on the underside of the POD board. (For connection details, refer to **Adapters** paragraph in this section.) A socket on the POD board holds the POD microcontroller (which is included with the POD board). After selecting a specific type of POD board, you have to decide how you want to set up the various POD board jumpers (and also switches in some cases). To do this, on the following pages refer to the board layout illustration, diagram of jumper locations and table describing the jumper choices for the type of pod board you are using.

Adapters

Connection to the empty microcontroller socket in the target system is made through socket pins provided on the underside of the POD board. Because these pins have the same footprint as the microcontroller to be emulated, you often can plug the pod directly into the socket. If height of components on the target system does not allow this, to obtain the necessary clearance try adding extra sockets as shown in Figure 1. If that approach does not work out, then use an adapter instead.

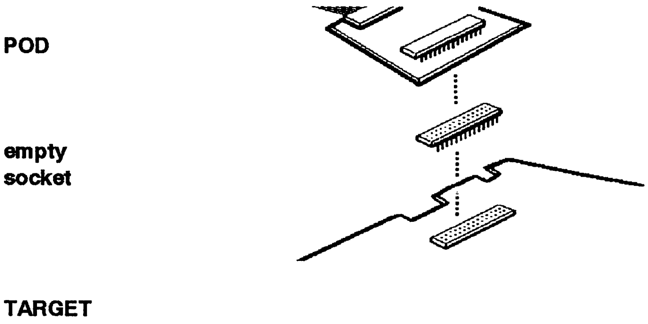


Figure 1. Empty Socket Inserted Between POD and Target System

DIP40-ISO This is a 2 1/2" high 40-pin DIP adapter with DIP switches for each of the 40 pins. Figure 2 shows how it raises the pod over the target system to make it easy to probe on the target under the pod. The DIP switches let you isolate signals when you are bringing up a new prototype or testing production boards.

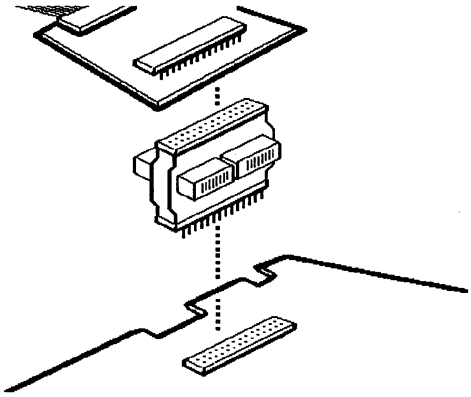


Figure 2. DIP40-ISO Adapter Inserted Between POD and Target

EXT-DIP40 This is a 6-inch long, 40-pin DIP extender cable. As shown in Figure 3, the EXT-DIP40 adapter could be a good solution if the target system is in a box where the pod will not fit. However, a potential problem is that the emulator gets more sensitive to noise.

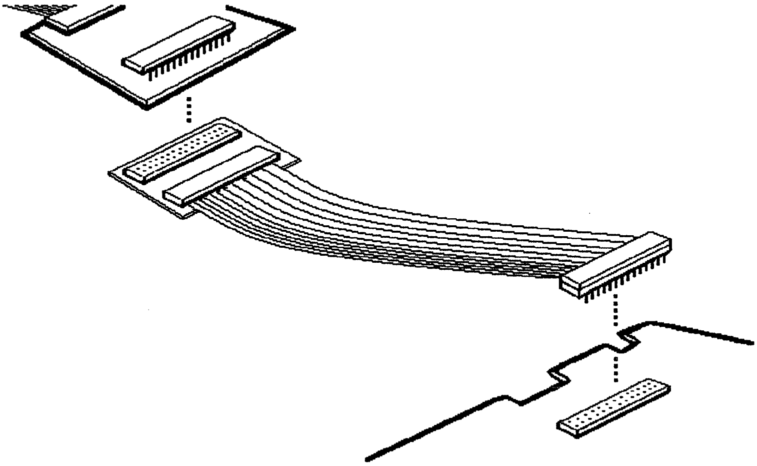


Figure 3. EXT40 Adapter Inserted Between POD and Target

PGA68-PLCC68 This is a 68-pin PGA to PLCC adapter. POD-C451-PGA, POD-535-PGA, and POD-C552-PGA all have a 68-pin PGA socket coming out on the back of the POD board. However, if your target board has a PLCC socket, you will need this adapter.

DIP40-PLCC44 This is a 40-pin DIP to 44-pin PLCC adapter. It is useful for connecting regular 40-pin DIP pods (such as POD-31) to a surface-mount board that has a 44-pin surface-mountable microprocessor. (Instead of soldering the microprocessor to the board, an AMP 44-pin PLCC socket is soldered instead, and this adapter fits into the socket.)

For more adapters and cables, please refer to a current price list.

"Generic" Board: *POD TYPE: External Mode*

40-Pin POD-31, POD-C31, POD-32, POD-C32, POD-44,
POD-C154, POD-C252/C51FA, POD-C321, POD-C652

Also higher frequency versions.

Identifying the POD

The POD has a 40-pin DIP socket on it and is shown in Figure 4. The silkscreen reads simply "POD BOARD".

To work together with the standard emulator boards EMUL51-PC/E32 (32K emulation memory) and EMUL51-PC/E128 (64K code memory and 64K external data memory), any one of several microcontrollers can be plugged into a 40-pin socket on the "generic" board, as follows:

POD-31 for 8031, **POD-C31** for 80C31, **POD-32** for 8032, **POD-C32** for 80C32, **POD-44** for 8344, **POD-C154** for 80C154, **POD-C252/C51FA** for 80C51FA, **POD-C321** for 80C321, **POD-C652** for 80C652.

These pods will all work at up to 12 MHz. In addition, the higher frequency PODs are guaranteed to run at up to their rated frequencies.

Two jumpers make it possible to use an external crystal or clock. Power to the microcontroller chip can be taken either from the target system or from the pod. (The pod logic receives its power from the PC).

Because you can exchange microcontrollers on the "generic" pod board, the same board can be used to emulate any of the above mentioned microcontrollers.

These pods emulate the microcontrollers only in their "external" mode. This means that P0 is used as a data port, P2 is used as an address port, P3.6 is used as WR, and P3.7 is used as RD. For "internal" emulation, refer to specification for POD-51, or POD-C51B.

For other POD boards, please refer to the remaining pages of this chapter.

POD-31, -C31, -32, -C32, -44, -C154, -C252/C51FA, -C321, -C652

Table 1. Designators for POD-31, POD-C31, POD-32, POD-C32, POD-44, POD-C154, POD-C252/C51FA, POD-C321, POD-C652

Designator	*Name*	*Description*
JB1	RST Selection	With JB1 installed, pin 9 on the POD board microprocessor (RST) is connected to the target system, which allows reset to be performed by S1 or the target system. With JB1 removed, RST is connected only to the S1 switch and the PC controlled reset line. This is useful if you have a software controlled watchdog that keeps resetting the microprocessor.
JB2	POD Board Power	This jumper determines where the pod board gets its power. With the jumper toward the 8031, power is received from the target. If the jumper is in the opposite position (see Figure 5), power is taken from the PC.
JB3	Port Connections	Ports P1-0 through P1-7 and P3-0 are connected to the trace board when the jumpers are in their preset positions. If the jumpers are removed the upper pins (closest to the edge of the board) may be connected to external signals with the optional EZ-hooks. These signals will then be traced instead of the ports.
JB4	Standby External Signal	These 4 pins can also be connected to external signals with the optional EZ-hooks. No jumpers are used here.
JB5, JB6	Crystal Selection	With JB6 jumpers installed, the crystal on the POD board is connected to the microprocessor. When the two jumpers are installed in JB5, the crystal of the target system is connected to the microprocessor.
JB7	PSEN Signal	In the factory installed position, the PSEN signal only goes out on the target system in emulation mode. In the alternate position, PSEN will always go out to the target system. In that case be sure that PSEN is not used to open anything that drives the databus. (Remove PROMs).
DS1	Emulation LED	This LED will be lit when emulation is running. It is driven by the EM/ signal on TP.
GND Wire	Ground Wire	This ground wire is to connect the POD ground to the target ground before the POD is plugged into the target. Connecting this ground reduces the chance of electrostatic discharge.

POD-31, -C31, -32, -C32, -44, -C154, -C252/C51FA, -C321, -C652

Processor types that can be used with the -p parameter in the invocation: For 8031, 80C31, or 80C31-1, use -p8031; for 8032, 80C32, or 80C32-1, use -p8032; for 8344 use -p8344; for 80C51FA, use -p8051FA. (NOTE: This type is not the same as the 8051 internal-ROM type part. See one of the following PODs for that type: POD-51, POD-C51B, POD-C51FX or POD-C52.) For 80C154 or 80C154-1, use -p80154; for 80C652 or 80C652-1, use -p80652.

POD-31, -C31, -32, -C32, -44, -C154, -C252/C51FA, -C321, -C652

Board Layout Figure 4 shows locations of jumpers and Figure 5 shows a diagram of the jumper pins.

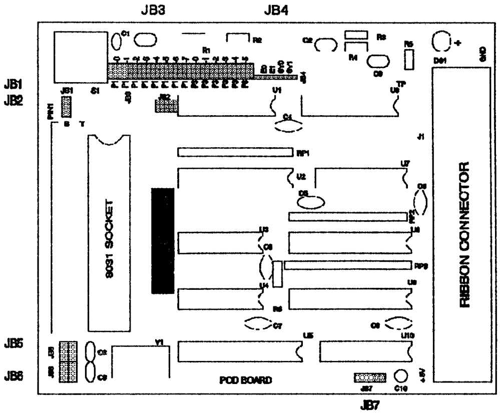
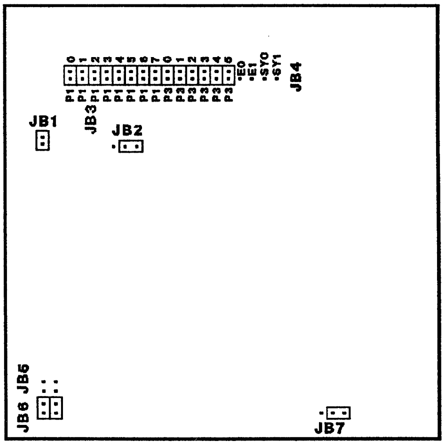


Figure 4. Locations of Jumpers on POD-31 Board

JB2: Internal Power Shown



JB6 Pair: POD Crystal

Figure 5. POD-31 Board Configured for internal power and internal crystal

POD-31S and Variations *POD TYPE: EXTERNAL MODE*

Table 2. Designators for POD-31-S

Designator	*Name*	*Description*
JB1 or J1:	Crystal Selection	If the two switches are in position I, the crystal on the pod is connected to the microcontroller. In position E the crystal on the target system is used.
JB2	Power Selection	When this switch is in position I, the power to the microcontroller comes from the PC. In position E, power comes from the target system.
JB3, JB4	Port Selection	Ports P1.0 through P1.7 and P3.0 through P3.5 connected to the trace board when these jumpers are installed. If the jumpers are removed, signals may be connected to the pins closest to edge of the board with the optional EZ-hooks.
JB5	PORT 3 I/O Control	Move this switch to position 3 when P3.6 and P3.7 are used as WR/ and RD/. When P3.6 and P3.7 are used as I/O, JB5 should be in position 1. (See table.)
JB6, JB7	PORT 3 I/O Control	These switches should be in position 1 when and P3.7 are used as WR/ and RD/. In position 3, P3.6 and P3.7 are used as I/O. (See table.)
JB8	PSEN Signal	If this switch is in position 1, the PSEN/ signal will only go out to the system in emulation mode. If the switch is in the opposite position, the PSEN signal will always go out to the target system. In that case, be sure that PSEN/ does not enable anything (such as a PROM) that drives Port 0, the data bus.

(Table continued next page)

POD-31S and Variations

JB9	RST Selection	With this jumper installed, pin 9 on the POD board microcontroller (RST) is connected to the target system, which allows reset to reach the micro-controller from your target system. With JB9 removed, RST is connected only to the S1 switch and the emulator-controlled reset line. This is useful if you have a software controlled watchdog that keeps resetting the microcontroller.
JB10	PORT 3 I/O Control	If the jumper is in the position nearest the edge of the PCB, then P3.6 is WR/. If the jumper is in the opposite position, then P3.6 is an OUTPUT. If no jumper is installed, P3.6 is an INPUT. (See table.)
JB11	PORT 3 I/O Control	If the jumper is in the position nearest the edge of the PCB, then P3.7 is RD/. If the jumper is in the opposite position, then P3.7 is an OUTPUT. If no jumper is installed P3.7 is an INPUT. (See table.)
S1	RESET	This switch only resets the POD's micro-controller, not the target system.
J2	EMULATOR	This pin is at a low state during emulation, which is also indicated by the LED.
J4	GND	This ground wire is to connect the POD ground to the target ground before the POD is plugged into the target. Connecting this ground reduces the chance of electrostatic discharge.
J5	SY0, SY1, E0, E1	These pins may be used to trace external signal.

POSSIBLE SETUPS

P3.7	P3.6	Batch	JB5	JB6	JB7	JB10	JB11
RD/	WR/	No "-s"	Pos 3	Pos 1	Pos 1	Edge	Edge
Output	Output	-soo	Pos 1	Pos 3	Pos 3	Middle	Middle
Input	Input	-sii	Pos 1	Pos 3	Pos 3	Remove	Remove
Output	Input	-soi	Pos 1	Pos 3	Pos 3	Remove	Middle
Input	Output	-sio	Pos 1	Pos 3	Pos 3	Middle	Remove

SETUPS NOT POSSIBLE

P3.7	P3.6
RD/	i/o
i/o	WR/
Bidirectional	

Parameter -p is the same as generic board variations: -p8031, -p8032, -p8344, -p8051FA, -p80154 or -p80652.

POD-31S and Variations

Board Layout

Figure 6 shows locations of jumpers and Figure 7 shows a diagram of the jumper pins.

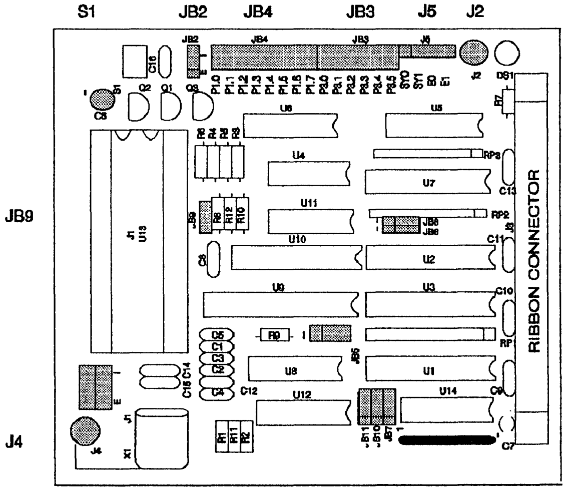


Figure 6. Locations of Jumpers on POD-31-S Board

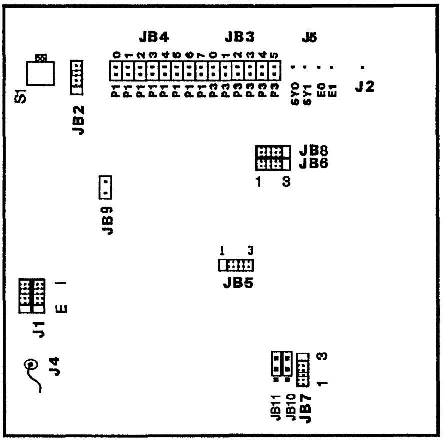


Figure 7. POD-31-S Board Configured for internal power and internal crystal.

POD-51, POD-C51 - Rev. B and C *POD TYPE: INTERNAL MODE*

POD-51/POD-C51 Specification The POD-51 Port Substitution POD emulates the 8051, 8052, 8751 and 8752 processors in their internal mode only. (With regard to 8052 and 8752, however, Timer 2 works differently in the emulation processor (80535) than on the 8052 and can not be emulated. Refer to Siemens data book for the 80535 or contact Nohau Corp.) By "internal mode only" we mean that ports P0 and P2 are used as general purpose ports and NOT as data and address ports. If P0 and P2 are used as data and address ports with RD (P3.7) and WR (P3.6) used only as read and write signals (in other words if you use any MOVX instructions) you should use POD-31/32 or POD-C51B, and not POD-51. Note that this POD is not the more commonly used POD-C51B bondout POD, which does both internal and external mode emulation.

Using POD-51 First you need to decide if P3.6 (WR) and P3.7 (RD) will be used as inputs or outputs. The two jumpers at JB4 decide this. In their preset positions (jumpers installed) they are used as outputs. If one or both jumpers are removed corresponding pins are used as inputs. The jumper closest to S1 is for P3.7.

We use the 80535 (Siemens) as the emulation processor. P4 and P5 are used to emulate P0 and P2. The addresses for P0 and P2 in the 8051 are 80H and A0H while P4 and P5 are at E8H and F8H. The IP register occupies different addresses in 8051 and in 80535. IP in 8051 is at B8H while it occupies A9H in the 80535. Therefore P0, P2 and IP must be redefined. Also, IP is not bit-addressable in the 80535. So you may not use a SETB or CLR instruction on the IP register in the POD-51.

This conversion should be taken care of in your source code. The following describes how this is done in different assemblers, PL/M-51 and in C51.

ASM51 (INTEL). Use the NOMOD51 option either in your invocation line or as an option line in the source code. This will cause the assembler to not recognize predefined symbols (like P0, P2 and IP). Thereafter, in the source code, use the include command to include one of the PDF files (Processor Definition Files). The PDF files are distributed together with the ASM51. However, before you use the PDF file use your editor to redefine P0, P2 and IP to E8H, F8H and A9H. Here is an example.

Command line: ASM51 MYPROG.SRC NOMO DB (NOMO is short for NOMOD51, DB means that you put symbols in your obj code.)

POD-51, POD-C51 - Rev. B and C

In source file: INCLUDE REG52.PDF (in the source code, REG52.PDF is the modified PDF file from your ASM51 diskette.)

Before you burn a PROM to run with the actual 8051, 8052, 8751 or 8752 in your target system you need to remove the NOMOD51 and INCLUDE directives from your source and reassemble the code to get the right addresses for P0, P2 and IP.

Please also refer to Appendix L in your ASM51 manual.

AVMAC51 (Avocet). Use the TEQ directive in the source code to redefine P0 and P2. Here is an example.

```
P0      TEQ E8H
P2      TEQ F8H
IP      TEQ A9H
```

Before you burn a PROM to run with the actual 8051, 8052, 8751 or 8752 in your target system, you need to remove these definitions from your source and run the assembler to get the right addresses for P0, P2 and IP.

PL/M-51 (Intel). Some declaration files are supplied together with the PL/M-51 compiler, and one of them is called "REG51.DCL". This file contains all REGISTER declarations for the 8051 processor and should be included in the source code with the $INCLUDE control. Copy this file to a new file "REG51E.DCL" and redefine P0, P2 and IP to E8H, F8H and A9H respectively.

When using the EMUL51-PC with POD-51, include this new file in the source code by writing: $INCLUDE (:F1:REG51.DCL) and then to make PROM-able code for the target system, change the include line in your source to: $INCLUDE (:F1:REG51.DCL)

Please also refer to Appendix I in your PL/M-51 manual.

Archimedes C-51. The file IO51.H on your C-51 contains the definitions of all registers. Please modify P0 to E8H, P2 to F8H and IP to A9H.

The assembler is more difficult. You have to use your editor to change P0, P2 and IP to for example P0E, P2E and IPE everywhere in the source file. On top of the file then define P0E as E8H, P2E as F8H and IPE as A9H. Before you burn a PROM to run with the actual 8051, 8052, 8751 or 8752 you only need to redefine P0E to 80H, P2E to A0H and IPE to B8H and assemble again to create a new obj file to be used with the PROM programmer. Also don't forget to recompile with the original values for P0, P2, and IP in IO51.H!

OTHER ASSEMBLERS. For assemblers not described above, you want to try to find way of redefining P0 or P2. If nothing else works, you can use your editor to change P0, P2 and IP to, for example, P0E, P2E and IPE, everywhere in the source file. On top of the file then define P0E as E8H, P2E as F8H and IPE as A9H. Before you burn a PROM to run with the actual 8051, 8052, 8751 or 8752, you only need to redefine P0E to 80H, P2E to A0H and IPE to B8H and assemble again to create a new obj file to be used with the PROM programmer.

ADDRESSES TO BE REDEFINED:

Register	8051-8052-8751-8752	POD-51
P0	80H	E8H
P2	A0H	F8H
IP	B8H	A9H

We would appreciate very much if you let us know other ways to obtain a redefinition of P0, P2 and IP using different kinds of assemblers and compilers.

Invoking the Emulator

When you load the EMUL51 program specify either 8051 or 8052 for processor type using the -p parameter. For 8051, 8051-1 and 80C51, use -p8051; for 8052 use -p8052. Note that timer 2 is not supported. You also need to specify if P3.6 and P3.7 are used as inputs or outputs. This is done on the command line by adding two letters in the -s option: "i" for input and "o" for output. The first letter denotes P3.7 and the second denotes P3.6. Some examples are given below.

```
EMUL51 -p8052 -e110 -m32 -t100 -f4 -sii
```

This means that you are emulating an 8052 (or 8752), both P3.6 and P3.7 will be used as inputs, the emulator board is at address 110H, 32K are installed on the emulator, the trace board is at 100H, and 4K are installed on the trace board.

```
EMUL51 -p8052 -e110 -m32 -t100 -f4 -soi
```

This is like the previous example, except that p3.7 will be used as an output.

POD-51, POD-C51 - Rev. B and C

It's a good idea to create a batch file, so that you won't have to type the complete command line every time you invoke the emulator! When you use the in-line assembler and disassembler in the EMUL51, you don't need to worry about the addresses of P0, P2 and IP as long as you use them as symbols. If you have invoked the emulator using 8051 or 8052 as processor type, it will automatically be taken care of. But if you use the hex equivalent of the addresses in any context, you must use E8H for P0, F8H for P2 and A9H for IP. If the disassembler finds address 80H (P0), A0H (P2) or B8H (IP), it will write P0-WARNING, P2_WARNING or IP_WARNING.

Determining Revision Level To determine which figure and table to use, you will need to find out which revision of the POD board you have.

Revision B of the board only has jumpers on it, and the only switch on the board is the reset push button. For this board, use Table 3 and Figures 8 and 9. Note that this POD-51 Rev B is a different POD than the POD-C51B bondout POD. See the separate section for that POD.

Revision C of the board has slide switches in place of some (but not all) of the jumper blocks. For this board, use Table 4 and Figures 10 and 11.

POD-51, POD-C51 - Rev. B

Table 3. Designators for POD-51 Rev.B (8051, 8751) and POD-C51 Rev.B (80C51 and 87C51)

Designator	*Name*	*Description*
JB1	PSEN signal	If jumper is installed as shown in Figure 9, PSEN only goes out to target system when target CODE memory is addressed. If jumper is installed in the opposite position, PSEN will always go out to target system.
JB2	PORT 1 and 2	With JB2 you decide if trace should be connected to port 1 or port 2. If a jumper is installed in position P2.4 then port 2 bit 4 will be traced. If no jumper is installed, the middle pin (or pins) could be connected to external signals with the optional EZ-hooks. These signals will then be traced instead of the port signals.
JB3	PORT 0 and 3	These jumpers work exactly like JB2, but on port 0 and port 3.
JB4	PORT 3.6 and 3.7	With JB4 you decide if port 3.6 and port 3.7 should be inputs or outputs. Jumper installed makes the port output, and not installed makes it an input. The jumper closest to the edge of the board controls P3.7, and the other jumper controls P3.6.
JB5	CHIP POWER	This jumper determines where the 80535 chip gets its power from. If a jumper is installed as shown in Figure 9, the 80535 chip receives its power from the PC. If a jumper is installed in the opposite position, the 80535 chip will be powered from the target system.
JB6	RST Selection	With jumper JB6 installed target system will be able to reset the pod CPU by pulling pin 9 on connector J3 high. If no jumper is installed, the target system will not be able to reset the CPU. (This is useful if you have a watchdog on your target system.)
JB7	CRYSTAL Selection	With jumpers installed as shown in Figure 9, the pod crystal will be connected to the CPU. With jumpers installed in the opposite position, your target crystal or osc. will be connected to the pod CPU.

(Table continued next page)

POD-51, POD-C51 - Rev. B

Designator	*Name*	*Description*
GND	GND Connection	Before you plug the pod board in to your target system, you should connect this point to your target systems GROUND. Just to be sure no static discharge or bad ground connection destroys the pod.
[NO JB]	CMOS CLK Swap	Because the external oscillator input differs between an 8051 and an 80C51, it may be necessary to swap the input lines on the OSC pins. Swapping is only necessary for CMOS, and only for an actively driven external clock (not crystal). Swap by wrapping criss-crossed wires on the "T" side of the JB7 jumpers. No swap is needed for the (non-CMOS) POD-51, nor for either NMOS or CMOS for internal or external quartz crystal.
J2	SY0 and SY1 Inputs	These 2 pins can be connected to trace external signals. SY0 may also be used to trigger breakpoints.
J3	TARGET Con.	This connector fits in your target system's CPU socket.
J4	TARGET Con.	If you are short of space it is possible to connect a flat-cable between your target- system and J4. This cable is an option.
S1	RESET Switch	By pressing this button you resets the pod. CPU The same thing as entering the command: "RES CHIP."
TP1	TRIGGER Point	This pin is low when the emulator is running. It goes high as soon as a break condition is detected. May be used to trigger a memory oscilloscope.
DS1	EMULATOR Status	This LED will be lit when the emulator is running.

POD-C51 vs. POD-51

On this board revision, the type of microcontorller (80C535 instead of 80535) plus a modification make the board a POD-C51. The modification ties pin 4 of the CMOS 80C535 to GND.

POD-51, POD-C51 - Rev. B

Board Layout Figure 8 shows locations of jumpers and Figure 9 shows a diagram of the jumper pins.

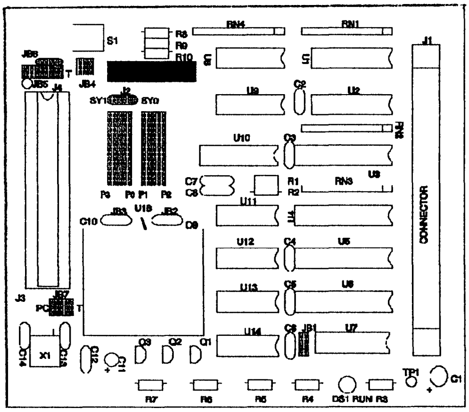
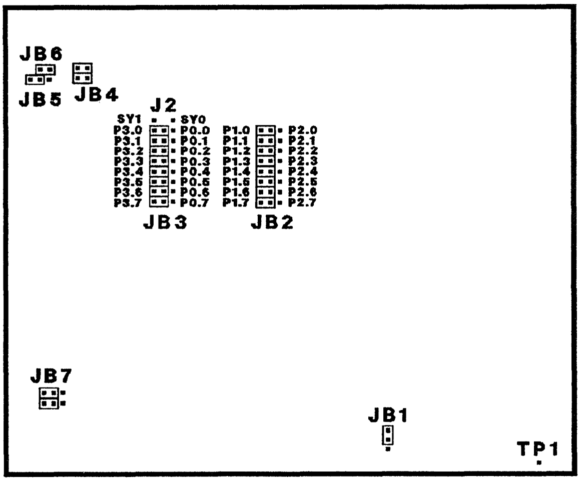


Figure 8. Locations of Jumpers on POD-51/POD-C51, Rev.B Boards

JB5: Power from POD shown



JB7: POD Crystal shown

Figure 9. POD-51/POD-C51, Rev.B Board Configured for power from POD and internal crystal

POD-51, POD-C51 - Rev. C

Table 4. Designators for POD-51 / POD-C51 Rev.C With Switches (8051, 80C51, and 87C51 Microcontrollers)

Designator	*Name*	*Description*
JB1	PSEN Signal	In the normal position, PSEN/ only goes out to your target board when CODE memory is addressed and MAPC = USER. But since the POD-51 can only fetch code from the emulator, target code memory cannot be read. In the other position, PSEN/ goes to the target all the time, even in monitor mode. Normal position for JB1 SWITCH: Position 1 (toward R4).
JB2	PORT 1 and 2	JB2 controls the connection of Trace to Port 1 or Port 2. For example, if a jumper is installed in position P2.0, then Port 2 bit 0 will be traced. In the other position, Port 1 bit 0 is traced. With the jumper removed, the middle pin can be connected to an external signal, which would be traced instead of the port line.
JB3	PORT 0 and 3	These jumpers work exactly like JB2, but on Port 0 and Port 3.
JB4	PORT 3.6 and 3.7	JB4 controls the input / output mode of ports 3.6 and 3.7. A jumper installed makes the port an output; not installed makes it an input. Also see "Invoking the Emulator" in the POD-51 Specification section.
JB5	CHIP POWER	This switch determines the voltage source for the micro-controller. Moving the switch toward the inside of the board (marked "I" for internal) selects power from the emulator; toward the outside of the board (marked "E" for external) brings power to the micro-controller from the target.
JB6	RESET	With jumper JB6 installed, the target reset line (Pin 9 on the 40-pin DIP plug) can reset the POD's micro-controller. If the jumper is removed, the target cannot reset the POD's part, which can be useful if there is a watchdog on the target.

(Table continued next page)

POD-51, POD-C51 - Rev. C

Designator	*Name*	*Description*
JB7	OSC SOURCE	This switch determines whether the POD uses the POD's crystal or the target's external crystal or oscillator. Move the two switches away from the edge of the board to use the POD crystal.
JB8	NMOS/ CMOS	This switch, plus the type of microcontroller (80535 or 80C535) determines whether the POD is a POD-51 or a POD-C51. A POD-51 uses the NMOS 80535 has JB8 switched away from the edge of the board ("N" for NMOS). A POD-C51 uses a CMOS 80C535, and has JB8 switched toward the edge of the board ('C" for CMOS). The switch ties Pin 4 of the CMOS 80C535 to GND.
JB9	CMOS CLK SWAP	Because the external oscillator input pin differs between an 8031 and an 80C31, it may be necessary to swap the input lines on the OSC pins using the JB9 switches. Swapping is only necessary for CMOS, and for an actively driven external clock (not crystal). Swap by moving the switches to the "C" position. No swap is needed for the (non-CMOS) POD-51, nor for either NMOS or CMOS for internal or external quartz crystal.
TP1	TRIG	This pin is the signal EM/, which is low when emulation is running and high when the emulator is in monitor mode. It can be used to trigger a memory oscilloscope when a breakpoint is taken.
TP2	GND	This ground wire near JB5 is to connect the POD ground to the target ground before the POD is plugged into the target. Connecting this ground reduces the chance of electrostatic discharge.
S1	RESET Switch	This switch resets the micro-controller. While the switch is pressed, the micro is held reset. The RESet CHIp command can also be used. Also see JB6.
DS1	EMUL LED	This LED display will be lit when emulation is running. It is driven by the EM/ signal on TP1.

POD-51, POD-C51 - Rev. C

Board Layout Figure 10 shows locations of jumpers and Figure 11 shows a diagram of the jumper pins and switches.

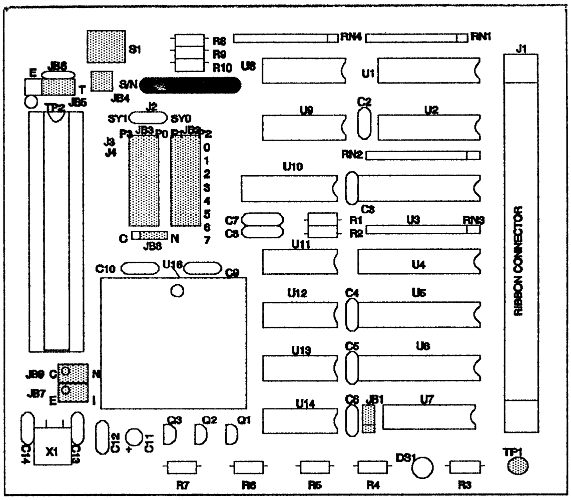
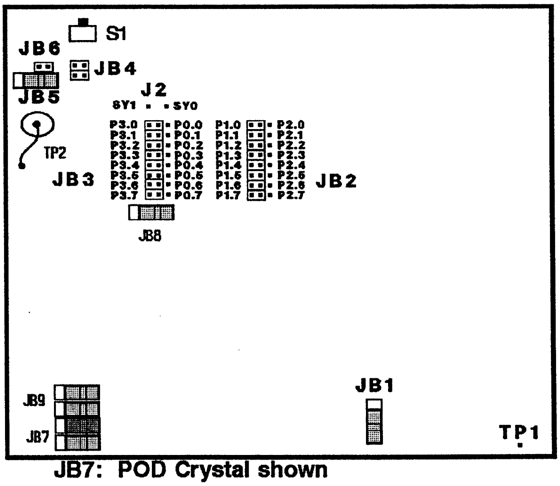


Figure 10. Locations of Jumpers on POD-51 / POD-C51 Boards Rev.C,

JB5: Power from POD shown



JB7: POD Crystal shown

Figure 11. POD-51/POD-C51 Rev.C Board Configured for power from POD and internal crystal

POD-C51B, POD-C652B *POD TYPE: Bondout*

Also higher frequency versions.
"B" stands for Bond-Out

Limited Warranty on Bond-Out Chips: Damage to expensive bond-out chips is not covered under warranty if the proper power-up and power-down sequences are not followed. Please review the power sequencing instructions on Page 4-5.

POD-C51B: The bond-out chip in the POD-C51B is a CMOS chip, but in most cases it can be used to emulate NMOS 8051's. (See Table 5 for the appropriate crystal jumper setting.)

Table 5. Designators for POD-C51B / POD-C652B (80C51, 80C652, and 80C851 Microcontrollers)

Designator	*Name*	*Description*
PSEN	PSEN Signal	PSEN gated/non-gated: In the non-gated position, the PSEN signal will be active to the user target during emulation if either the EA/ pin is low or the PSEN address is above the top of the memory boundary set with JB7, regardless of MAPC setting. A target EPROM might be unintentionally enabled in this case. The non-gated PSEN output is from the bond-out chip. In monitor (not emulating) mode, the signal will only be active for CBYTE and code window display if the MAPC setting is to target. In the gated PSEN position, during emulation the signal is only active if the MAPC is set to target and either the EA/ pin is low or the PSEN address is above the memory boundary. In monitor mode, the signal will be active to the target for CBYTE and code window display if MAPC is set to target and the EA/ pin is low or the address is above the boundary. The gated PSEN output is from a 74AS32 with a 330 ohm pullup.

(Table continued next page)

POD-C51B, POD-C652B

Designator	Name	Description
RD WR	RD Signal WR Signal	WR & RD gated/non-gated: For P3.6 and P3.7 as ports, use the non-gated position. Non-gated will ignore MAPX settings during emulation. For monitor mode XBYTE commands and external data reads, the signals will be active if MAPX is set to target. The non-gated port lines are directly connected to the bond-out chip. In the gated positions, the read and write signals will follow the setting of the MAPX in both emulation and monitor modes. Use gated if the lines are used as external memory write and read lines. ENABLE XDATA in the emulator to use the external data window. Each gated output is from a 74AS32 with a 330 ohm pullup.
X1 X2	CRYSTAL Selection	With both jumpers installed in the INT position (Figure 13), the pod crystal is connected to the bond-out chip. With jumpers installed in the EXT position, the target crystal or oscillator will be connected to the pod bond-out chip. When emulating NMOS external oscillator (not crystal), the X2 center pin is jumpered to the EXT pin of X1.
V	POWER	This jumper selects the source of power for the bond-out chip. If a jumper is installed for INT (Figure 13), the bond-out chip receives its power from the PC. If a jumper is installed in the opposite position (EXT), the bond-out chip will be powered from the target system.
JB7 JB14 JB15	MEM & CPU Selection	**MEM SIZE:** On JB7, the two jumpers nearest the ribbon cable connector select the emulated memory size. Figure 13 shows the memory set to 4k. Select 8k by inserting a jumper on the second pin away from the ribbon cable. Select 16k by removing both jumpers. Select 32k by

(Table continued on next page)

POD-C51B, POD-C652B

Designator	*Name*	*Description*
		installing both jumpers.* Memory size selection is independent of processor type selection. **CPU:** CPU type is selected by the three jumpers in JB7 farthest from the ribbon cable and by JB14 and JB15 which select the IC pin functions. **80C31/80C51** is selected as shown in figure 13; install the jumpers on JB14 and JB15 toward the reset button. Select **8XC652** by inserting jumpers on the two pins of JB7 nearest the ribbon connector and leaving the right pin open; install the jumpers on JB14 and JB15 toward the ground wire. Select **8XC654** by setting the jumpers as for 80C652, but set the memory size jumpers to 16k. Select **8XC662*** by removing jumpers on all three of the pins of JB7 nearest the ribbon connector; install the jumpers on JB14 and JB15 toward the reset button. Select **8XC851*** by inserting a jumper on the pin of JB7 nearest the ribbon connector and leaving the right two pins open; install the jumpers on JB14 and JB15 toward the reset button. **8052:** This POD cannot emulate 8052 because it does not have Timer 2. However, it can emulate 256 bytes of internal RAM if you set it up to emulate 8XC662.
JB10	PORT 0 and 1	JB10 selects either Port 0 or 1 to be traced. If a jumper is installed in position P0.4 then port 0 bit 4 will be traced. If no jumper is installed, the middle pin (or pins) can be connected to external signals with the optional EZ-hooks. These signals will then be traced instead of the port signals.
JB11	PORT 2 and 3	These jumpers work exactly like JB10, but on ports 2 and 3.
RES	Ext Reset Enable	External reset is enabled with the jumper installed.
EM	Mode Signal	LOW = Emulation Mode; HIGH = Monitor Mode.
INTA	Interrupt Signal	This pin puts out a negative pulse at the start of an interrupt routine.

* 8XC662, 8XC851 and 32k memory require that the bond-out chip be an 85C582.

(Table continued next page)

POD-C51B, POD-C652B

Designator	*Name*	*Description*
C1	Inst. Fetch Signal	The signal on this pin is high when the first byte of an instruction is fetched; it is multiplexed when ALE is high.
SY0, SY1	SYO and SY1 Inputs	These 2 pins can be connected to trace external signals. SY0 may also be used to trigger breakpoints.
GND	Ground Wire	This point should be connected to the target system GROUND before plugging the pod board into the target system.
M	Monitor LED	Lights RED when in the Monitor mode.
E	Emulation LED	Lights GREEN when in Emulation mode.
P	Power-down LED	Lights ORANGE when the CPU is in Power-down mode.
I	Idle LED	Lights ORANGE when the CPU is in the Idle mode.
R	Reset LED	Lights RED when the CPU receives a RESET signal.

Software Notes: EMUL51-PC Software must be Version 5.44 or higher. The PROM on the emulator board must be COM 1.2 or higher.

The POD-C51B must be invoked with -p8051b or -p80652b. Examples:

```
EMUL51 -p8051b -e110 -m128 -t100 -f16    ;typical invocation
EMUL51 -p80652b -e110 -m128 -t100 -f16   ;invocation for '652, '654, '662
```

Internal ROM or EPROM must be mapped to the emulator memory. Due to the bondout chip used in the POD-C51B and POD-C552B-PGA there is a quirk in the operation of the serial port. If you hit a breakpoint in the middle of a serial transmission, the character will be cut off. After resuming from the breakpoint, the flag telling that a new character can be sent will be inhibited, as well as the corresponding interrupt, if interrupts are used.

POD-C51B, POD-C652B

Timers are normally stopped at a breakpoint, and restarted when emulation is resumed. There are currently two exceptions to normal timer operation:

1. Setting a breakpoint at a MOVX instruction causes the timers to miss one tick. 2. When emulation is STARTED in TARGET, the timers will increment one extra tick at the next breakpoint. However, if the next break is at a MOVX instruction the two problems cancel and the timers will be correct.

POD-C652B: Same board as POD-C51B, but the "bond-out" chip is used in a different mode. Change jumpers to choose 83C652 / 80C652 emulation (see Table 5). To invoke:

```
EMUL51 -p80652b -e110 -m128 -t100 -f16     ;typical invocation
```

POD-C851B: Same board as POD-C51B, but the "bond-out" chip is placed in again another mode of operation.

The 83C851 has EEprom onboard the controller. Configure the pod for this application (see Table 5). This pod must be invoked the same as the POD-C51B.

Note: To suppport the 8XC851, the bond-out chip on the pod board must be an 85C582. If you need to support the 8XC851 chip and you currently do not have the proper chip on the pod, please contact NOHAU for an exchange of the pod board.

All of the other information is the same as above.

POD-C51B, POD-C652B

Board Layout Figure 12 shows locations of jumpers and Figure 13 shows the jumper pins and LED's in detail.

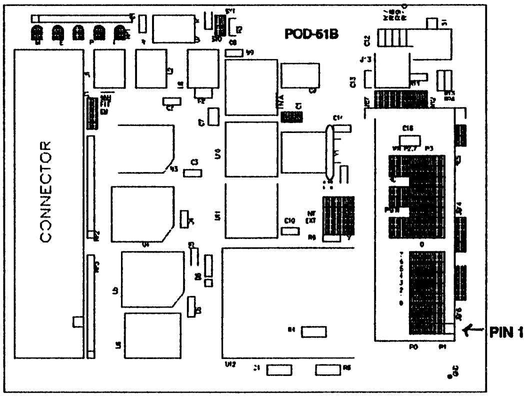
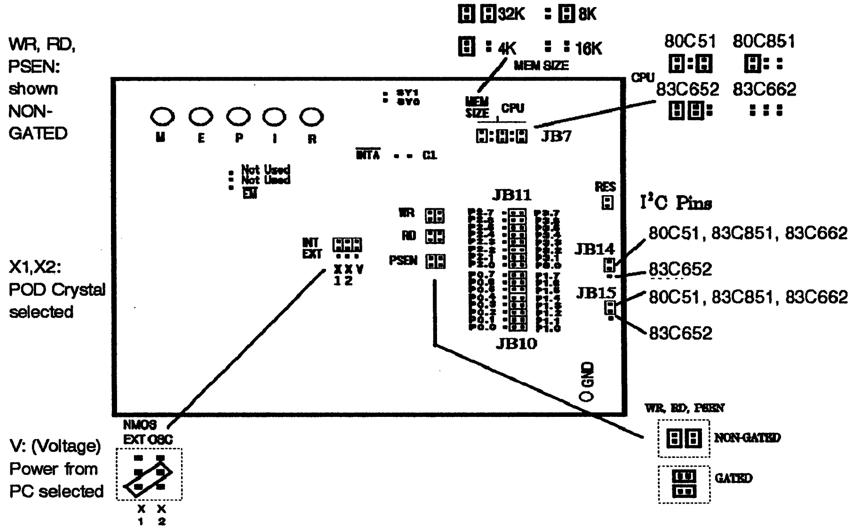


Figure 12. Locations of Jumpers on POD-C51B / POD-C652B

JB7: MEM SIZE selected = 4K; CPU selected = 80C51 RES: External Reset Enable selected



JB10, JB11: TRACE Ports 1 and 3 (all bits) shown JB14, JB15: 80C51 / 80C662 / 80C851 selected

Figure 13. POD-51B / POD-C652B

POD-C51FX *POD TYPE: "HOOKS" MODE*

Also, higher frequency versions.

The software must be 5.6F or later. Invoke the EMUL51-PC as in this example (please check Installation Chapter for option details):

```
EMUL51 -p8751FC -e110 -m128 -t100 -f16  ; typical POD-C51FX invocation
```

The POD-C51FX adapter is used to emulate 8xC51FA, 8xC51FB, and 8xC51FC. It will also support 80C51, 80C31, 87C51, 80C52, 80C32, and 87C52.

The EPROM on the emulator board must be version COM 1.4 or later. If it is older, replace it with an EPROM enclosed with the pod.

The solder side of the board has a 40 pin DIP plug. The plug goes into the target system replacing the microcontroller IC.

If a 44 pin PLCC is used, an adapter can be supplied.

The pod boards have a special version of the Intel 87C51FC chip using a special emulation mode. This emulation mode also uses a programmable gate array from XILINX (LCA).

The pod board has five LEDs named MON, EMUL, PWD, IDLE, and RES:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

PWD is yellow, and means that the processor is in power down mode.

POD-C51FX

IDLE is yellow, and means that the microcontroller is in idle mode.

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RES is red, and means that the processor RESET input is high (ON).

JP5 determines the mode of operation of P0, P2, P3.6 and P3.7. If only internal memory is to be emulated, both jumper tops (M1 and M2) should be in. This enables P0, P2, P3.6 and P3.7 to be used as general purpose ports.

If the lower jumper (M1) is out, P0 will have address and data and P2 will have address or Port 2 data for external code fetches. PSEN will be active. The emulator mapping (MAPC) will determine whether code is read from the emulator or from the target system.

If the upper jumper (M2) is out, P0 and P2 will have address and data for MOVX instructions. P3.6 and P3.7 will be active as RD and WR if data (MAPX) is mapped to the target system.

NOTE:
The mode jumpers M1, M2 can be changed at any time but you have to issue the command RES CHIP so that the LCA can properly switch operational modes.

The board has three jumpers for power and crystal connection to the bond out chip:

Jumper PWR, JP6, is used for power to the processor. Power should be supplied from the emulator and the jumper should be across the two left pins (INT). Taking power from the target system, with the jumper across the two right pins (EXT) is NOT RECOMMENDED. Power should be on the "internal" position, INT, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout PODs.) The processor should be powered from the emulator since its operation is closely connected with the LCA.

POD-C51FX

Jumpers XTAL, JP3 and JP4 determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should

be in the right two positions for the crystal on the pod board (INT), and the left two positions for target supplied crystal or clock (EXT). The oscillator is on the LCA, so do NOT drive the clock inputs when the emulator is powered off.

If you use external clock, note that XTAL1 (pin 19) is input, and XTAL2 (pin 18) should be left open.

EXTERNAL NMOS DRIVE To accommodate external driving OSC signal (not a crystal) designed for NMOS parts, you can do either of the following: Change the target wiring so that XTAL1 (pin 19) is the input and XTAL2 (pin 18) is open; or, you can cross-connect the XTAL jumpers to accomplish the same result. See the "NMOS external OSC connection" illustration in Figure 15. If you use a jumper, you will have to bend the pins slightly. The jumper block is 0.1 inches but the diagonal spacing between the pins is 0.14 inches. Alternatively, you can wire wrap a connection between the pins. Remember to not drive the clock input when the emulator is powered off.

All jumpers for PWR and XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

JP7 (RST) connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulation and JP7 should be removed.

JP2 is used to connect external signals to the emulator. They are used for the trace function. The right pin, SY0, can also be used in the breakpoint logic. The name of the left pin is SY1. The silk screen may be wrong on early boards.

JP1 carries signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The other pins are not used currently, but may be used in the future.

POD-C51FX

Jumper P1/PO upper pins carry the processor port 1 pins in order with P1.0 on the left, and P1.7 on the right. Port 1 is connected directly to the processor. The lower pins carry the port 0 with P0.0 on the left and P0.7 on the right. Port 0 is emulated in the LCA. This means that PO has better sink capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P1/PO is normally connected to P1 using jumper tops. The signal level on the middle pins will be reflected on the trace display. The board is delivered with jumpers, so that P1 will be traced. External signals can be connected to the middle pins if the jumpers to P1 are removed. The input load is one ALS input (74ALS258).

Jumper P3/P2 upper pins carry the processor port 3 pins in order with P3.0 on the left, and P3.7 on the right. Port 3 is connected directly to the processor, except for P3.6 and P3.7 that go through a 74HC4066 that has an approximately 100 ohm series resistance. The lower pins carry the port 2 with P2.0 on the left and P2.7 on the right. Port 2 is emulated in the LCA. This means that P2 has better sink and source capability than a processor. Output signals also appear two clock cycles later than on a real part.

The middle row of jumper P3/P2 is normally connected to P3 using jumper tops. It can be used for P2 or other external signals the same way as P1/PO. The input load is one ALS input (74ALS258).

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Note:

Ports PO, and P2 are emulated, and have slightly different AC and DC characteristics:

> Emulated I/O pins sink minimum 4 mA, and source (P2) minimum 4 mA for two clock cycles after an output low to high transition. As high outputs or inputs, P2 ports have a 22 kohm pull-up resistor.
>
> Emulated PO outputs will change state three clock cycles later than a normal output.

POD-C51FX

Emulated P2 outputs will change state two clock cycles later than anormal output.

If your emulator board only has 32k RAM, data (MAPX) should normally be mapped to target.

P3.6 and P3.7 go through a 74HC4066 which adds about 100 ohms to their output impedance.

When accessing target memory, P2, P0, and ALE will be delayed approximately 25 ns.

The clock circuitry is emulated, but is very close to the processor clock.

The silk screen may be reversed for SY0 and SY1. SY1 is on the left.

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

If you break emulation using the 'Esc' key, and IDLE or PWD is lit, a number of "timeout error" messages will be displayed. This is normal, and you must issue the command RESET CHIP before you do anything else.

The processor on the board is a special version of 87C51FC and can NOT be replaced by a standard 87C51FC. If you think the processor has been damaged, the POD has to be sent to NOHAU for repair. A replacement POD is usually available.

If the POD is used to emulate other processors that 87C51FC, note that all the 87C51FC special function registers still are active, and you will have to be careful not use functions that are not available in the processor you want to emulate.

POD-C51FX

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-C51FX

Board Layout Figure 14 shows locations of jumpers and Figure 15 shows the jumper pins in detail.

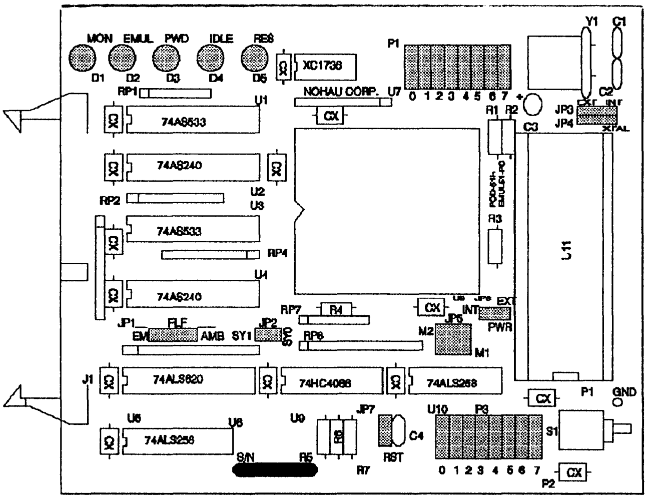


Figure 14. Locations of Jumpers on POD-C51FX

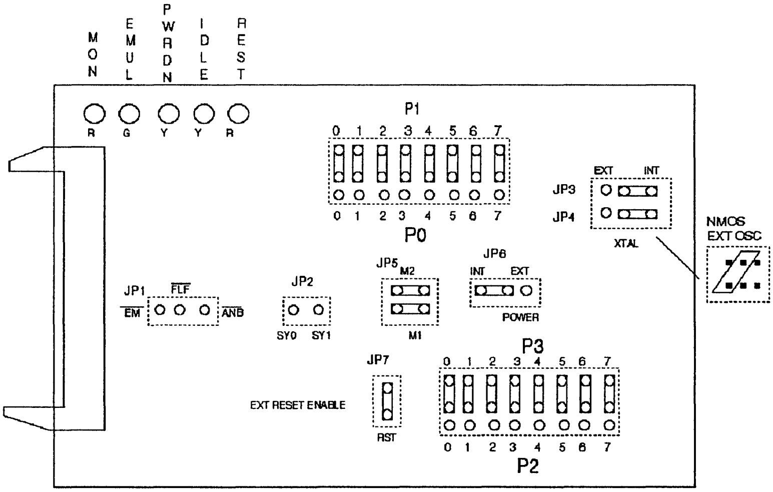


Figure 15. POD-C51FX Jumper Diagram

POD-C51GB-PGA *POD TYPE: EXTERNAL MODE*

Please refer to Figure 17 for the physical placement of the jumpers.

Emulator EPROM should be version 1.31 only.

Table 6. Designators for POD-C51GB-PGA

Designator	*Name*	*Description*
JP2	(RES)	RES jumper is used to connect the processors reset pin to the target system. When installed the reset line is connected to the target. If removed, the reset is not connected.
JP6	PSEN Signal	This jumper is used to control the operation of the PSEN signal. In the GP position this signal is gated, or controlled by the emulator. In the P position the PSEN signal will always go out to the target system. See Figure 17 for settings.
X, X, V	XTAL, PWR (Voltage)	These three jumpers are used to select the XTAL source and the micro-controllers power source. The two jumpers marked with the (X) are for the crystal and the jumper for the power is marked with the (V). If the jumper is towards the EXT position the source is from TARGET, or in the INT position the source is from the POD. See Figure 17 for settings.

(Table continued next page)

POD-C51GB-PGA

Designator	*Name*	*Description*
SYO, SY1	SY0, SY1	These are two inputs that can be used to qualify a trace condition or to be traced. Also SY0 can be used with the EMULATOR board as a qualifier.
P3 / P5 & P1 / P4	Port Connections	These jumpers are used to connect the controllers ports to the trace board. They can also be used to connect other signals from the target to the trace board by removing the jumper and connecting a wire to the middle pin and then to the target signal you wish to monitor.
S1	RESET	This button is a reset switch for the POD's micro-controller.
ANB/ , EM/		The ANB/ signal is not used. The EM/ signal goes low when the emulator is in full emulation mode, this signal follows the LED labeled (EMUL).
D1	MON	This led is on when the emulator is in monitor mode.
D2	EMUL	Is on when emulation takes place.
D3	RES	Monitors the RESET line.

POD-C51GB-PGA

Board Layout Figure 16 shows locations of jumpers and Figure 17 shows the jumper pins in detail.

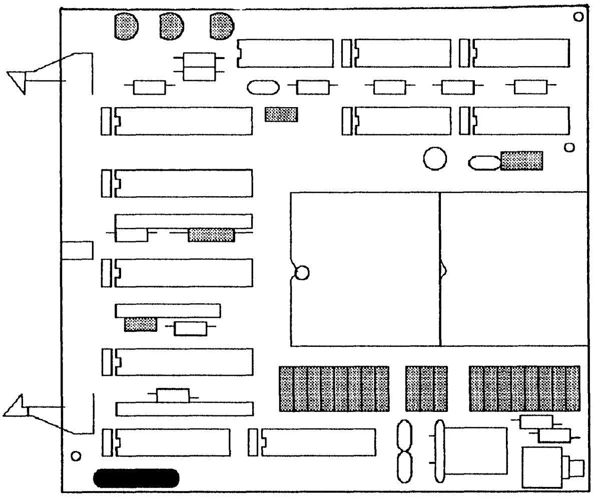


Figure 16. Locations of Jumpers on POD-C51GB-PGA

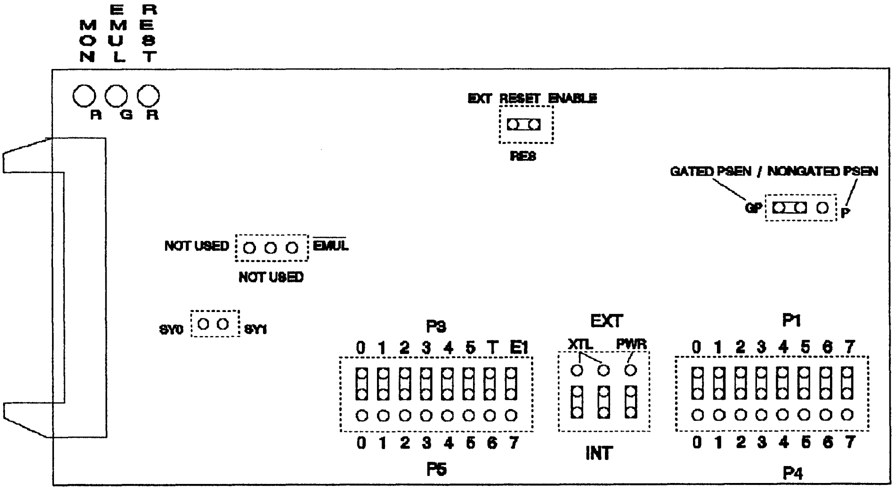


Figure 17. POD-C51GB-PGA Jumper Diagram

POD-C51SL-16 *POD TYPE: EXTERNAL MODE*

The software must be 5.7F or later. Invoke the EMUL51-PC as in this example (please check Installation Chapter for option details):

```
EMUL51 -p8051SL -e110 -m128 -t100 -f16; typical POD-C51SL invocation
```

The POD-C51SL adapter is used to emulate Intel 80C51SL-BG in external memory mode only.

The EPROM on the emulator board must be version COM 1.42. The EPROM should be enclosed with the pod if it is not already on the emulator board. This EPROM is a special version for the POD-C51SL. It will also support other pods except "hooks mode" pods like POD-C51FX.

The solder side of the board has four 26 pin headers. They are designed to plug into the following Emulation Technology adapters:

EP5-100-QF01S	(Surface Mount Pad)
EP5-100-QF01A	(AMP Socket)
EP5-100-QF01T	(3M/Textool Socket)

The Emulation Technology adapters can be ordered from Nohau, or directly from Emulation Technology.

The adapter goes into the target system replacing the microcontroller IC.

The pod boards have a regular Intel 87C51SL-BG chip. This part can be replaced by the user if is gets damaged. The pod also uses a programmable gate array from XILINX (LCA). This part can also be replaced by the user, if it gets damaged.

The pod board has five LEDs, named MON, EMUL, PWRDN, IDLE, and RES:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

POD-C51SL-16

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

PWRDN is yellow, and means that the processor is in power down mode.

IDLE is yellow, and means that the microcontroller is in idle mode.

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RES is red, and means that the processor RESET input is high (ON).

If the jumper M1 is out, P3.6 and P3.7 will be active as RD and WR if data (MAPX) is mapped to the target system. If the jumper is in, P3.6 and P3.7 always go to the target system independent of the mapping.

The board has three jumpers for power and crystal connection to the bond out chip:

Jumper PWR is used for power to the processor. If the power is to be supplied from the emulator, the jumper should be across the two upper pins (INT). If power is to come from the target system, the jumper should be across the two lower pins (EXT). The processor should normally be powered from the emulator, since it is working intimately with the LCA.

Jumpers XTAL determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the upper two positions for the crystal on the pod board (INT), and the lower two positions for target supplied crystal or clock (EXT). The oscillator is on the LCA, so do NOT drive the clock inputs when the emulator is powered off.

All jumpers for PWR and XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

Jumper RST connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulation and RST should be removed.

Jumper pins SY0 and SY1 are used to connect external signals to the emulator. They are used for the trace function. The upper pin, SY0, can also be used in the breakpoint logic. The name of the lower pin is SY1.

The jumper just below U1 carries signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The other pins are used with the Advanced Trace Board (ATR). See the documentation for the ATR.

Rev A pod boards have a jumper below the EM/ jumper marked A1/VCC. The jumper should normally be in the VCC position. However if you are using the A1 input (pin 92) on newer version chips, this jumper should be in the A1 position. Make sure the pod has the newer version parts in this case.

Jumper P1/PCDB upper pins carry the processor port 1 pins in order with P1.0 on the left, and P1.7 on the right. Port 1 is connected directly to the processor. The lower pins carry the port PCDB with PCDB0 on the left and PCDB7 on the right.

The middle row of jumper P1/PCDB is normally connected to P1 using jumper tops. The signal level on the middle pins will be reflected on the trace display. The board is delivered with jumpers, so that P1 will be traced. PCDB0-7 will be traced if the juper tops are moved to the lower pins. External signals can be connected to the middle pins if the jumpers to P1 are removed. The input load is one ALS input (74ALS258).

Jumper P3/misc upper pins carry six of the processor port 3 pins in order with P3.0 on the left. The two upper right pins carry signals LED0 and LED1. Port 3 is connected directly to the processor. The lower pins are connected to processor pins in order from the left: A0, CSL, RDL, WRL, PCOBF, GATE20, RCL, and MEMCSL.

The middle row of jumper P3/misc is normally connected to P3 using jumper tops. It can be used for the signals on the lower row or other external signals the same way as P1/PCDB. The input load is one ALS input (74ALS258).

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Note:
If you use external clock, note that XTAL1 (pin 70) is input, and XTAL2 (pin 69) should be left open.

P3.6 and P3.7 go through a 74HC4066 which adds about 100 ohms to their output impedance.

PSENL goes trough a 74ACT32 to the target system. This may add an extra delay up to 6 ns.

Signals CSL, RDL, WRL, and LOADREN are pulled high by 100 kohm resistors on the pod.

The clock circuitry is emulated, but is very close to the processor clock.

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

If you break emulation using the 'Esc' key, and IDLE or PWD is lit, a number of "timeout error" messages will be displayed. This is normal, and you must issue the command RESET CHIP before you do anything else.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Turn on power to the PC first, and then to the target system.
When turning off power, turn off the target first, then the PC.

POD-C51SL-16

Board Layout Figure 18 shows locations of jumpers and Figure 19 shows the jumper pins in detail.

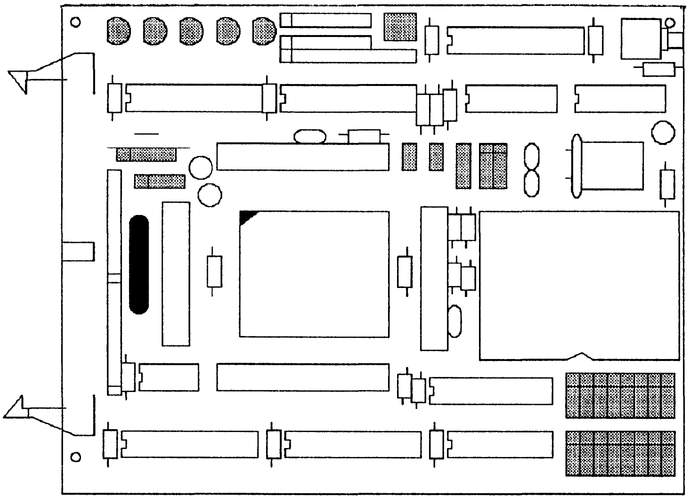


Figure 18. Locations of Jumpers on POD-C51SL-16

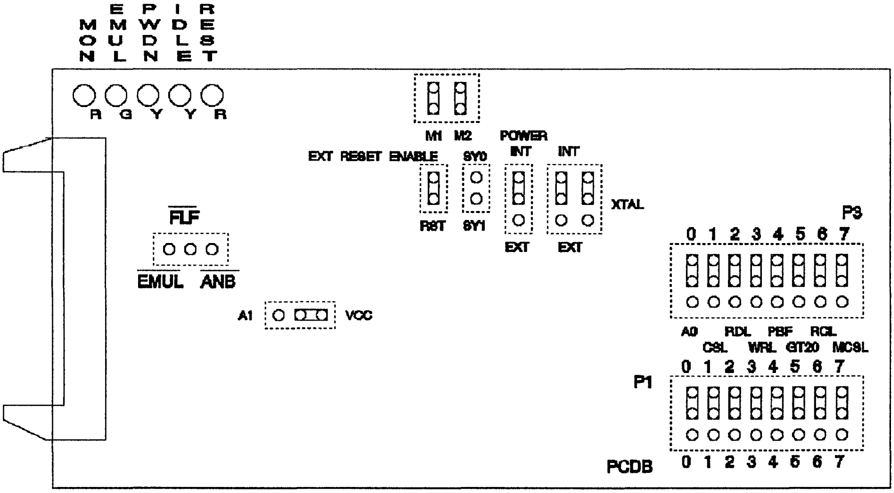


Figure 19. POD-C51SL-16 Jumper Diagram

POD-C52 *POD TYPE: "HOOKS" MODE*

Also higher frequency versions.

The software must be 5.6A or later. Invoke the EMUL51-PC as in this example (please check Installation Chapter for option details):

```
EMUL51 -p8752 -e110 -m128 -t100 -f16   ; typical POD-C52 invocation
```

The POD-52 adapter is used to emulate 80C52, 80C32, and 87C52. It will of course also support 80C51, 80C31, and 87C51.

The EPROM on the emulator board must be version COM 1.4 or later. If it is older replace it with an EPROM enclosed with the pod.

The solder side of the board has a 40 pin DIP plug. The plug goes into the target system replacing the microcontroller IC.

If a 44 pin PLCC is used, an adapter can be supplied.

The pod board has a Signetics 87C52 chip using a special emulation mode. This emulation mode also uses a programmable gate array from XILINX (LCA).

The pod board has five LEDs, named MON, EMUL, PWD, IDLE, and RES:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

PWD is yellow, and means that the processor is in power down mode.

POD-C52

IDLE is yellow, and means that the microcontroller is in idle mode.

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RES is red, and means that the processor RESET input is high (ON).

JP5 determines the mode of operation of P0, P2, P3.6 and P3.7. If only internal memory is to be emulated, both jumper tops (M1 and M2) should be in. This enables P0, P2, P3.6 and P3.7 to be used as general purpose ports.

If the lower jumper (M1) is out, P0 will have address and data and P2 will have address or Port 2 data for external code fetches. PSEN will be active. The emulator mapping (MAPC) will determine whether code is read from the emulator or from the target system.

If the upper jumper (M2) is out, P0 will have address and data and P2 will have address or Port 2 data for MOVX instructions. P3.6 and P3.7 will be active as RD and WR if data (MAPX) is mapped to the target system.

If only internal 83C52 RAM is used and your emulator board has 128k RAM, the data (MAPX) should be mapped to the emulator. M2 should be in.

NOTE:
The mode jumpers M1, M2 can be changed at any time but you have to issue the command RES CHIP so that the LCA can properly switch operational modes.

The board has three jumpers for power and crystal connection to the bond out chip:

Jumper PWR, JP6, is used for power to the processor. Power should be supplied from the emulator and the jumper should be across the two left pins (INT). Taking power from the target system, with the jumper across the two right pins (EXT) is NOT RECOMMENDED. Power should be on the "internal" position, INT, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout PODs.) The processor should be powered from the emulator since its operation is closely connected with the LCA.

Jumpers XTAL, JP3 and JP4 determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the right two positions for the crystal on the pod board (INT), and the left two positions for target supplied crystal or clock (EXT). The oscillator is on the LCA, so do NOT drive the clock inputs when the emulator is powered off.

If you use external clock, note that XTAL1 (pin 19) is input, and XTAL2 (pin 18) should be left open.

EXTERNAL NMOS DRIVE To accommodate external driving OSC signal (not a crystal) designed for NMOS parts, you can do either of the following: Change the target wiring so that STAL1 (pin 19) is the input and XTAL2 (pin 18) is open; or, you can cross-connect the XTAL jumpers to accomplish the same result. See the "NMOS external OSC connection" illustration in Figure 21. If you use a jumper, you will have to bend the pins slightly. The jumper block is 0.1 inches but the diagonal spacing between the pins is 0.14 inches. Alternatively, you can wire wrap a connection between the pins. Remember to not drive the clock input when the emulator is powered off.

All jumpers for PWR and XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

JP7 (RST) connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulation and JP7 should be removed.

JP2 is used to connect external signals to the emulator. They are used for the trace function. The right pin, SY0, can also be used in the breakpoint logic. The name of the left pin is SY1. The silk screen may be wrong on early boards.

POD-C52

JP1 carries signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The other pins are not used currently, but may be used in the future.

Jumper P1/P0 upper pins carry the processor port 1 pins in order with P1.0 on the left, and P1.7 on the right. Port 1 is connected directly to the processor. The lower pins carry the port 0 with P0.0 on the left and P0.7 on the right. Port 0 is emulated in the LCA. This means that P0 has better sink capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P1/P0 is normally connected to P1 using jumper tops. The signal level on the middle pins will be reflected on the trace display. The board is delivered with jumpers, so that P1 will be traced. External signals can be connected to the middle pins if the jumpers to P1 are removed. The input load is one ALS input (74ALS258).

Jumper P3/P2 upper pins carry the processor port 3 pins in order with P3.0 on the left, and P3.7 on the right. Port 3 is connected directly to the processor, except for P3.6 and P3.7 that go through a 74HC4066 that has an approximately 100 ohm series resistance. The lower pins carry the port 2 with P2.0 on the left and P2.7 on the right. Port 2 is emulated in the LCA. This means that P2 has better sink and source capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P3/P2 is normally connected to P3 using jumper tops. It can be used for P2 or other external signals the same way as P1/P0. The input load is one ALS input (74ALS258).

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Note:

Ports P0, and P2 are emulated, and have slightly different AC and DC characteristics:

POD-C52

Emulated I/O pins sink minimum 4 mA, and source (P2) minimum 4 mA for two clock cycles after an output low to high transition. As high outputs or inputs, P2 ports have a 22 kohm pull-up resistor.

Emulated outputs will change state three clock cycles later than a normal output.

If your emulator board only has 32k RAM, data (MAPX) should normally be mapped to target.

P3.6 and P3.7 go through a 74HC4066 which adds about 100 ohms to their output impedance.

When accessing target memory, P2, P0, and ALE will be delayed approximately 25 ns.

The clock circuitry is emulated, but is very close to the processor clock.

The silk screen may be reversed for SY0 and SY1. SY1 is on the left.

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

If you break emulation using the 'Esc' key, and IDLE or PWD is lit, a number of "timeout error" messages will be displayed. This is normal, and you must issue the command RESET CHIP before you do anything else.

If you for some reason want to change processor, you have to use a Philips/Signetics 87C52. The 87C52 has to have lock bit one programmed. Other bits do not affect the emulation mode.

POD-C52

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-C52

Board Layout Figure 20 shows locations of jumpers and Figure 21 shows the jumper pins in detail.

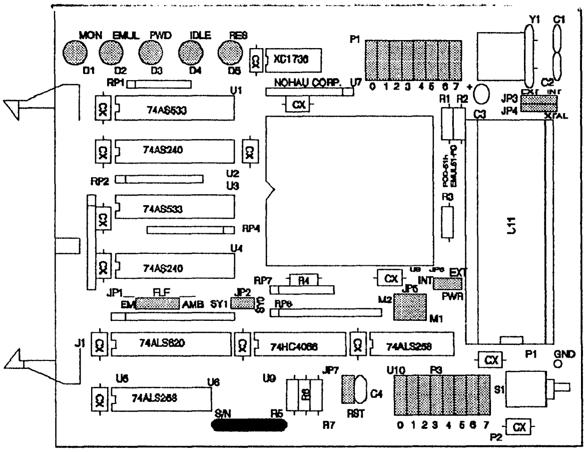


Figure 20. Locations of Jumpers on POD-C52

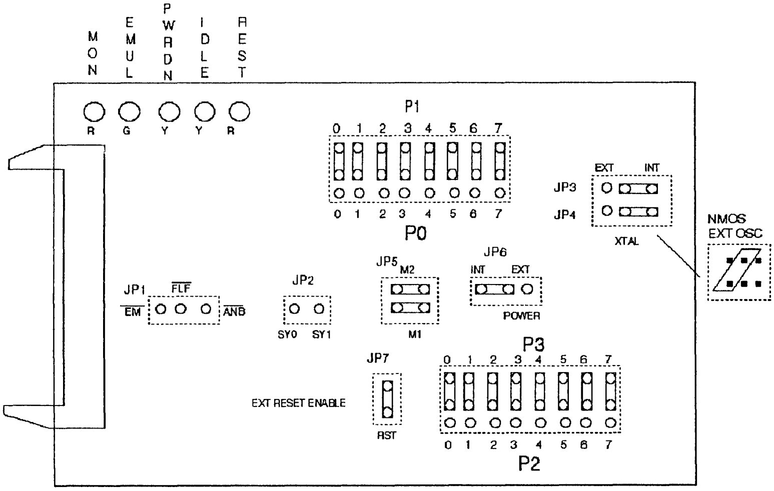


Figure 21. POD-C52 Jumper Diagram

POD-C054 *POD TYPE: "HOOKS" MODE*

The software must be 5.6B or later. Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

Example:

```
EMUL51 -p87054 -e110 -m128 -t100 -f16     ; typical invocation with 128k
                                           emulator
EMUL51 -p87054 -e110 -m32 -t100 -f16      ; typical invocation with 32k emulator
```

The POD-054 adapter is used to emulate Philips/Signetics microcontrollers 83C053, and 87C054.

The EPROM on the emulator board must be version COM 1.4 or later. If it is older replace it with an EPROM enclosed with the pod.

The solder side of the board has a 42 pin DIP plug. The plug goes into the target system replacing the microcontroller IC.

The pod board uses a Signetics 87C054 chip in a special emulation mode. This emulation mode also uses a programmable gate array from XILINX (LCA).

The pod board has four LEDs, named MON, EMUL, PD, and RST:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a special RAM on the emulator board.

PD is yellow, and means that the processor is in power down mode.

If you break in Power Down mode, you must issue "reset chip" to regain communication with the processor.

POD-C054

RST is red, and means that the processor RESET input is high (ON).

The board has three jumpers for power and crystal connection to the bond out chip:

Jumper PWR is used for power to the processor. Power should be supplied from the emulator and the jumper should be across the two left pins (INT). Taking power from the target system, with the jumper across the two right pins (EXT) is NOT RECOMMENDED. Power should be on the"internal" position, INT, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout PODs.) The processor should be powered from the emulator since its operation is closely connected with the LCA.

Jumpers XTAL determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the left two positions for the crystal on the pod board (INT), and the right two positions for target supplied crystal or clock (EXT). The oscillator is on the LCA, so do NOT drive the clock inputs when the emulator is powered off.

All jumpers for PWR and XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

Jumper pins marked RES connect the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulation and the jumper top should be removed.

SY0 and SY1 are used to connect external signals to the emulator. They are also used for the trace function. The left pin, SY0, can also be used in the breakpoint logic.

Jumper pins EM/, FLF/, and ANB/ carry signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The other pins are not used currently, but may be used in the future.

Jumper P3/P1 upper pins carry the processor port 3 pins in order with P3.0 on the left, and P3.7 on the right. Port 3 is emulated in the LCA. This means that P3 has better sink capability than a processor. P3.0, P3.4, and P3.7 are not directly connected to the processor pins, but go through a level

POD-C054

converter, since these pins may have voltages up to 12 volts. Output signals also appear three clock cycles later than on a real part. The lowerpins carry the port 1 with P1.0 on the left. The lower pins also have the processor signals: BF, VCTRL, HSYNC, and VSYNC. The lower pins are connected directly to the processor.

The middle row of jumper P3/P1 is normally connected to P3 using jumper tops. The signal level on the middle pins will be reflected on the trace display. The board is delivered with jumpers, so that P3 will be traced. External signals can be connected to the middle pins if the jumpers to P3 are removed. The input load is one ALS input (74ALS258). If a P1 pin could be pulled up above 5.5 volts, do NOT connect it to a middle pin.

Jumper P2/P0 upper pins carry the processor port 2 pins in order with P2.0 on the left, and P2.7 on the right. Port 2 is emulated in the LCA. This means that P2 has better sink and source capability than a processor. Output signals also appear three clock cycles later than on a real part. The lower pins carry the port 0 with P0.0 on the left and P0.7 on the right. Port 0 is connected directly to the processor.

The middle row of jumper P2/P0 is normally connected to P2 using jumper tops. It can be used for P0 or other external signals the same way as P3/P1. The input load is one ALS input (74ALS258). If a P0 pin could be pulled up above 5.5 volts, do NOT connect it to a middle pin.

JB1 and JB2 are factory set and should not be changed.

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Note:
Ports P2, and P3 are emulated, and have slightly different AC and DC characteristics:

> Emulated I/O pins P2.4 - P2.7, P3.1 - P3.3 sink minimum 4 mA.
>
> P3.0, P3.4, and P3.7 sink minimum 40 mA (7417).
>
> Emulated outputs will change state three clock cycles later than a normal output.

POD-C054

If you use external clock, note that XTAL1 (pin 31) is input, and XTAL2 (pin 32) should be left open.

The clock circuitry is emulated, but is very close to the processor clock.

The delay from reset low to processor start is longer than in the normal part. The delay is about 80 clock cycles.

The timer/counters are stopped at breakpoints, but counts one step extra when used with internal clock.

A reset during emulation will generate about five "garbage" frames in the trace buffer.

If you break emulation using the 'Esc' key, and PD is lit, a number of "timeout error" messages will be displayed. This is normal, and you must issue the command RESET CHIP before you do anything else.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Plug the board carefully into the target system observing that ground and +5V are in the same orientation as the chip on board.

VERY IMPORTANT: Turn on power to the PC first, and then to the target system.

VERY IMPORTANT: When turning off power, turn off the target first, then the PC.

Note: The 83C053 and 87C054 do NOT support MOVX instructions. The emulator should behave reasonably if you try to execute an illegal instruction, but it is not guaranteed.

POD-C054

Board Layout Figure 22 shows locations of jumpers and Figure 23 shows the jumper pins in detail.

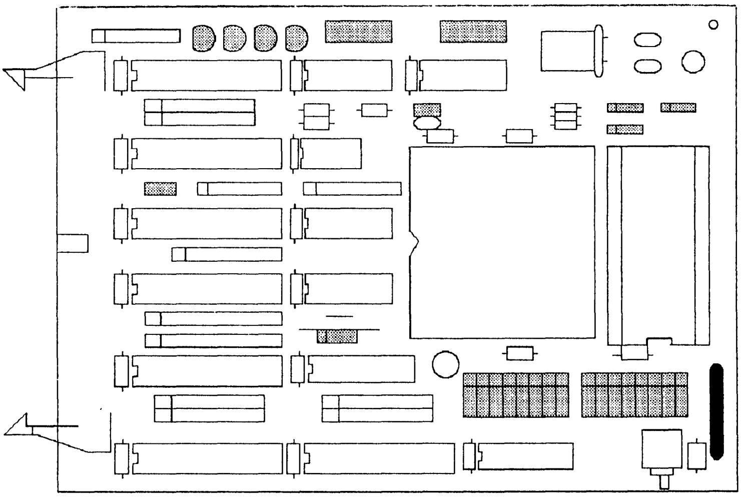


Figure 22. Locations of Jumpers on POD-C054

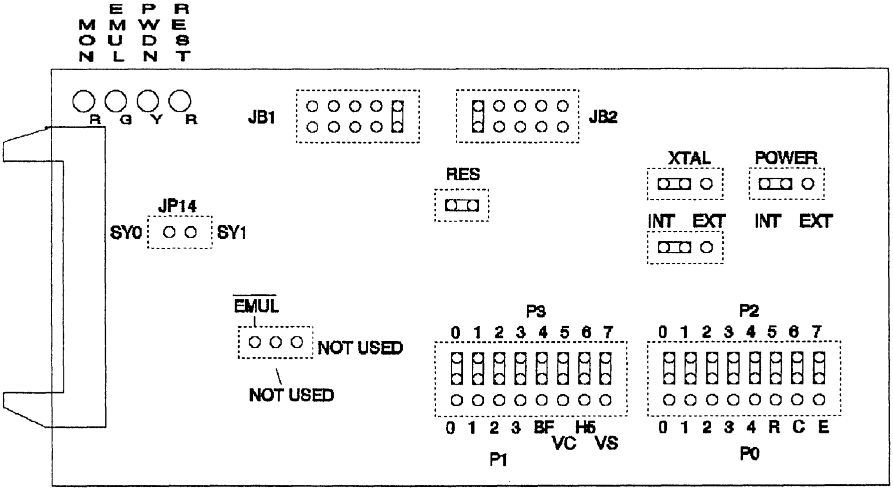


Figure 23. POD-C054 Jumper Diagram

POD-C152-DIP *POD TYPE: EXTERNAL MODE*

Also higher frequency versions.

This pod comes with a 48 pin DIP 80C152 microcontroller from INTEL. Supports Multiprotocol Serial Communication SDLC, HDLC, CSMA/CD and user defined protocols, (1.5 Mbps/2.4 Mbps Max). Dual on-chip DMA channels.

The POD-C152-DIP works up to 12 MHz; POD-C152-DIP-16 up to 16.5 MHz. (On board crystal is 12 MHz). Two jumpers make it possible to use an external crystal or clock. Power to microcontroller chip can be taken either from target system or from the pod. (The pod logic receives its power from the PC).

This pod emulates the 80C152 only in its "external" mode. This means that P0 is used as a data port, P2 as an address port, P3.6 as WR, and P3.7 as RD.

You may use this POD with a standard emulator board. But for applications which use the DMA feature of the chip, you should use a "DMA-modified" option of the emulator board. This DMA option will also work with non-DMA applications and PODs.

With very few exceptions, the emulator does not take resources away from the emulated chip. But the DMA-modified emulator board and software use two internal 80C152 RAM bytes. The default addresses for the emulator to save DCON0 and DCON1 are 7E and 7F. If you need to have the emulator use different addresses other than 7E and 7F, you can use the NB command to change the addresses. The NB command (Nohau Byte) is similar to a debug command and allows modification of bytes in the emulator program.

In the following example, the addresses get changed to 2A and 2B. Note that the two addresses must be consecutive and that the first data address must be input twice. If these values are not suitable, you can select others which would fit with your application.

Example:

```
        NB  E090 = 2A
        NB  FF06 = 2A
        NB  FF0A = 2B
```

These values will get set back to the default 7E and 7F each time you start the emulator software from DOS, or if you use a RESet EMulator command. However, you could put the commands in an include file to have them be set automatically each time you start the emulator software.

POD-C152-DIP

Table 6. Designators for POD-C152-DIP

Designator	*Name*	*Description*
JB1	RST Selection	JB1 connects the target RESET to the POD CPU. With JB1 installed the signal on J3 pin 9 will be connected to RESET on POD CPU. With no jumper installed the signal will not reach the POD CPU. (This is useful if you have a watchdog on your target system).
JB2	CHIP POWER	This jumper (or switch) selects the POD CPU power source. If the slide SWITCH is switched toward the latching connector (or if the JUMPER is installed toward the latching connector), the CPU is powered from the POD. If the slide switch is switched toward the reset push button (or if the JUMPER is installed toward the reset push button), the CPU is powered from the target. On some boards, the E and I silkscreen legends are incorrectly swapped, but the description above is correct.
JB3	PORT 1	(8 jumpers). These jumpers connect the TRACE to port 1. If you remove a jumper you can connect an external signal to the pin away from the 80C152, and have this signal traced.
JB4	PORT 3 and 4	(6 jumpers). These jumpers connect either PORT 3 or PORT 4 to the TRACE. If a jumper is installed away from the 80C152, the trace will be connected to PORT 4. With a jumper installed toward the 80C152 (as shown in Figure 25), PORT 3 will be traced.
JB5	PORT 4	(2 jumpers). This is PORT 4 bit 6 and bit 7. If a jumper is installed you are tracing PORT 4, and if no jumper is installed you can connect an external signal to be traced.

(Table continued next page)

POD-C152-DIP

Designator	*Name*	*Description*
JB6	CRYSTAL	With jumpers installed as shown, the pod crystal will be connected to the POD CPU. With jumpers installed in the opposite position, the target system's crystal or osc will be connected to the POD CPU. NOTE: Rev. B boards have a different configuration; see the separate drawing included with these boards.
JB7	PSEN Signal	If the slide SWITCH is switched away from the edge of the board (or if the JUMPER is installed toward the edge of the board), then PSEN only goes out to the target system when target CODE memory is accessed. If the slide SWITCH is switched toward the edge of the board (or if the JUMPER is installed away from the edge of the board), then PSEN will always go out to the target system, even during monitor mode.
J2	SY0 and SY1	These 2 pins can be connected to trace external signals. SY0 may also be used to trigger breakpoints.
J3	TARGET Con.	This connector fits in your target system's CPU socket.
S1	RESET	By pressing this button you reset the pod CPU. The same thing as entering the command: "RES CHIP".
TP1	TRIGGER Point	This pin is low when the emulator is running. It goes high as soon as a break condition is detected. May be used to trigger a memory oscilloscope.
DS1	EMULATOR	This LED will be lit when emulator is running.
GND	GROUND conn.	Before you plug the POD board into your target system, you should connect this point with your target GND.

POD-C152-DIP

Board Layout Figure 24 shows locations of jumpers and Figure 25 shows a diagram of jumper pins.

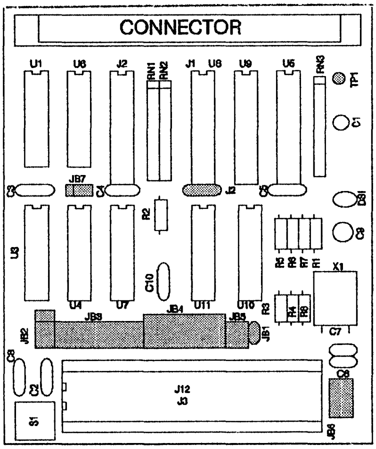
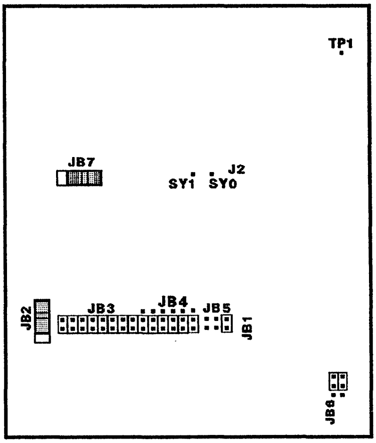


Figure 24. Locations of Jumpers on POD-C152-DIP Board



JB2: Power from POD shown

JB6 Pair: POD crystal shown

Figure 25. POD-C152-DIP Board Configured for internal power and internal crystal

POD-C152-PGA *POD TYPE: EXTERNAL MODE*

Also higher frequency versions.

The PLCC socket on this pod accepts an Intel 80C152 PLCC which supports Multiprotocol Serial Communication SDLC, HDLC, CSMA/CD and user defined protocols (1.5Mbps/2.4Mbps Max), and provides dual on-chip DMA channels.

POD-C152-PGA pod works up to 12 Mhz. **POD-C152-PGA-16** works up to 16.5 Mhz. Power to the microcontroller chip can be taken either from the target system or from the pod. (The pod logic receives its power from the PC).

This pod emulates the 80C152 only in its "external" mode. This means that P0 is used as a data port, P2 as an address port, P3.6 as WR, and P3.7 as RD.

You may use this POD with a standard emulator board. But for applications which use the DMA feature of the chip, you should use a "DMA-modified" option of the emulator board. This DMA option will also work with non-DMA applications and PODs.

With very few exceptions, the emulator does not take resources away from the emulated chip. But the DMA-modified emulator board and software use two internal 80C152 RAM bytes. The default addresses for the emulator to save DCON0 and DCON1 are 7E and 7F. If you need to have the emulator use different addresses other than 7E and 7F, you can use the NB command to change the addresses. The NB command (Nohau Byte) is similar to a debug command and allows modification of bytes in the emulator program.

In the following example, the addresses get changed to 2A and 2B. Note that the two addresses must be consecutive and that the first data address must be input twice. If these values are not suitable, you can select others which would fit with your application.

Example:

```
NB  E090 = 2A
NB  FF06 = 2A
NB  FF0A = 2B
```

These values will get set back to the default 7E and 7F each time you start the emulator software from DOS, or if you use a RESet EMulator command. However, you could put the commands in an include file to have them be set automatically each time you start the emulator software.

POD-C152-PGA

Table 7. Designators for POD-C152-PGA

Designator	*Name*	*Description*
JB1	RESET	Jumper JB1 allows you to disconnect the reset line between you target system and the 80C152. This is useful if there is a watchdog in your target system.
JB2	CPU POWER	This switch selects the source of power for the 80C152. If the switch is toward J3, the 80C152 will receive power from the target system. If the switch is toward the ribbon cable, the 80C152 will get the power from the computer.
JB3	PORT 1	These eight jumpers connect the trace input to either Port 1, or to the signals selected on JB4. If a jumper is removed, an external signal can be connected to the center pin. That external signal will be traced instead of the port pin.
JB4	PORT 5 & 6	These eight jumpers connect either port 5 or port 6 to JB3. If a jumper is installed towards the 80C152, port 6 can be traced (through JB3). Jumpers toward the edge of the board connect port 5.

(Table continued next page)

POD-C152-PGA

Designator	*Name*	*Description*
JB5	PORT 3 & 4	The six jumpers numbered 0 through 5 connect the trace to either port 3 or to port 4. P4.0 through P4.7 are on the eight pins away from the edge of the board. If the jumpers are in that direction, port 4 will be traced. If they are toward the edge of the board, port 3 will be traced. The center pins at positions 6 and 7 are external trace bits E0 and E1. The pins at the edge of the board are EBEN at position 6, and EPSEN/ at position 7. If a jumper is removed, an external signal can be connected to the center pin. This external signal will then be traced instead of the port pin.
JB6	CRYSTAL	If both switches are in position E, toward the microcontroller, your target system's crystal or oscillator will be connected to the 80C152. With both switches in position I, the crystal on the pod will be connected to the 80C152. (Exception: see sketch enclosed with Revision A boards.)
JB7	PSEN/	If this switch is in position 2, toward the ribbon cable, the signal PSEN/ only goes out to target system when coed is being executed from the target, as specified by the "MAPC" settings in the emulator. If the switch is in position 1, PSEN/ will always reach the target system.

(Table continued next page)

POD-C152-PGA

Designator	*Name*	*Description*
J2	SY0 & SY1	These 2 pins can be used to trace external signals. SY0 may also be used to trigger breakpoints.
J3	TARGET	This connector fits into your target system's 80C152 socket.
S1	RESET	Pressing this button resets the 80C152. The reset signal does not go out to the target.
TP1	TRIGGER	This pin is low when the emulator is running and goes high as soon as a break condition is detected. It may be used to trigger an oscilloscope.
DS1	EMULATOR	This LED indicates when the emulation is taking place.
GND	GROUND	Before you plug the pod into your target system, connect this clip to your target system's GND. This is to make sure that there is a good ground connection and to protect the pod and the target from static discharge.

POD-C152-PGA

Board Layout Figure 26 shows locations of jumpers and Figure 27 shows the jumper pins in detail.

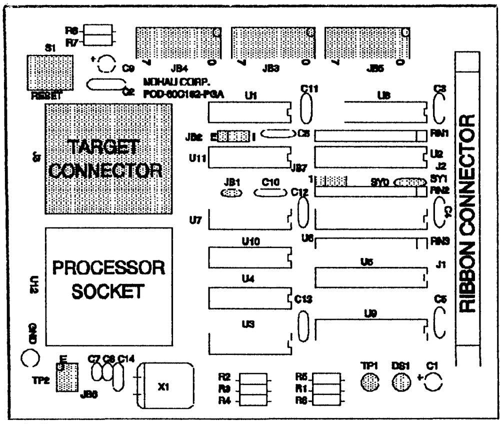


Figure 26. Locations of Jumpers on POD-C152-PGA / POD-C152-PGA-16

JB4: Port 6 selected JB3: TRACE Port 1 selected JB5: TRACE Port 3 selected

JB2: POWER from PC selected

JB7: PSEN/only when code executing selected

JB6: INT Crystal selected

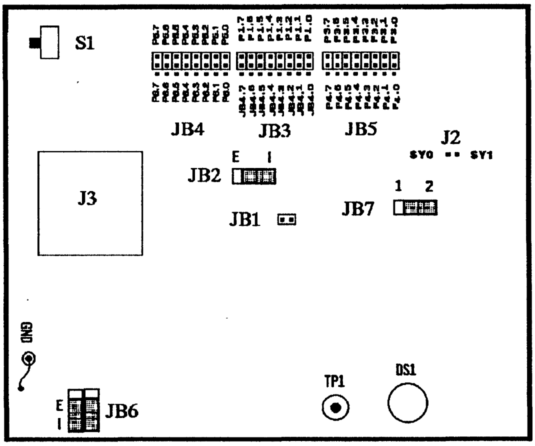


Figure 27. POD-C152-PGA / POD-C152-PGA-16

POD-407-14.75 *POD TYPE: EXTERNAL MODE*

The software must be 5.7Y or later. Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

```
EMUL51 -p407 -e110 -m128 -t100 -f16  ; typical invocation
```

The POD-407 adapter is used to emulate Yamaha GTM407.

The EPROM on the emulator board must be version COM 1.44. The EPROM should be enclosed with the pod if it is not already on the emulator board. This EPROM is a special version for the POD-407. This EPROM will not work well with regular external mode pods since it will alter the contents of port P1.

The solder side of the board has two 16 pin headers and two 24 pin headers. They are designed to plug into this Emulation Technology adapter for surface mount pads: EPP-080-QF08-LG. The pod will also plug into some Yamaha evaluation boards without adapter.

The Emulation Technology adapters can be ordered from Nohau, or directly from Emulation Technology.

The adapter goes into the target system replacing the microcontroller IC.

The pod board has a standard Yamaha GTM407 chip. It can be replaced by the user if it gets damaged.

The pod board has three LEDs, named MON, EMUL, and RESET:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a

POD-407-14.75

special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

RESET is red, and means that the processor PCRESET input is high (ON).

Jumpers XTAL determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the upper two positions for the crystal on the pod board (INT), and the lower two positions for target supplied crystal or clock (EXT).

Jumper POWER is used for power to the processor. If the power is to be supplied from the emulator, the jumper should be across the two left pins (INT). If power is to come from the target system, the jumper should be across the two right pins (EXT).

All jumpers for POWER and XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

Jumper RAM is used to enable and disable the 128 kbyte RAM on the pod board. To enable it the jumper should be across the two right pins (EN). To disable the RAM, the jumper should be across the two left pins (DIS). The RAM must be disabled when the POD is connected to a target system with RAM. The RAM is intended to be used for software testing without a target system.

Jumper RST connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulator in monitor mode and RST should be removed.

Jumper pins SY0 and SY1 are used to connect external signals to the emulator. They are used for the trace function. The left pin, SY0, can also be used in the breakpoint logic.

The top of the board on the right side are two 8x3 pin headers. These are used for the trace and are displayed in the P1 and P3 columns.

The left 8x3 pin header, top row of pins has the following processor signals in order from the left: PCA0, PCA1, PCA2, /PCRD, /PCWR, PCINT, /PCS, and /PCDEN. The bottom row of pins has these signals from the left: OH, /RXRDY, /TXRDY, /MCS, /MINT, /PWRDN, MRSTN, and /PCS. The middle row of pins are reflected on the trace display as P3. The board is delivered with jumpers so that the top row is traced. The jumpers can be moved to the lower row, so that it will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

The right 8x3 pin header, top row of pins has the following processor signals in order from the left: PCD0, PCD1, PCD2, PCD3, PCD4, PCD5, PCD6, and PCD7. The bottom row of pins has these signals from the left: PCA3 (/DSR), PCA4 (/CD), PCA5 (/RTS), PCA6 (/CTS), PCA7 (/TXD), PCA8 (/RXD), PCA9 (/DTR), and /RI. The middle row of pins are reflected on the trace display as P1. The board is delivered with jumpers so that the top row is traced. The jumpers can be moved to the lower row, so that it will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

The jumper pins between U3 and U4 carry signals to and from the emulator and trace boards. The right pin EM/ is high in monitor mode, and low in emulation mode. The middle pin FLF/ and the right pin ANB/ and are used with the Advanced Trace Board (ATR). See the documentation for the ATR.

S1 push-button is used to reset the processor, and can be used instead of target system reset.

If you want to use the built in bankswitching in the GTM407 to access more than 64 k code, you need to have an emulator board with a special modification for the POD-407. In this case the XDATA has to be mapped to target.

The bankswitch bits P1.5, P1.6, and P1.7 are saved at breakpoint and can be displayed and changed by accessing a byte in a special RAM area in the emulator. The command is NB, and the location is E510. This command works just like the CB command (see Commands Chapter) but in a reserved memory area of the emulator. Do not alter any other NB memory

POD-407-14.75

location. Do not try to change the bankswitch bits by using a RB command.

Examples for bankswitching:

```
NB E510              ; Displays bankswitch bits as upper three bits.
NB E510 = C0         ; set P1.7 = P1.6 = 1, P1.5 = 0
NB E510 = 60         ; set P1.7 = 0, P1.6 = P1.5 = 1
```

This can be used to load and display program in different banks of the emulator board.

GTM407 Code Bankswitch Logic

A15int	P1.7	P1.6	A15ext	A16
0	X	X	0	0
1	1	1	1	0
1	1	0	1	1
1	0	1	0	1
1	0	0	0	0

Note:
The signals /ECS, /MRD, and /MWR go through a 74ACT32 to the target system. This will add an extra delay up to 9 ns.

SY0, and SY1 have each a 100 kohm pullup resistor.

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

The current version of GTM407 turns off interrupts a little slower than a standard 8051 part. This means that you can get interrupts just after instructions like: JBC EA,adr and ANL IE,#07Fh. This creates a problem in the emulator if there is an interrupt at about the same time as a breakpoint. In this case the emulator may accidentally turn off the EA bit. If the program counter is in the very beginning of an interrupt routine at breakpoint, check the IE register, and make sure the EA (MSB) is set.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-407-14.75

Board Layout Figure 28 shows locations of jumpers and Figure 29 shows the jumper pins in detail.

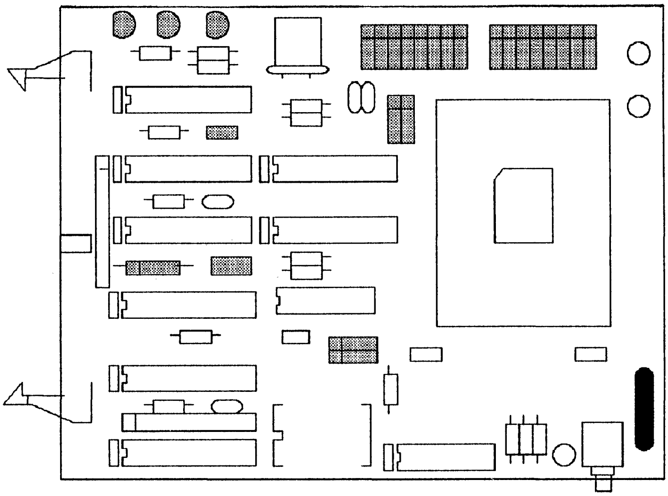


Figure 28. Locations of Jumpers on POD-407-14.75

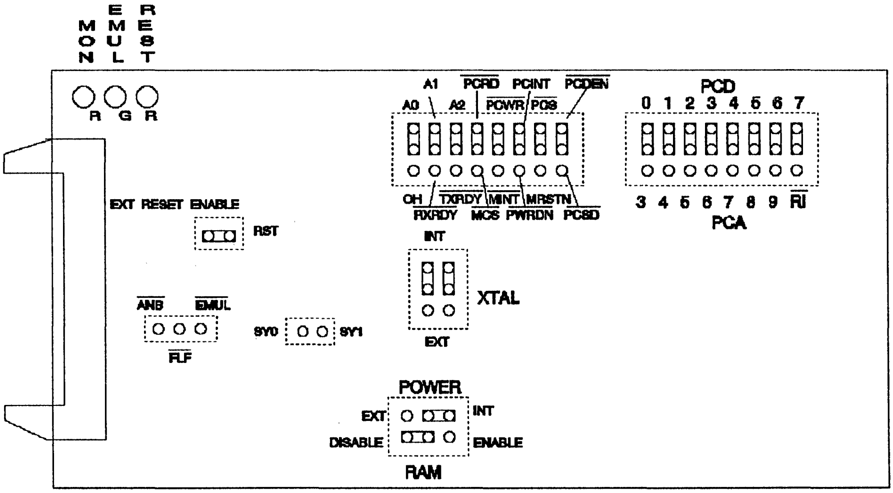


Figure 29. POD-407-14.75 Jumper Diagram

POD-CL410 *POD TYPE: BONDOUT*

Software must be 5.5R or later. Invoke the EMUL51-PC like in these examples (please see Installation Chapter for option details):

```
EMUL51 -p83410 -e110 -m128 -t100 -f16        ; typical invocation
```

The pod board is used to emulate Philips/Signetics' PCF83CL410, PCF83CL610, 80CL51 and 80CL31. To emulate the 80CL51 and 80CL31, use external pullups on P1.6 and P1.7. Use 3 kohms or higher, depending on signal current requirements.

This chip contains registers for I^{2}C operation to emulate the 'CL410 and 'CL610. It contains normal UART SBUF registers for 'CL31 'CL51 emulation. It also contains 256 bytes of internal RAM. However, since it does not contain the third timer, it is not intended to emulate the 'CL32 and 'CL52.

The EPROM on the emulator board must be version COM 1.2 or later. If it is older replace it with the EPROM enclosed with the pod.

The bottom side of the board has a 40 pin DIP plug. The pod board goes into the target system, replacing the microcontroller IC.

The pod board has a Philips PCF86C001WP bondout chip to emulate the microcontroller chip.

The pod board has five LEDs marked MON, EMUL, PWRDN, IDLE, and RES:

MON is red, and if lit means that the system is in monitor mode. In monitor mode, the bondout chip is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and if lit means that the system is in emulation mode. In emulation mode the bondout chip is executing user code. This code is normally in a special RAM on the emulator board.

POD-CL410

PWRDN is orange, and if lit means that the system is in power down mode. An interrupt or reset can take it out of this mode.

IDLE is orange, and if lit means that the system is in idle mode. An interrupt or reset can take it out of this mode.

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RES is red, and if lit means that the bondout chip RESET input is high (ON).

The jumper marked PWR close to the pushbutton is used for power to the bondout chip. If the power is to be supplied from the emulator (5 volts), the jumper should be across the two left pins (INT). If power is to come from the target system (1.5 - 5 volts), the jumper should be across the two right pins (EXT).

The jumpers marked XTAL close to the crystal determine if the crystal or clock is from the target system, or if the crystal on the pod is used. The jumper should be in the upper two positions for the crystal on the pod board (INT), and the lower two positions for target supplied crystal or clock (EXT).

The maximum frequency is 12 MHz at 5 volts. If you get timeout errors due to slower oscillator speeds, especially if the emulator is in a fast computer, use a -d delay option on startup. See Parameter Options in the Installation Chapter.

The jumpers above must be in internal (INT) positions when no target system is connected. This is also the default position.

The jumper marked RES int the middle of the board connects the target RESET pin to the emulator. If the target system has a watch dog, it will probably interfere with monitor mode and the jumper should be removed.

The jumpers marked SY0 and SY1 close to the ribbon cable connector are used to connect external signals to the emulator. They are used for the trace function. The left pin, SY0, can also be used in the breakpoint logic.

POD-CL410

The name of the right pin is SY1.

In the middle of the board and towards the left are three jumper pins: The left pin EM is high in monitor mode, and low in emulation mode. The other pins FLF, and ANB are not used currently, but may be used in the future.

For jumpers P2/P3: the bottom row of pins is connected to port P3, in order, with P3.7 at the left and P3.0 at the right. The top row of pins is connected to port P2, with P2.7 at the left and P2.0 at the right. The middle row of pins is reflected on the trace display of P3. The board is delivered with jumpers so that P3 is traced. If the jumpers are moved to the top row, P2 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

For jumpers P0/P1: the bottom row of pins is connected to port P1, in order, with P1.7 at the left and P1.0 at the right. The top row of pins is connected to port P0, with P0.7 at the left and P0.0 at the right. The middle row of pins is reflected on the trace display of P1. The board is delivered with jumpers so that P1 is traced. If the jumpers are moved to the top row, P2 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumper INTA/C1 just below the bondout chip has the bondout chip signals INTA (interrupt acknowledge) and C1 (first op code fetch).

EXTERNAL INTERRUPTS 2 - 9

These interrupts have a polarity register, IX1. This register is initialized to 0's (active low) on reset. Due to the design of the bondout chip, your code must write to this register before these interrupts will work. This is true even if your code does not need to change IX1 from its reset value (0's). This additional write is not necessary on the PCF83CL410 or PCF83CL610 chips. However, this extra write will work in both the case of the emulator and the final chips.

If you want to trace C1 or INTA the corresponding pins must be connected to the "middle row" of P2/P3 jumper rows thereby replacing those ports in

POD-CL410

the trace display.

JB1 close to the crystal is used to set the amount of internal memory emulated. It is also used to emulate the "ROM PROTECT" feature. See Figure 30 and Figure 31.

S1 pushbutton is used to reset the bondout chip, and can be used instead of the target system reset.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC, and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Plug the board carefully into the target system. With the pod board oriented so the long ribbon cable goes out to the left. Make sure that the target system pin 1 is towards the bottom of the board.

Turn on power to the PC first, and then to the target system. When turning power off, turn the target off first, then the PC.

Note:
If the target system is running at a voltage lower than 5 volts, the power jumper should be in the EXT position. In this situation the bondout chip will be powered from the target system. Since the voltage level translation circuitry uses some power from the bondout chip signal lines, the current consumption will be higher than for a "real" chip. The current may be up to 10 mA.

Since the PCF83CL410 processor is mainly intended for ROM code with P0 and P2 used as port pins, and because of the amount of circuitry involved for voltage level translation, the emulation support for external memory access is limited.

MOVX instructions are not fully supported. XDATA can not be mapped to the emulator. Do not use breakpoints on MOVX read or write cycles. The address and data for MOVX instructions is not correct in the trace display.

CODE can be executed in the target system, but breakpoints will not work properly. The trace will show addresses correct, but the actual code bytes read from the target system will be incorrect. So CODE should normally be mapped to the emulator memory.

POD-CL410

Board Layout Figure 30 shows locations of jumpers and Figure 31 shows a diagram of the jumper pins.

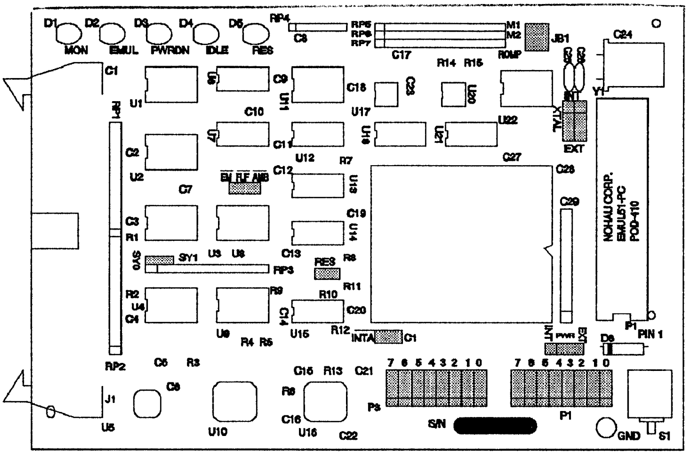


Figure 30. Location of Jumpers on POD-CL410

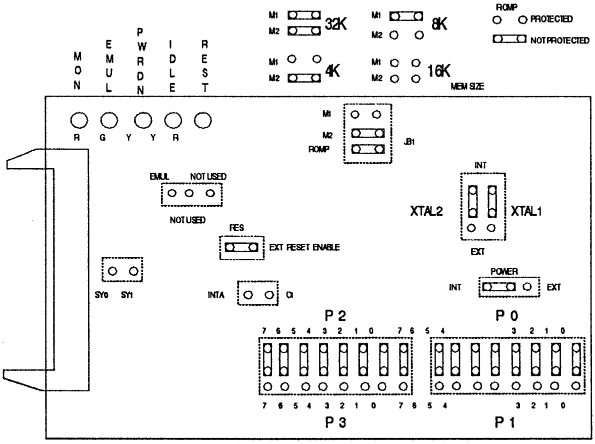


Figure 31. POD-CL410 Jumper Diagram

POD-C451-DIP *POD TYPE: EXTERNAL MODE*

The DIP socket on this pod accepts a Signetics 80C451 microcontroller with seven 8-bit I/O ports. Centronics compatible printer interface using port 6. Microprocessor bus compatible mailbox on port 6 supports peripheral processing for 8086 and 68000 type processors. This pod works up to 12 MHz. Two jumpers make it possible to use an external crystal or clock. Power to the microcontroller chip can be taken either from the target system or from the pod. (The pod receives its power from the PC.) This pod emulates the 80C451 only in its "external" mode. This means that P0 is used as a data port, P2 is used as an address port, P3.6 is used as WR, and P3.7 is used as RD.

Table 8. Designators for POD-C451-DIP

Designator	*Name*	*Description*
JB1	PSEN Signal	In the factory installed position, the PSEN signal only goes out on the target system in emulation mode. In the alternate position, PSEN will always go out to the target system. In that case be sure that PSEN is not used to open anything that drives the databus. (Remove PROMs).
JB2	RST Selection	With JB2 installed, pin 31 on the POD board microprocessor (RST) is connected to the target system, which allows reset to be performed by S1 or the target system.

(Table continued next page)

POD-C451-DIP

Designator	*Name*	*Description*
		With JB2 removed, RST is connected only to the S1 switch and the PC controlled reset line. This is useful if you have a software controlled watchdog, that keeps resetting the microprocessor.
JB3-JB6	Crystal Selection	With JB4 and JB6 jumpers installed, the crystal on the POD board is connected to the micro processor. When the two jumpers are installed in JB3 and JB5, the crystal of the target system is connected to the microprocessor.
JB7-JB8	POD Board Power	This jumper determines where the POD board gets its power. With the jumper in JB7 position (see figure), the power is received from the PC. If the jumper is changed to the JB8 position, power is taken from the target system.
P1, P3, P4, P5, P6	Port Connections	Ports P1.0 through P1.7 and P3.0 through P3.5 are connected to the trace board when the jumpers are in their preset positions. If the jumpers are removed the middle pins may be connected to external signals with the optional EZ-hooks. These signals will then be traced instead of the ports. If you want to trace P5 you will have to connect P5.0 through P5.7 to the middle pins with jumper wires

(Table continued next page)

POD-C451-DIP

Designator	*Name*	*Description*
		(EZ-hooks). If the jumpers are placed in the opposite position, ports P6 and P4 will be traced.
SY0, SY1 E0, E1 IDS, ODS	Standby External Signals	The pins close to IDS and ODS are E0 and E1 (unmarked on the board). These four pins may be used to trace external signals. If jumpers are placed on IDS and ODS these two signals will be traced as E0 and E1.
TP1	Trigger	This pin is low when the emulator is running. It goes high as soon as a break condition is detected. This may be used to trigger a memory oscilloscope.

POD-C451-DIP

Board Layout Figure 32 shows locations of jumpers and Figure 33 shows a diagram of the jumper pins.

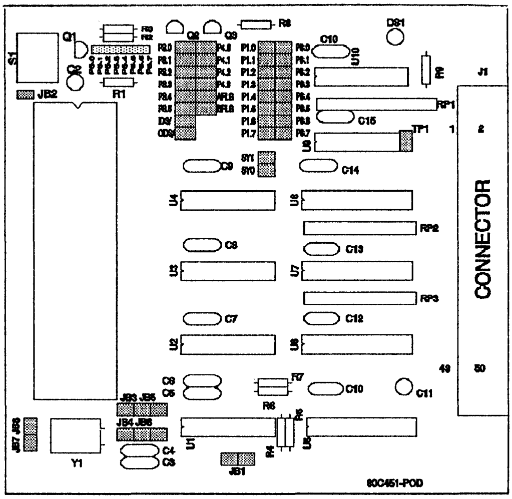
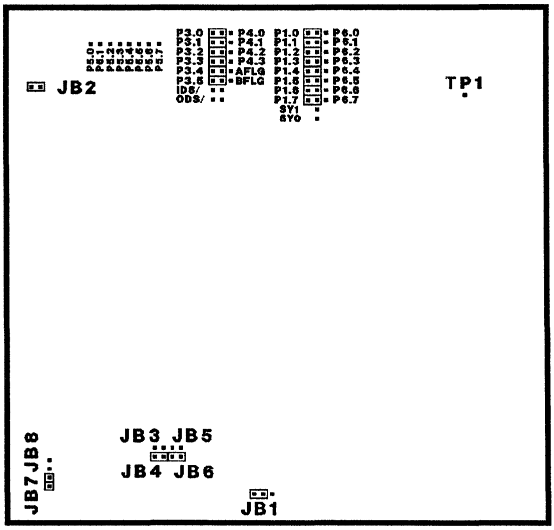


Figure 32. Locations of Jumpers on POD-C451-DIP Board



JB4 and JB6: POD crystal

JB7: Power from POD

Figure 33. POD-C451-DIP Board Configured for target power and internal crystal

POD-C451-PGA *POD TYPE: EXTERNAL MODE*

The PGA socket on this pod accepts a Signetics 80C451 microcontroller with seven 8-bit I/O ports. Centronics compatible printer interface using port 6. Microprocessor bus compatible mailbox on port 6 supports peripheral processing for 8086 and 68000 type processors. This pod works up to 12 MHz. Two jumpers make it possible to use an external crystal or clock. Power to the microcontroller chip can be taken either from the target system or from the pod. (The pod receives its power from the PC.) This pod emulates the 80C451 only in its "external" mode. This means that P0 is used as a data port, P2 is used as an address port, P3.6 is used as WR, and P3.7 is used as RD.

Table 9. Designators for POD-C451-PGA

Designator	*Name*	*Description*
JB1	RST Selection	With JB1 installed, pin 35 on the POD board microprocessor (RST) is connected to the target system, which allows reset to be performed by S1 or the target system. With JB1 removed, RST is connected only to the S1 switch and the PC controlled reset line. This is useful if you have a software controlled watchdog, that keeps resetting the microprocessor.
JB2, JB4	POD Board Power	This jumper determines where the POD board gets its power. With the jumper in JB4 position (see figure),

(Table continued next page)

POD-C451-PGA

Designator	*Name*	*Description*
		the power is received from the PC. If the jumper is changed to the JB2 position, power is taken from the target system.
P1, P3 P4, P5 P6	Port Connections	Ports P1.0 through P1.7 and P3.0 through P3.5 are connected to the trace board when the jumpers are in their preset positions. If the jumpers are removed the middle pins may be connected to external signals with the optional EZ-hooks. These signals will then be traced instead of the ports. If you want to trace P4.4 - P4.7 or P5 you will have to connect these pins to the middle pins with jumper wires (EZ-hooks). If the jumpers are placed in the opposite position, ports P6 and P4 will be traced.
SY0, SY1 E0, E1 IDS, ODS	Standby External Signals	The pins close to IDS and ODS are E0 and E1 (unmarked on the board). These four pins may be used to trace external signals. If jumpers are placed on IDS and ODS these two signals will be traced as E0 and E1.
JB3, JB5-JB7	Crystal Selection	With JB3 and JB5 jumpers installed, the crystal on the POD board is connected to the micro processor. When the two jumpers are installed in JB6 and JB7, the crystal of the target system is connected to the microprocessor.

(Table continued next page)

POD-C451-PGA

Designator	*Name*	*Description*
JB8	PSEN Signal	In the factory installed position, the PSEN signal only goes out on the target system in emulation mode. In the alternate position, PSEN will always go out to the target system. In that case be sure that PSEN is not used to open anything that drives the databus. (Remove PROMs).
TP1	Trigger	This pin is low when the emulator is running. It goes high as soon as a break condition is detected. This may be used to trigger a memory oscilloscope.

POD-C451-PGA

Board Layout Figure 34 shows locations of jumpers and Figure 35 shows a diagram of the jumper pins.

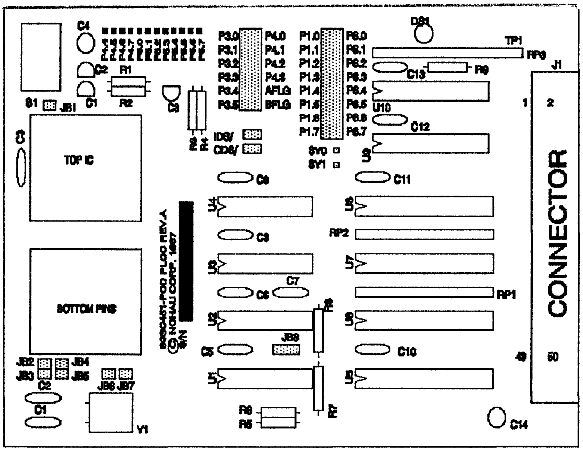


Figure 34. Locations of Jumpers on POD-C451-PGA Board

JB4: Power from POD

JB3 and JB5: POD Crystal

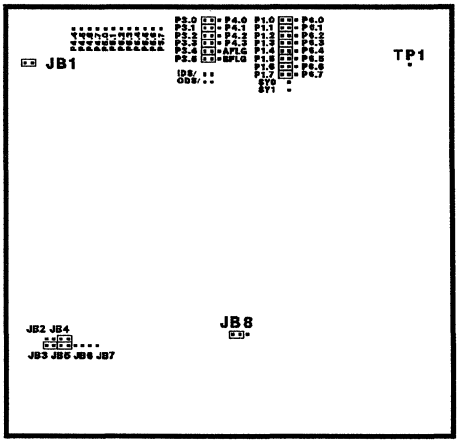


Figure 35. POD-C451-PGA Board Configured for target power and internal crystal

POD-C451B-PGA *POD TYPE: BONDOUT*

The software must be 5.47 or later. Invoke the EMUL51-PC as in this example:

```
EMUL51 -p80451b -e110 -m128 -t100 -f16        ; typical invocation
```

The pod board is used to emulate Philips/Signetics' 80C451, 83C451, and 87C451.

The EPROM on the emulator board must be version COM 1.2 or later. If it is older replace it with the EPROM enclosed with the pod.

The bottom side of the board has a 68 pin PGA plug. If the target system has a PGA socket, the pod can be used directly. If the target is PLCC or DIP, an adapter has to be used. The pod goes into the target system, replacing the microcontroller IC.

The pod board has a Signetics bondout chip to emulate the microcontroller chip.

The pod board has five LEDs marked M, E, P, I, and R:

"M" is red, and if lit means that the system is in monitor mode. In monitor mode, the bondout chip is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

"E" is green, and if lit means that the system is in emulation mode. In emulation mode the bondout chip is executing user code.

"P" is orange, and if lit means that the system is in power down mode. Only reset can take it out of this mode.

"I" is orange, and if lit means that the system is in idle mode. An interrupt or reset can take it out of this mode.

"R" is red, and if lit means that the bondout chip RESET input is high (ON).

POD-C451B-PGA

Note: The references below to 'JB' are all on the PC board. Some of the references are on Figures 36 and 37.

The board has three jumpers for power and crystal connection to the bondout chip. They are located between the bondout chip U12, and the crystal Y1.

The right jumper marked V is used for power to the bondout chip. If the power is to be supplied from the emulator, the jumper should be across the two upper pins (INT). If power is to come from the target system, the jumper should be across the two lower pins (EXT).

The two left jumpers marked X determine if the crystal or clock is from the target system, or if the crystal on the pod is used. The jumper should be in the upper two positions for the crystal on the pod board (INT), and the lower two positions for target supplied crystal or clock (EXT).

The jumpers above must be in internal (INT) positions when no target system is connected. This is also the default position.

The jumper marked RES connects the target RESET pin to the emulator. If the target system has a watch dog, it will probably interfere with monitor mode and the jumper should be removed.

The jumpers marked SY0 and SY1 are used to connect external signals to the emulator. They are used for the trace function. The lower pin, SY0, can also be used in the breakpoint logic. The name of the upper pin is SY1.

J1 carries signals from the emulator. The lower pin EM is high in monitor mode, and low in emulation mode. The other pins FLF, and ANB are not used currently, but may be used in the future.

The jumpers marked WR, RD, and PSEN determine whether the signals PSEN/, RD/, and WR/ go directly to the target system or go through gates. The gates will disable the signals, when not used to access the target system. If signals are not gated, the target system may be accessed when in monitor mode and accessing memory in the emulator board. RD/ and WR/ should not be gated, when used as regular port pins (P3.6 and P3.7).

POD-C451B-PGA

Jumper positions are in Figures 36 and 37.

For jumpers P6/P3: the right row of pins is connected to port P3, in order, with P3.7 at the top and P3.0 at the bottom. The left row of pins is connected to port P6, with P6.7 at the top and P6.0 at the bottom. The middle row of pins is reflected on the trace display of P3. The board is delivered with jumpers so that P3 is traced. If the jumpers are moved to the left row, P6 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

For jumpers P2/P1: the right row of pins is connected to port P1, in order, with P1.7 at the top and P1.0 at the bottom. The left row of pins is connected to port P2, with P2.7 at the top and P2.0 at the bottom. The middle row of pins is reflected on the trace display of P1. The board is delivered with jumpers so that P1 is traced. If the jumpers are moved to the left row, P2 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

For jumpers P4/P5 (JB16): the left row of pins is connected to port P4, in order, with P4.0 at the top and P4.7 at the bottom. The right row of pins is connected to port P5, with P5.0 at the top and P5.7 at the bottom.

For jumpers JB3: the left row of pins is connected to port P0, in order, with P0.0 at the top and P0.7 at the bottom. The right row of pins is connected to miscellaneous control lines: C1 is high every time the processor fetches the first byte of an instruction from memory. INTA goes low for one cycle for every interrupt acknowledged by the CPU. AFLAG, BFLAG, IDS, ODS are connected to the corresponding processor pin.

If you want to trace any of P0, P4, P5, or C1, INTA etc. the corresponding pins must be connected to the "middle row" of either P3/P6 or P2/P1 jumper rows thereby replacing those ports in the trace display.

JB7 is used to set the amount of internal memory emulated. See Figures 36 and 37.

POD-C451B-PGA

S1 pushbutton is used to reset the bondout chip, and can be used instead of the target system reset.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC, and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Plug the board carefully into the target system. With the pod board oriented so the long ribbon cable goes out to the left. Make sure that the target system pin 1 is towards the bottom of the board.

Turn on power to the PC first, and then to the target system.

When turning power off, turn the target off first, then the PC.

POD-C451B-PGA

Board Layout Figure 36 shows locations of jumpers and Figure 37 shows the jumper pins in detail.

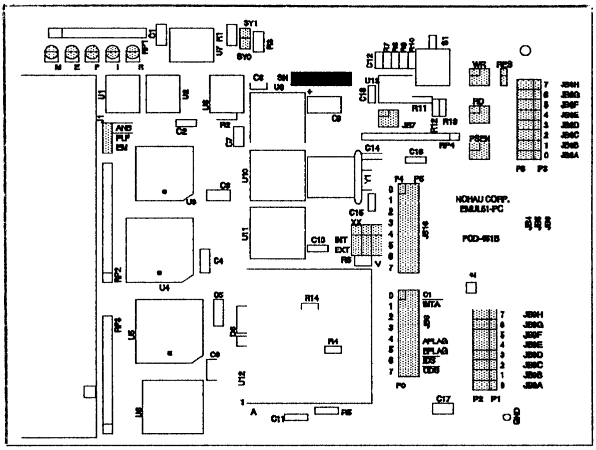


Figure 36. Locations of Jumpers on POD-C451B-PGA

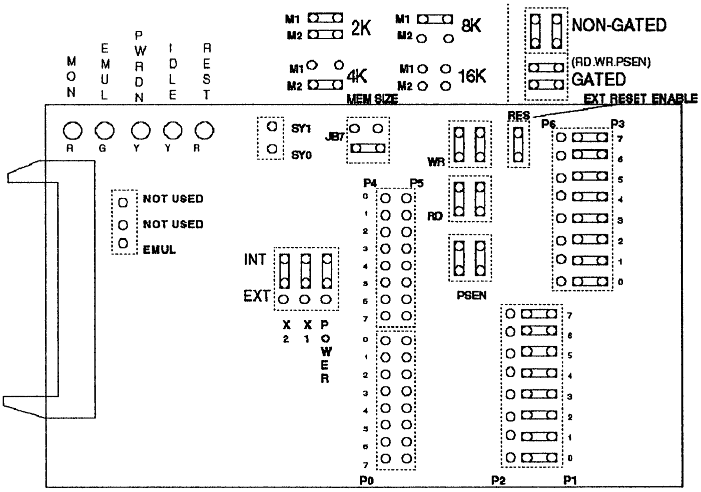


Figure 37. POD-C451B-PGA Jumper Diagram

POD-C452-PGA *POD TYPE: EXTERNAL MODE*

Also higher frequency versions.

This pod comes with a 68-pin PGA 80C452 microcontroller from INTEL. Dual on-chip DMA channels. 128-byte Bi-Directional FIFO Slave Interface.

This pod works up to 16 MHz. (On board crystal is 12 MHz). Two jumpers make it possible to use an external crystal or clock. Power to microcontroller chip can be taken either from target system or from the pod. (Pod receives its power from the PC).

This pod emulates the 80C452 only in its "external" mode. This means that P0 is used as a data port, P2 as an address port, P3.6 as WR, and P3.7 as RD.

You may use this POD with a standard emulator board. But for applications which use the DMA feature of the chip, you should use a "DMA-modified" option of the emulator board. This DMA option will also work with non-DMA applications and PODs.

With very few exceptions, the emulator does not take resources away from the emulated chip. But the DMA-modified emulator board and software use two internal 80C452 RAM bytes. The default addresses for the emulator to save DCON0 and DCON1 are 7E and 7F. If you need to have the emulator use different addresses other than 7E and 7F, you can use the NB command to change the addresses. The NB command (Nohau Byte) is similar to a debug command and allows modification of bytes in the emulator program.

In the following example, the addresses get changed to 2A and 2B. Note that the two addresses must be consecutive and that the first data address must be input twice. If these values are not suitable, you can select others which would fit with your application.

Example:

```
NB  E090 = 2A
NB  FF06 = 2A
NB  FF0A = 2B
```

These values will get set back to the default 7E and 7F each time you start the emulator software from DOS, or if you use a RESet EMulator command. However, you could put the commands in an include file to have them be set automatically each time you start the emulator software.

POD-C452-PGA

Table 10. Designators for POD-C452-PGA Board

Designator	*Name*	*Description*
JB1	EXT. RST.	With JB1 removed the target RESET is disconnected from the pod. This could be useful if there is a watchdog on the target system.
JB2	CHIP POWER	If jumpers are installed in positions away from the inside of the board, the chip Vcc and the chip Vpp will come from the target system. If jumpers are installed in positions toward the inside of the board, the chip Vcc and the chip Vpp will come from the pod. The jumpers closest to the 80C452 are for Vcc, and the jumpers closer to the capacitor are for Vpp.
JB3	PORT 1 and DB0 - 7	(8 jumpers). With JB3 you select if PORT 1 or the Host Bus(HB) should be traced. If the jumper is removed, and a EZ-hook wire is connected to the middle pin, any external signal with TTL levels may be traced.
JB4	PORT 3 and HB control	(8 jumpers). With JB4 you select if PORT 3, DRQI, DRQO or A0 - A2 and HB select signals should be traced. If the jumper is removed, and a EZ-hook wire is connected to the middle pin, any external signal with TTL levels may be traced.

(Table continued next page)

POD-C452-PGA

Designator	*Name*	*Description*
JB5	CRYSTAL selection	(2 jumpers). With jumpers installed toward the edge of the board, the crystal on the POD is selected. With jumpers installed away from the edge of the board, the target system crystal or clock is selected.
JB6	PSEN signal	If jumper is installed toward the quartz crystal, PSEN only goes out to target system when target CODE memory is addressed. If jumper is installed away from the crystal, PSEN will always go out to target system. (In this case PSEN must not open the data bus. Remove PROMs!).
J4	PORT 4	This is the connection to trace PORT 4. A jumper can be installed from this connector to any connection in the middle of JB3 or JB4 which is not used.

POD-C452-PGA

Board Layout Figure 38 shows locations of jumpers and Figure 39 shows a diagram of jumper pins.

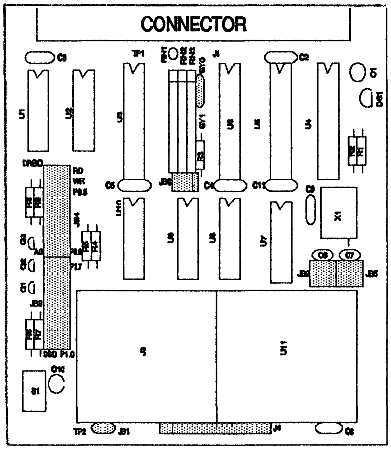
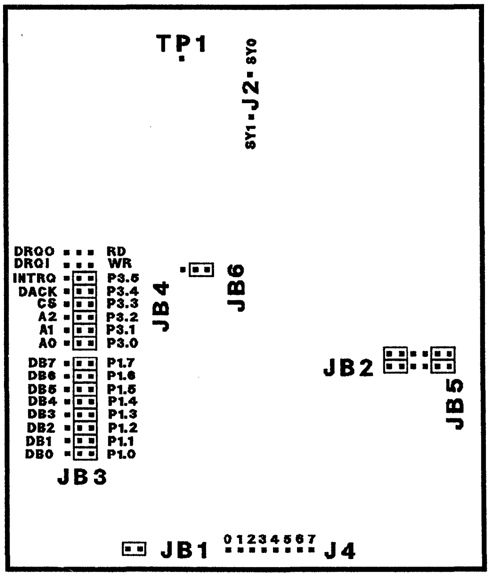


Figure 38. Locations of Jumpers on POD-C452-PGA Board



JB2 Pair: Power from POD shown

JB5 Pair: POD Crystal shown

Figure 39. POD-C452-PGA Board Configured for POD power and internal crystal

SAB C502 EMULATION

The SAB C502 can be emulated with the POD-C517AB-PGA-18, an adapter and user software accommodation of the device differences between the SAB 80C517A and the SAB C502.

EMUL51-PC/AD-502

The adapter is the EMUL51-PC/AD-502. It has a 68-pin socket on top that accepts the pin grid array (PGA) pins of the POD-C517AB-PGA-18. The bottom of the adapter is a 44-pin PGA plug. It can plug directly into a PGA socket if the there is a PGA socket on the user target board. Or it can plug into a PGA-to-PLCC adapter to plug into a target PLCC socket.

Siemens has produced a detailed document about the emulation strategy. The document describes the differences between the SAB C502 and the 80C517AE bondout chip. The 80C517AE is the bondout chip on the POD-C517AB-PGA-18. The document has one or the other of the following titles, but is the same document:

"EMULATION OF THE 8 BIT MC
SAB C502
with a standard emulation using the
Bondoutchip SAB 80C517AE"

or

"Technical Notes
for the AD-502 Adapter"

The document describes the register differences and other software considerations for emulation of the C502. It also describes the jumper settings for the adapter.

SAB C503 EMULATION

The SAB C503 can be emulated with the POD-C517AB-PGA-18, an adapter and user software accommodation of the device differences between the SAB 80C517A and the SAB C503.

EMUL51-PC/AD-503

The adapter is the EMUL51-PC/AD-503. It has a 68-pin socket on top that accepts the pin grid array (PGA) pins of the POD-C517AB-PGA-18. The bottom of the adapter is a 44-pin PGA plug. It can plug directly into a PGA socket if the there is a PGA socket on the user target board. Or it can plug into a PGA-to-PLCC adapter to plug into a target PLCC socket.

Siemens has produced a detailed document about the emulation strategy. The document describes the differences between the SAB C503 and the 80C517AE bondout chip. The 80C517AE is the bondout chip on the POD-C517AB-PGA-18. The document has one or the other of the following titles, but is the same document:

"EMULATION OF THE 8 BIT MC
SAB C503
with a standard emulation using the
Bondoutchip SAB 80C517AE"

or

"Technical Notes
for the AD-503 Adapter"

The document describes the register differences and other software considerations for emulation of the C503. It also describes the jumper settings for the adapter.

POD-C515A-PGA-18, Built from 515 Fab *POD TYPE: EXTERNAL MODE*

Identifying the POD

To use this description, the chip in the POD must be an 80C515A. If it is not, refer to the appropriate description for POD-532-PGA, POD-535-PGA and POD-C535-PGA.

To use this description, the board fabrication must match the layout shown in Figures 40 and 41. If it does not match, your POD is built from a 535 fab, not a 515 fab. Refer to the section that does match.

The software must be 5.7F or later. Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

```
EMUL51 -p80515A -e110 -m128 -t100 -f16 ; typical invocation
```

The POD-C515A-PGA-18 adapter is used to emulate Siemens 80C515A in external memory mode only.

The adapter goes into the target system replacing the microcontroller IC.

The solder side of the board has a 68 pin PGA plug. It is recommended to use a PGA socket in your target system for mechanical stability. If a 68 pin PLCC is used, an adapter can be supplied.

The pod has a regular Siemens 80C515A chip. It can be replaced by the user if it gets damaged.

The pod board has three LEDs, named MON, EMUL, and RESET:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

RESET is red, and means that the processor RESET# input is low (ON).

Processors 80535, 80C535, and 80C515A have different functions for pins 4, 37, and 68. This pod board has jumpers to connect these pins in the appropriate way. The table below shows the function of the processor pins

POD-C515A-PGA-18, Built from 515 Fab

and the default jumper configuration to run the board stand alone (factory installed):

Pin	80535 *(Reference Only)*	80C535 *(Reference Only)*	80C515A *(This POD)*
4	Vpd - V(cc)	PE# - GND	PE#/SWD - GND
37	Vbb - TARG	Vcc - VCC	Vcc - VCC
68	Vcc - VCC	Vcc - VCC	HWPD# - TARG

Jumpers XTAL determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the left two positions for the crystal on the pod board (INT), and the right two positions for target supplied crystal or clock (EXT).

The jumpers for XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

Jumper PSEN/ is normally in the left GT position. This means that PSEN/ is held high (inactive), when code in the emulator memory is accessed. This is to avoid collisions with memory on the target board. If the target is dependent on PSEN/ for timing or other purposes, the jumper can be moved to the right NG (non gated) position. However all target code memory has to be removed in this case to avoid collisions on the data bus.

Jumper RST connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulator in monitor mode and RST should be removed.

Jumper pins SY0 and SY1 are used to connect external signals to the emulator. They are used for the trace function. The left pin, SY0, can also be used in the breakpoint logic. The name of the right pin is SY1.

POD-C515A-PGA-18, Built from 515 Fab

Jumpers P1/P5, upper row of pins are connected to port P1 in order with P1.0 on the left and P1.7 on the right. The lower row of pins are connected to port P5, with P5.0 on the left and P5.7 on the right. The middle row of pins are reflected on the trace display of P1. The board is delivered with jumpers so that P1 is traced. If the jumpers are moved to the lower row, P5 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumpers P3/P4, upper row of pins are connected to port P3 in order with P3.0 on the left and P3.5 on the right. The lower row of pins are connected to port P4, with P4.0 on the left and P4.7 on the right. The left six pins in the middle row are reflected on the trace display of P3. The right two pins are used as E0 and E1 in the trace. The board is delivered with jumpers so that P3 is traced. If the jumpers are moved to the lower row, P4 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumpers P6, are connected to P6 in order with P6.0 on the left, and P6.7 on the right.

The jumper pins between U4 and U5 carry signals to and from the emulator and trace boards. The left pin EM/ is high in monitor mode, and low in emulation mode. The middle pins are called FLF/ and ANB/ and are used with the Advanced Trace Board (ATR). See the documentation for the ATR. The right pin (BSW) is used for bank switching.

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Processor pin 68 has a 100 kohm pullup resistor to Vcc (R9). The reason is to pull this pin high for 80C515A in order to avoid Hardware Power Down(HWPD#). This resistor may be removed if it is a problem.

POD-C515A-PGA-18, Built from 515 Fab

Processor pin 68 has a 0.1 uF decoupling capacitor (C14) to ground. For 80C515A this may be a problem for proper operation of HWPD#. It may be removed for 80C515A.

P3.6 and P3.7 go through two 74AS32 gates to the target system. This may add an extra delay up to 10 ns.

In gated mode, PSEN/ goes trough a 74AS32 to the target system. This may add an extra delay up to 6 ns.

SY0, SY1, E0, and E1 have each a 100 kohm pullup resistor.

For the 80C515A, it is highly recommended to emulate the internal XRAM using the emulator ram. This means that the XMAP0 bit in the SYSCON register should not be set to '0'. It XMAP0 is set to '0', breakpoints will not work properly.

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-C515A-PGA-18, Built from 515 Fab

Board Layout Figure 40 shows locations of jumpers and Figure 41 shows the jumper pins in detail.

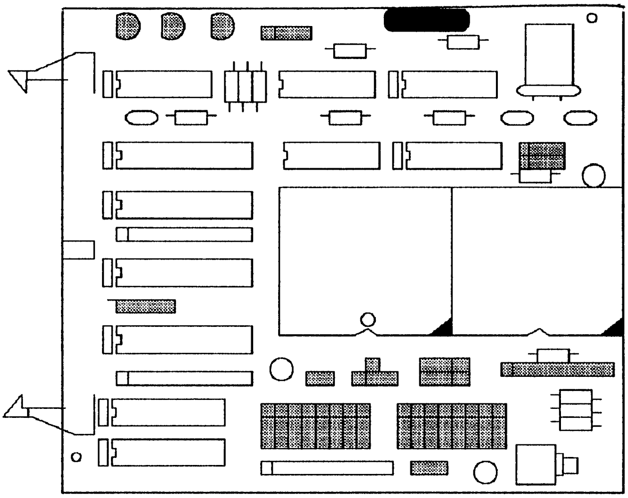


Figure 40. Locations of Jumpers on POD-C515A-PGA-18

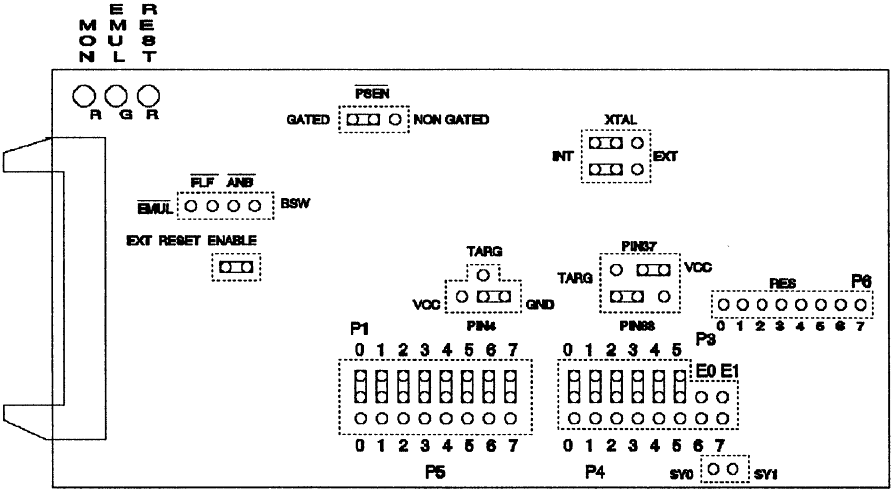


Figure 41. POD-C515A-PGA-18 Jumper Diagram

POD-C515A-PGA-18, Built from 535 Fab *POD TYPE: EXTERNAL MODE*

Identifying the POD

To use this description, the chip in the POD must be an 80C515A. If it is not, refer to the appropriate description for POD-532-PGA, POD-535-PGA and POD-C535-PGA.

To use this description, the board fabrication must match the layout shown in Figures 42 and 43. If it does not match, your POD is built from a 535 fab, not a 515 fab. Refer to the section that does match.

The socket on this pod has a Siemens 80C515A microcontroller with on-chip A/D converter with programmable reference voltages, watchdog timer, six 8-bit ports and 3 x 16 bit timer/counter. This pod works up to 18 MHz. Two jumpers make it possible to use an external crystal or clock. Power to the microcontroller chip can be taken either from the target system or from the pod. (The pod receives its power from the PC.) This pod emulates the 80C515A only in its "external" mode. This means that P0 is used as a data port, P2 is used as an address port, P3.6 is used as WR, and P3.7 is used as RD.

Hardware considerations:

The POD is a POD-535 modified for Siemens SAB80C515A. Pin 68 (HWPD#) is isolated from Vcc and connected directly from the 80C515A to the "BOTTOM PIN". The POD also has an added 100 kohm pullup resistor on this pin so that the POD will work without being hooked up to a target system.

JB2, A side, is connected to pin 37 of the 80C515A instead of pin 68.

JB2, B side, left (factory) position, connects pin 4 (PE#/SWD) to ground instead of Vcc so that the watch dog is disabled when the POD is not hooked to a target system.

Software considerations:

The EMUL51 does not fully support the internal XRAM of 80C515A. The SYSCON register should not be changed by the software. Instead the XRAM should be emulated by mapping the XRAM to the emulator. See the MAP command in the Commands Chapter.

POD-C515A-PGA-18, Built from 535 Fab

If the XMAPO bit is set to zero, breakpoints will not work properly. Also the "F" flag in the trace will not work properly which means that the disassembly of trace will not work. If the XMAP1 bit is set to one, Trace will work better, but the chip has to be reset after breakpoint.

Note that the XMAPO bit cannot be set to one after being reset to zero by software. Only RESET can set it to one.

The POD currently supports stepping code/marking ES-AB. Consult Nohau if you have other parts.

Table 11. Designators for POD-C515A-PGA-18, Built from 535 Fab

Designator	*Name*	*Description*
JB1	RST Selection	With JB1 installed, pin 10 on the POD board microprocessor (RST) is connected to the target system, which allows reset to be performed by S1 or the target system. With JB1 removed, RST is connected only to the S1 switch and the PC controlled reset line. This is useful if you have a software controlled watchdog, that keeps resetting the microprocessor.
JB2	POD Board Power, PE#/SWD	This jumper determines where the POD board gets its power, and PE#/SWD is connected. With the jumpers as preset by the factory (see figure), power is received from the PC and where PE#/SWD is grounded. If the jumpers are changed to the opposite positions, power is taken from the target system and PE#/SWD is connected to the target. The B side of the jumpers is connected to PE#/SWD (pin 4), and the A side is connected to Vcc (pin 37 of 80C515A).

(Table continued next page)

POD-C515A-PGA-18, Built from 535 Fab

Designator	*Name*	*Description*
JB3-JB6	Crystal Selection	With JB4 and JB5 jumpers installed, the crystal on the POD board is connected to the microprocessor. When the two jumpers are installed in JB3 and JB6, the crystal of the target system is connected to the microprocessor.
JB7	PSEN Signal	As shipped, the PSEN signal only goes out to the target system when code is executed from target system PROM. In the other position PSEN always goes out to the target.
P1-0 thru P5-7	Port Connections	Ports P1-0 through P1-7 and P3-0 through P3-5 are connected to the trace board when the jumpers are in their preset positions. If the jumpers are removed, external signals to be traced can be connected to the middle pins with optional EZ-hooks. With the jumpers in the opposite position, ports P4 and P5 will be traced.
P4-6 P4-7	Port or External Connections	If jumpers are mounted here, P4-6 and P4-7 will be traced instead of E0 and E1. To trace external signals (E0, E1) attach EZ-hooks to the pins closest to connector. (Marked E0, E1 in the figure.)
SY0, SY1	External Signals	These two pins can also be connected to external signals with the optional EZ- hooks. No jumpers are used here. (SY0 may be used to trigger breakpoints.)
TP1	Trigger Point	This pin is low when the emulator is running. It goes high as soon as a break condition is detected. This may be used to trigger a memory oscilloscope.

POD-C515A-PGA-18, Built from 535 Fab

Board Layout Figure 42 shows locations of jumpers and Figure 43 shows a diagram of the jumper pins.

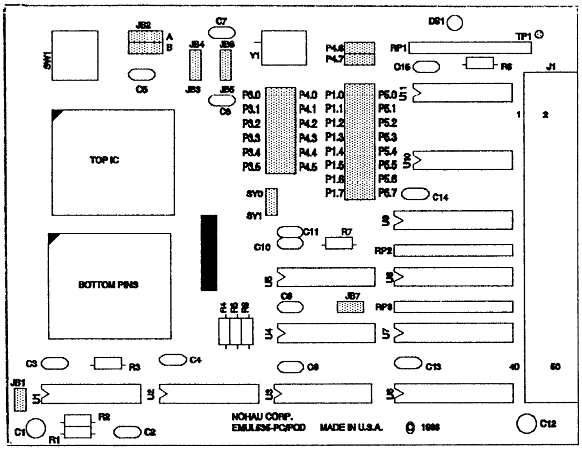


Figure 42. Locations of Jumpers on POD-C515A-PGA-18, Built from 535 Fab

JB2 Pair: Power from POD shown

JB4 and JB5: POD crystal

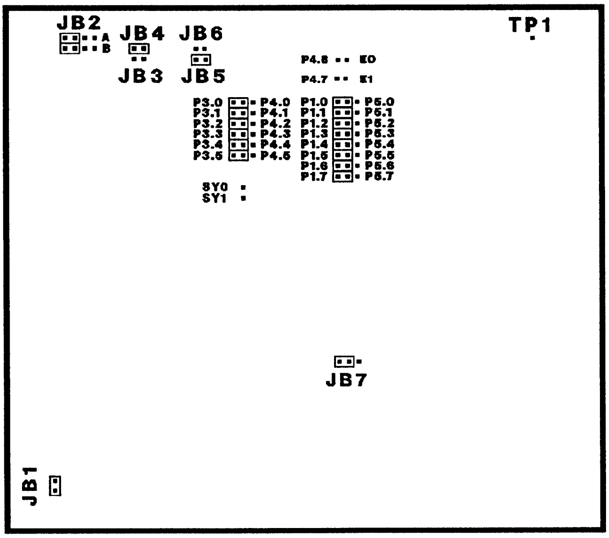


Figure 43. POD-C515A-PGA-18, Built from 535 Fab, Configured for power from POD and internal crystal

POD-C515A-PGA-18, Built from 535 Fab

WATCHDOG APPLICATIONS If you enable the watchdog, use a special EPROM for the emulator board. Use software version 5.5D or higher. Use COM PROM 1.32.

Follow these instructions before you enable the watchdog in your application. Your application will need to service the watchdog.

ENABLE Watchdog control (Example):

1. Install COM EPROM 1.32
2. Rename 80535.STR 80535.OLD
3. Copy 80535WD.STR 80535.STR

DISABLE Watchdog control (Example):

1. Remove COM EPROM 1.32 and reinstall the regular EPROM
2. Delete 80535.STR
3. Rename 80535.OLD 80535.STR

POD-C517A-PGA-18 *POD TYPE: EXTERNAL*

Hardware considerations:

The POD-517A is a modified POD-537 POD for Siemens SAB80C517A. Pin 60 (HWPD#) is isolated from ground and connected from the IC to the target. The POD also has an added 100 kohm pullup resistor on this pin so that the POD will work without being hooked up to a target system.

Software considerations:

Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

```
EMUL51 -p80517A -e110 -m128 -t100 -f16  ; typical invocation
```

The EMUL51 with POD-517A does not fully support the internal XRAM. The SYSCON register should not be changed by the software. Instead the XRAM should be emulated by mapping the XRAM to the emulator. See the MAP command in the manual.

If the XMAP0 bit is set to zero, breakpoints will not work properly. Also the "F" flag in the trace will not work properly which means that the disassembly of trace will not work. If the XMAP1 bit is set to one, Trace will work better, but the chip has to be reset after breakpoint.

Note that the XMAP0 bit can not be set to one after being reset to zero by software. Only RESET can set it to one.

POD-C517A-PGA-18

Board Layout Figure 44 shows locations of jumpers and Figure 45 shows the jumper pins in detail.

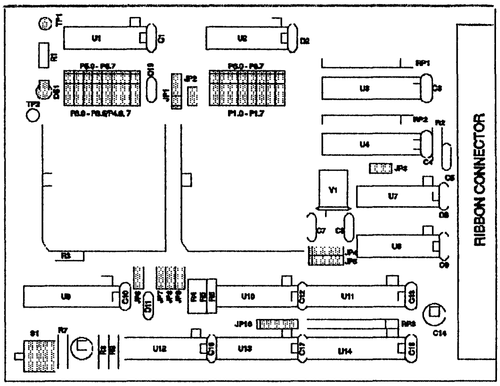
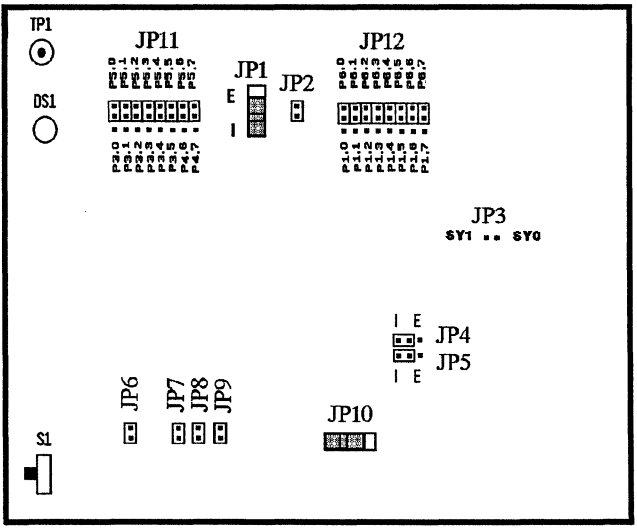


Figure 44. Locations of Jumpers on POD-C517A-PGA-18

JP1: Power from PC shown



JP10: Gated PSEN/ selected JP4, JP5: Pod Crystal selected

Figure 45. POD-C517A-PGA-18 Jumper Diagram

POD-C517AB-PGA-18 *POD TYPE: BONDOUT*

The POD-C517AB-PGA-18 is used to emulate the Siemens microcontrollers: SAB80C517A, SAB83C517A, SAB80C515A, and SAB83C515A.

Identifying the POD

The silkscreen on this POD reads "POD-517B REV. C". This assembly uses the same fab as the revision C of POD-517B-PGA (non-A bondout).

The bondout chip on the POD is an 80C517AE, and not an 80C517E. The identification of the chip may not be visible. If you are not sure whether your POD is A or non-A, contact your supplier to determine which model it is.

The EPROM on the emulator board must be version COM 1.2 or later. If it is older replace it with the EPROM enclosed with the pod.

The pod goes into the target system replacing the microcontroller IC.

The bottom side of the board has a 84 pin PGA plug. If the target system is a 517A and has a PGA socket, the pod can be used directly. If it is a 517A PLCC, an adapter has to be used. If the target system is a 515A, an 84 pin to 68 pin PLCC adapter has to be used. This adapter is not the same as the 84 pin to 68 pin adapter for the non-A POD-C517B-PGA bondout POD.

The POD-C517AB-PGA-18 board has a SAB80C517AE bondout chip.

The pod board has three LEDs named MON, EMUL, and RES:

MON is red, and if lit means that the system is in monitor mode. In monitor mode, the bondout chip is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and if lit means that the system is in emulation mode. In emulation mode the bondout chip is executing user code.

RES is red, and if lit means that the bondout chip RESET input is high (ON).

Note: The references below to 'JB' are all on the PC board. Some of the references are on Figure 47.

POD-C517AB-PGA-18

The board has three jumpers for power and crystal connection to the bondout chip:

JB7 is marked PWR and is used for power to the bondout chip. If the power is to be supplied from the emulator, the jumper should be across the two left pins (INT). If power is to come from the target system, the jumper should be across the two right pins (EXT).

JB11, and JB12 are marked XTAL and determine if the crystal or clock is from the target system, or if the crystal on the pod is used. The jumper should be in the lower two positions for the crystal on the pod board (INT), and the upper two positions for target supplied crystal or clock (EXT).

All jumpers JB7, JB11, and JB12 must be in internal (INT) positions when no target system is connected. This is also the default position.

JB10 connects the target RESET pin to the emulator. If the target system has a watch dog, it will probably interfere with the emulation and JB10 should be removed.

JB9 is used to connect external signals to the emulator. They are used for the trace function. The upper pin, SY0, can also be used in the breakpoint logic. The name of the lower pin is SY1.

JB1 carries signals from the emulator. The left pin E (EM/) is high in monitor mode, and low in emulation mode. The middle pin F (FLF/) and the right pin A (ANB/) and are used with the Advanced Trace Board (ATR). See the documentation for the ATR.

The jumper positions of JB2, JB3, and JB4 determine whether the signals PSEN/, RD/, and WR/ go directly to the target system or go through gates. The gates will disable the signals when not used to access the target system. If not gated, the target system may be accessed when in monitor mode and accessing memory in the emulator board. RD/ and WR/ should not be gated when used as regular port pins (P3.6 and P3.7).

Jumper SWD is used to connect the PE#/SWD pin (pin 4) to ground to disable the watch dog.

POD-C517AB-PGA-18

Jumper JP1 must be out to use this POD as POD-C517AB-PGA-18. JP1 connects pin 60 (HWPD# or HWPD/) to ground.

The pod has a 100 kohm pullup resistor on the HWPD# signal, so that HWPD# will be high if the POD-C517AB-PGA-18 is used stand-alone.

Jumpers P3/P5, upper row of pins are connected to port P3 in order with P3.7 on the left and P3.0 on the right. The lower row of pins are connected to port P5, with P5.7 on the left and P5.0 on the right. The middle row of pins are reflected on the trace display of P3. The board is delivered with jumpers so that P3 is traced. If the jumpers are moved to the lower row, P5 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumpers P1/P2, upper row of pins are connected to port P1 in order with P1.7 on the left and P1.0 on the right. The lower row of pins are connected to port P2, with P2.7 on the left and P2.0 on the right. The middle row of pins are reflected on the trace display of P1. The board is delivered with jumpers so that P1 is traced. If the jumpers are moved to the lower row, P2 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumpers P0/P4, upper row of pins are connected to port P0 in order with P0.7 on the left and P0.0 on the right. The lower row of pins are connected to port P4, with P4.7 on the left and P4.0 on the right.

Jumpers P6/P7, upper row of pins are connected to port P6 in order with P6.7 on the left and P6.0 on the right. The lower row of pins are connected to port P7, with P7.7 on the left and P7.0 on the right. If you are emulating 515A, the lower pins marked P7 are instead P6, and the upper row is not used.

Jumpers P8 are connected to P8 in order with P8.3 on the left, and P8.0 on the right. They are not used for 515A.

POD-C517AB-PGA-18

Under P8 are two jumper pins: RO is connected to the chip output RO#, which goes low when at internal reset. C is high when the chip fetches the first byte of an instruction. It is also high in the first frame of an interrupt internal call.

JB8 is used to set the emulated processor, and to set how much internal memory is to be emulated. See Figure 47.

S1 push-button is used to reset the bondout chip, and can be used instead of target system reset.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground in the target system.

Plug the board carefully into the target system. With the pod board oriented so the long ribbon cable goes out to the left. Make sure that the target system pin 1 is towards the top of the board.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-C517AB-PGA-18

Board Layout

Figure 46 shows the locations of jumpers and Figure 47 shows the jumper pins in detail.

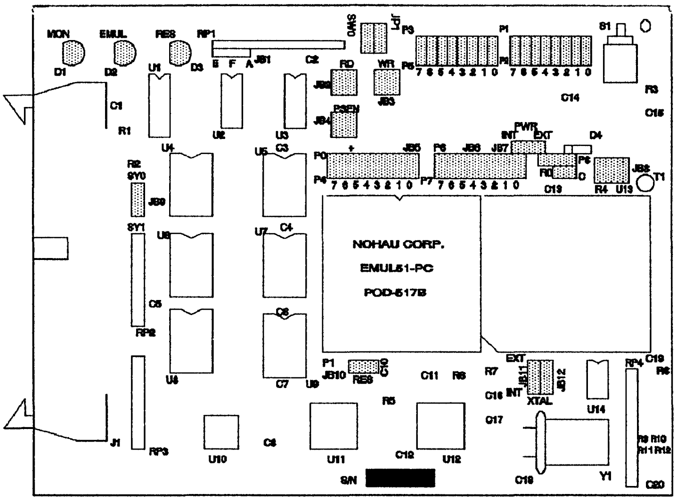


Figure 46. Locations of Jumpers on POD-C517AB-PGA

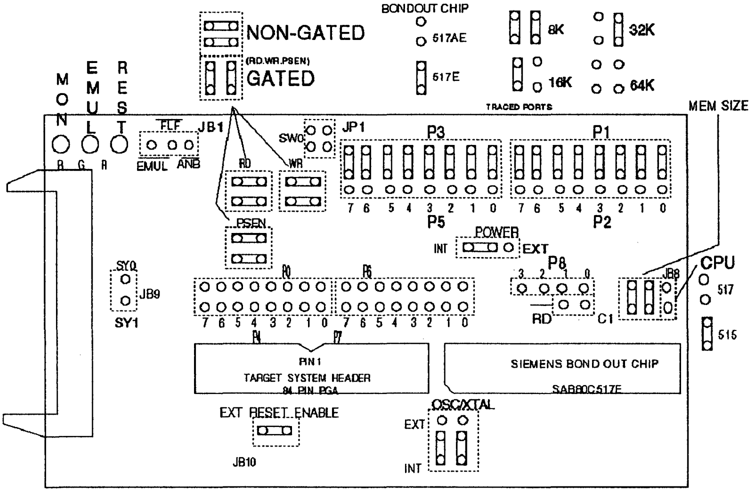


Figure 47. POD-C517AB-PGA Jumper Diagram

POD-C517B-PGA, Rev. A or B *POD TYPE: BONDOUT*

The pod board is used to emulate the Siemens microcontrollers: SAB80C517/80C537, SAB80C515/80C535, SAB80515/80535, and SAB80512/80532.

Identifying the POD

The silkscreen on this POD reads "POD-517B". It has no revision designation below the "POD-517B" mark. If the silkscreen reads "REV. C", the POD is either a POD-C517B-PGA Rev. C. or a POD-C517AB-PGA-18.

The EPROM on the emulator board must be version COM 1.2 or later. If it is older replace it with the EPROM enclosed with the pod.

The pod goes into the target system replacing the microcontroller IC.

The bottom side of the board has a 84 pin PGA plug. If the target system is 517/537 and has a PGA socket, the pod can be used directly. If it is 517/537 and PLCC, an adapter has to be used. If the target system is 515/535/515/535, an 84 pin to 68 pin PLCC adapter has to be used. Take care to use the correct adapter, since there is a similar but different adapter for the POD-C517AB-PGA "A" part bondout POD.

The board is delivered with an extra 84 pin PGA socket with pin 4 connected to Vss (ground). If pin 4 (PE/SWD) is left open, the watchdog for 517/537 will be enabled after reset. If running without a target system, and the software does not handle the watchdog, the socket can be plugged into, to disable the watchdog. Please note the position of pin 1.

The pod board has a Siemens SAB80C517E bondout chip to emulate the microcontroller chip.

The pod board has three LEDs named MON, EMUL, and RES:

MON is red, and if lit means that the system is in monitor mode. In monitor mode, the bondout chip is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and if lit means that the system is in emulation mode. In emulation mode the bondout chip is executing user code.

RES is red, and if lit means that the bondout chip RESET input is high (ON).

POD-C517B-PGA, Rev. A or B

Note: The references below to 'JB' are all on the PC board, some of the references are on Figure 49.

The board has three jumpers for power and crystal connection to the bondout chip:

JB7 is marked PWR and is used for power to the bondout chip. If the power is to be supplied from the emulator, the jumper should be across the two left pins (INT). If power is to come from the target system, the jumper should be across the two right pins (EXT).

JB11, and JB12 are marked XTAL and determine if the crystal or clock is from the target system, or if the crystal on the pod is used. The jumper should be in the lower two positions for the crystal on the pod board (INT), and the upper two positions for target supplied crystal or clock (EXT).

All jumpers JB7, JB11, and JB12 must be in internal (INT) positions when no target system is connected. This is also the default position.

JB10 connects the target RESET pin to the emulator. If the target system has a watch dog, it will probably interfere with the emulation and JB10 should be removed.

JB9 is used to connect external signals to the emulator. They are used for the trace function. The upper pin, SY0, can also be used in the breakpoint logic. The name of the lower pin is SY1.

JB1 carries signals from the emulator. The left pin E is high in monitor mode, and low in emulation mode. The other pins F, and A are not used currently, but may be used in the future.

The jumper positions of JB2, JB3, and JB4 determine whether the signals PSEN/, RD/, and WR/ go directly to the target system or go through gates. The gates will disable the signals when not used to access the target system. If not gated, the target system may be accessed when in monitor mode and accessing memory in the emulator board. RD/ and WR/ should not be gated when used as regular port pins (P3.6 and P3.7).

POD-C517B-PGA, Rev. A or B

PSEN gated/non-gated: In the non-gated position, the PSEN signal will be active to the user target during emulation if either the EA/ pin is low or the PSEN address is above the top of the memory boundary set with JB7, regardless of MAPC setting. A target EPROM might be unintentionally enabled in this case. The non-gated PSEN output is from the microcontroller. In monitor (not emulating) mode, the signal will only be active for CBYTE and code window display if the MAPC setting is to target.

In the gated PSEN position, during emulation the signal is only active if the MAPC is set to target and either the EA/ pin is low or the PSEN address is above the memory boundary. In monitor mode, the signal will be active to the target for CBYTE and code window display if MAPC is set to target and the EA/ pin is low or the address is above the boundary. The gated PSEN output is from a 74AS32 with a 330 ohm pullup.

WR & RD gated/non-gated: For P3.6 and P3.7 as ports, use the non-gated position. Non-gated will ignore MAPX settings during emulation. For monitor mode XBYTE commands and external data reads, the signals will be active if MAPX is set to target. The non-gated port lines are directly connected to the microcontroller.

In the gated positions, the read and write signals will follow the setting of the MAPX in both emulation and monitor modes. Use gated if the lines are used as external memory write and read lines. ENABLE XDATA in the emulator to use the external data window. Each gated output is from a 74AS32 with a 330 ohm pullup.

Jumpers P3/P5, upper row of pins are connected to port P3 in order with P3.7 on the left and P3.0 on the right. The lower row of pins are connected to port P5, with P5.7 on the left and P5.0 on the right. The middle row of pins are reflected on the trace display of P3. The board is delivered with jumpers so that P3 is traced. If the jumpers are moved to the lower row, P5 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumpers P1/P2, upper row of pins are connected to port P1 in order with P1.7 on the left and P1.0 on the right. The lower row of pins are connected

POD-C517B-PGA, Rev. A or B

to port P2, with P2.7 on the left and P2.0 on the right. The middle row of pins are reflected on the trace display of P1. The board is delivered with jumpers so that P1 is traced. If the jumpers are moved to the lower row, P2 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumpers P0/P4, upper row of pins are connected to port P0 in order with P0.7 on the left and P0.0 on the right. The lower row of pins are connected to port P4, with P4.7 on the left and P4.0 on the right.

Jumpers P6/P7, upper row of pins are connected to port P6 in order with P6.7 on the left and P6.0 on the right. The lower row of pins are connected to port P7, with P7.7 on the left and P7.0 on the right. If emulating 515/535/512/532, the lower pins marked P7 are instead P6, and the upper row is not used.

Jumpers P8, are connected to P8 in order with P8.3 on the left, and P8.0 on the right. Not used for 515/535/512/532.

Under P8 are two jumper pins: RO is connected to the chip output /RO, which goes low, when at internal reset. C is high when the chip fetches the first byte of an instruction. It is also high in the first frame of an interrupt internal call.

JB8 is used to set the emulated processor, and to set how much internal memory is to be emulated. See Figure 49.

S1 push-button is used to reset the bondout chip, and can be used instead of target system reset.

The EA pin has a 220 kohm pull-up resistor, so if it is left open, it will float high.

Internal ROM or EPROM must be mapped to the emulator memory.

The timers are stopped at a breakpoint and resumed again when emulation continues. The timers will therefore always advance the correct number of "ticks" with the following exceptions:

If the breakpoint is at a MOVX instruction, the timers will increment one "tick" less than it should have.

If emulation is STARTED in TARGET, the timers will always increment one "tick" extra at the next break, except if the break was at a MOVX in which case they will advance the correct number of "ticks".

Multiple Data Pointers and DPSEL

The emulator does not directly handle more than one data pointer. But you can write a macro to change the data pointer selection.

Preliminary Code Setup for Following Macros:

```
        1000 MOV .DPH,#00
        1002 MOV .DPL,#00
        1004 MOV .DPSEL,#00
        1006 NOP
```

Macro To Set DPSEL:

```
        CBYTE 1001 = %0
        CBYTE 1003 = %1
        CBYTE 1005 = %2
        GO FROM 1000 TO 1004
```

Example:

```
                :DW 5, 12, 14
```

Macro To Read:

```
        CBYTE 1005 = %0
        GO FROM 1004 TO 1006
```

Example:

```
                :DR 5
```

DPTR values for "5" will be shown.

POD-C517B-PGA, Rev. A or B

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC, and the target system.

Connect the black cable with the EZ hook to ground in the target system.

Plug the board carefully into the target system. With the pod board oriented so the long ribbon cable goes out to the left. Make sure, that the target system pin 1 is towards the top of the board.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-C517B-PGA, Rev. A or B

Board Layout

Figure 48 shows locations of jumpers and Figure 49 slows the jumper pins in detail.

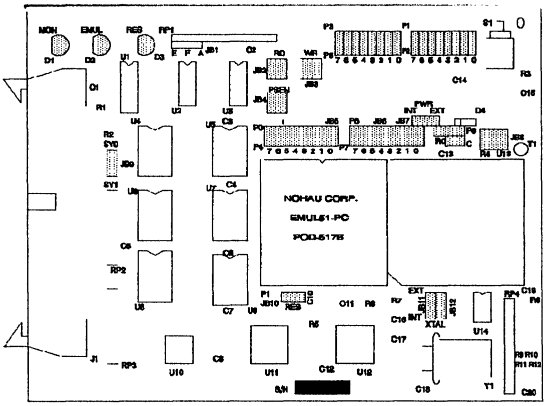


Figure 48. Locations of Jumpers on POD-C517B-PGA

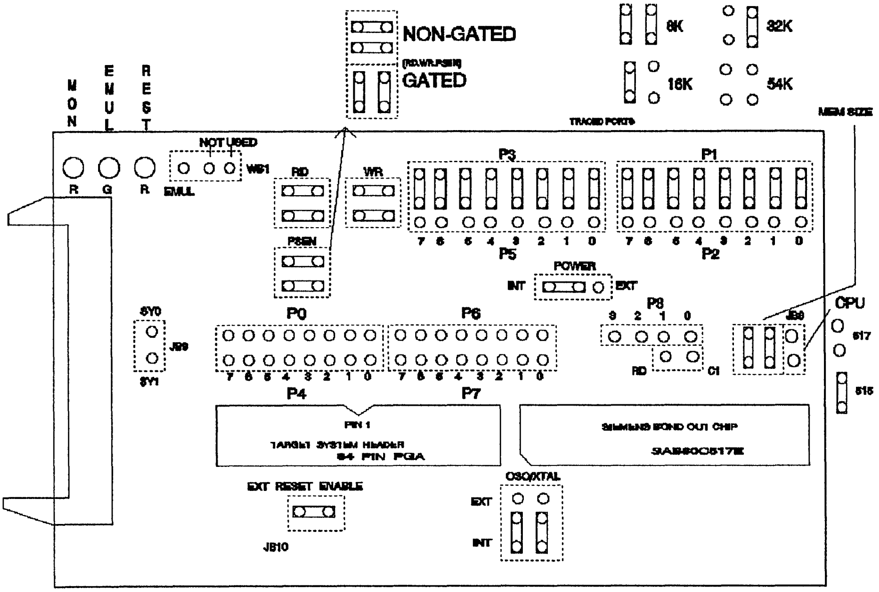


Figure 49. POD-C517B-PGA Jumper Diagram

POD-C517B-PGA, Rev. C *POD TYPE: BONDOUT*

Also higher frequency versions.

The POD-C517B-PGA is used to emulate the Siemens microcontrollers: SAB80C517/80C537, SAB80C515/80C535, SAB80515/80535, and SAB80512/80532.

Identifying the POD

The silkscreen on this POD reads "POD-517B REV. C". This assembly uses the same fab as the POD-517AB-PGA-18 (A part bondout).

The bondout chip on the POD is an 80C517E, not an 80C517AE. The identification of the chip may not be visible. If you are not sure whether your POD is A or non-A, contact your supplier to determine which model it is.

The EPROM on the emulator board must be version COM 1.2 or later. If it is older replace it with the EPROM enclosed with the pod.

The bottom side of the board has a 84 pin PGA plug. If the target system is 517/537 and has a PGA socket, the pod can be used directly. If it is 517/537 PLCC, an adapter has to be used. If the target system is 515/532/535, an 84 pin to 68 pin PLCC adapter has to be used. The pod goes into the target system replacing the microcontroller IC.

The POD-C517B-PGA has a Siemens SAB80C517E bondout chip to emulate the microcontroller chip.

The pod board has three LEDs named MON, EMUL, and RES:

MON is red, and if lit means that the system is in monitor mode. In monitor mode, the bondout chip is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and if lit means that the system is in emulation mode. In emulation mode the bondout chip is executing user code.

RES is red, and if lit means that the bondout chip RESET input is high (ON).

Note: The references below to 'JB' are all on the PC board. Some of the references are on Figures 50 and 51.

POD-C517B-PGA, Rev. C

The board has three jumpers for power and crystal connection to the bondout chip:

JB7 is marked PWR and is used for power to the bondout chip. If the power is to be supplied from the emulator, the jumper should be across the two left pins (INT). If power is to come from the target system, the jumper should be across the two right pins (EXT).

JB11, and JB12 are marked XTAL and determine if the crystal or clock is from the target system, or if the crystal on the pod is used. The jumper should be in the lower two positions for the crystal on the pod board (INT), and the upper two positions for target supplied crystal or clock (EXT).

All jumpers JB7, JB11, and JB12 must be in internal (INT) positions when no target system is connected. This is also the default position.

JB10 connects the target RESET pin to the emulator. If the target system has a watch dog, it will probably interfere with the emulation and JB10 should be removed.

JB9 is used to connect external signals to the emulator. They are used for the trace function. The upper pin, SY0, can also be used in the breakpoint logic. The name of the lower pin is SY1.

JB1 carries signals from the emulator. The left pin E (EM/) is high in monitor mode, and low in emulation mode. The middle pin F (FLF/) and the right pin A (ANB/) and are used with the Advanced Trace Board (ATR). See the documentation for the ATR.

The jumper positions of JB2, JB3, and JB4 determine whether the signals PSEN/, RD/, and WR/ are going directly to the target system or are going through gates. The gates will disable the signals when not used to access the target system. If not gated, the target system may be accessed when in monitor mode and accessing memory in the emulator board. RD/ and WR/ should not be gated when used as regular port pins (P3.6 and P3.7).

Jumper SWD is used to connect the PE#/SWD pin (pin 4) to ground to disable the watch dog.

POD-C517B-PGA, Rev. C

Jumper JP1 must be in for this assembly, POD-C517B-PGA. JP1 connects pin 60 (HWPD# or HWPD/) to ground.

The pod has a 100 kohm pullup resistor on the HWPD# signal, so that HWPD# will be high if the POD is used stand alone.

Jumpers P3/P5, upper row of pins are connected to port P3 in order with P3.7 on the left and P3.0 on the right. The lower row of pins are connected to port P5, with P5.7 on the left and P5.0 on the right. The middle row of pins are reflected on the trace display of P3. The board is delivered with jumpers so that P3 is traced. If the jumpers are moved to the lower row, P5 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumpers P1/P2, upper row of pins are connected to port P1 in order with P1.7 on the left and P1.0 on the right. The lower row of pins are connected to port P2, with P2.7 on the left and P2.0 on the right. The middle row of pins are reflected on the trace display of P1. The board is delivered with jumpers so that P1 is traced. If the jumpers are moved to the lower row, P2 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumpers P0/P4, upper row of pins are connected to port P0 in order with P0.7 on the left and P0.0 on the right. The lower row of pins are connected to port P4, with P4.7 on the left and P4.0 on the right.

Jumpers P6/P7, upper row of pins are connected to port P6 in order with P6.7 on the left and P6.0 on the right. The lower row of pins are connected to port P7, with P7.7 on the left and P7.0 on the right. If emulating 512/515/532/535, the lower pins marked P7 are instead P6, and the upper row is not used.

Jumpers P8 are connected to P8 in order with P8.3 on the left, and P8.0 on the right. Not used for 512/515/532/535.

POD-C517B-PGA, Rev. C

Under P8 are two jumper pins: RO is connected to the chip output RO#, which goes low, when at internal reset. C is high when the chip fetches the first byte of an instruction. It is also high in the first frame of an interrupt internal call.

JB8 is used to set the emulated processor, and to set how much internal memory is to be emulated. See Figure 51.

S1 push-button is used to reset the bondout chip, and can be used instead of target system reset.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground in the target system.

Plug the board carefully into the target system. With the pod board oriented so the long ribbon cable goes out to the left. Make sure that the target system pin 1 is towards the top of the board.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-C517B-PGA, Rev. C

Board Layout Figure 50 shows locations of jumpers and Figure 51 shows the jumper pins in detail.

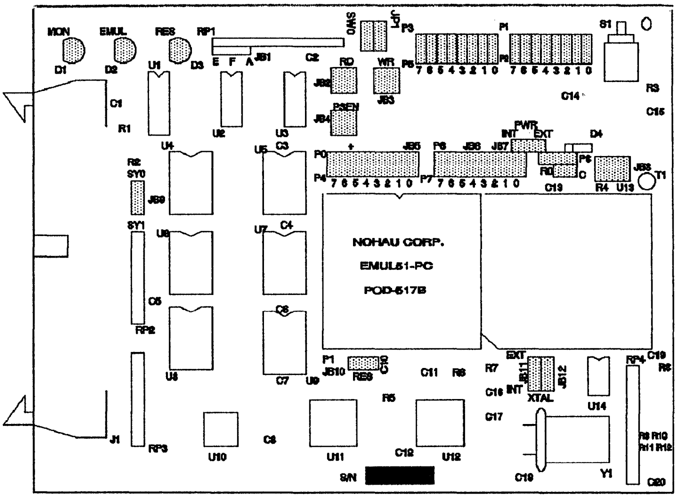


Figure 50. Location of Jumpers on POD-C517B-PGA, Rev. C.

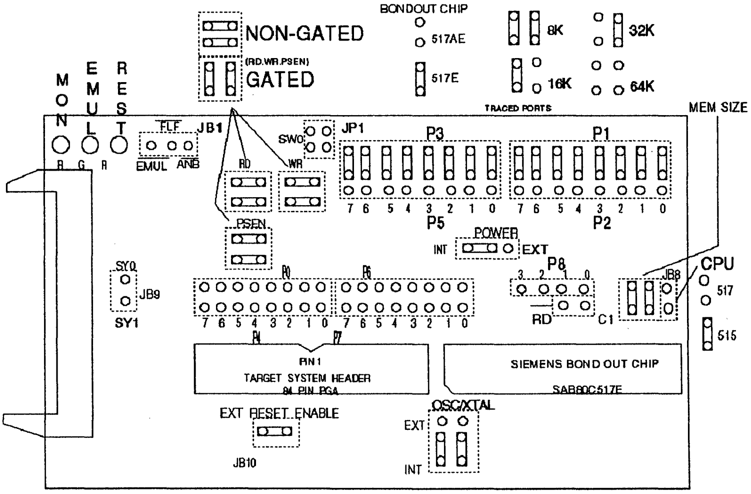


Figure 51. POD-C517B-PGA, Rev. C, Jumper Diagram

POD-C528 *POD TYPE: "HOOKS" MODE*

Also higher frequency versions.

The software must be 5.6A or later. Invoke the EMUL51-PC as in this example (please check Installation Chapter for option details):

```
EMUL51 -p80528 -e110 -m128 -t100 -f16   ; typical invocation
```

The POD-528 adapter is used to emulate Signetics microcontrollers 83C528.

The EPROM on the emulator board must be version COM 1.4 or later. If it is older replace it with an EPROM enclosed with the pod.

The solder side of the board has a 40 pin DIP plug. The plug goes into the target system replacing the microcontroller IC.

If a 44 pin PLCC is used, an adapter can be supplied.

The pod board has a Signetics 83C528 chip using a special emulation mode. This emulation mode also uses a programmable gate array from XILINX (LCA).

The pod board has five LEDs, named MON, EMUL, PWD, IDLE, and RES:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

PWD is yellow, and means that the processor is in power down mode.

POD-C528

IDLE is yellow, and means that the microcontroller is in idle mode.

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RES is red, and means that the processor RESET input is high (ON).

JP5 determines the mode of operation of P0, P2, P3.6 and P3.7. If only internal memory is to be emulated, both jumper tops (M1 and M2) should be in. This enables P0, P2, P3.6 and P3.7 to be used as general purpose ports.

If the lower jumper (M1) is out, P0 will have address and data and P2 will have address or Port 2 data for external code fetches. PSEN will be active. The emulator mapping (MAPC) will determine whether code is read from the emulator or from the target system.

If the upper jumper (M2) is out, P0 will have address and data and P2 will have address or Port 2 data for MOVX instructions. P3.6 and P3.7 will be active as RD and WR if data (MAPX) is mapped to the target system.

If only internal 83C528 RAM is used and your emulator board has 128k RAM, the data (MAPX) should be mapped to the emulator. M2 should be in.

NOTE:
The mode jumpers M1, M2 can be changed at any time but you have to issue the command RES CHIP so that the LCA can properly switch operational modes.

The board has three jumpers for power and crystal connection to the bond out chip:

Jumper PWR, JP6, is used for power to the processor. Power should be supplied from the emulator and the jumper should be across the two left pins (INT). Taking power from the target system, with the jumper across the two right pins (EXT) is NOT RECOMMENDED. Power should be on the "internal" position, INT, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout PODs.) The processor should be powered from the emulator since its operation is closely connected with the LCA.

Jumpers XTAL, JP3 and JP4 determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the right two positions for the crystal on the pod board (INT), and the left two positions for target supplied crystal or clock (EXT). The oscillator is on the LCA, so do NOT drive the clock inputs when the emulator is powered off.

If you use external clock, note that XTAL1 (pin 19) is input, and XTAL2 (pin 18) should be left open.

EXTERNAL NMOS DRIVE To accommodate external driving OSC signal (not a crystal) designed for NMOS parts, you can do either of the following: Change the target wiring so that STAL1 (pin 19) is the input and XTAL2 (pin 18) is open; or, you can cross-connect the XTAL jumpers to accomplish the same result. See the "NMOS external OSC connection" illustration in Figure 53. If you use a jumper, you will have to bend the pins slightly. The jumper block is 0.1 inches but the diagonal spacing between the pins is 0.14 inches. Alternatively, you can wire wrap a connection between the pins. Remember to not drive the clock input when the emulator is powered off.

All jumpers for PWR and XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

JP7 (RST) connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulation and JP7 should be removed.

JP2 is used to connect external signals to the emulator. They are used for the trace function. The right pin, SY0, can also be used in the breakpoint logic. The name of the left pin is SY1. The silk screen may be wrong on early boards.

POD-C528

JP1 carries signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The other pins are not used currently, but may be used in the future.

Jumper P1/P0 upper pins carry the processor port 1 pins in order with P1.0 on the left, and P1.7 on the right. Port 1 is connected directly to the processor. The lower pins carry the port 0 with P0.0 on the left and P0.7 on the right. Port 0 is emulated in the LCA. This means that P0 has better sink capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P1/P0 is normally connected to P1 using jumper tops. The signal level on the middle pins will be reflected on the trace display. The board is delivered with jumpers, so that P1 will be traced. External signals can be connected to the middle pins if the jumpers to P1 are removed. The input load is one ALS input (74ALS258).

Jumper P3/P2 upper pins carry the processor port 3 pins in order with P3.0 on the left, and P3.7 on the right. Port 3 is connected directly to the processor, except for P3.6 and P3.7 that go through a 74HC4066 that has an approximately 100 ohm series resistance. The lower pins carry the port 2 with P2.0 on the left and P2.7 on the right. Port 2 is emulated in the LCA. This means that P2 has better sink and source capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P3/P2 is normally connected to P3 using jumper tops. It can be used for P2 or other external signals the same way as P1/P0. The input load is one ALS input (74ALS258).

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Note:

Ports P0, and P2 are emulated, and have slightly different AC and DC characteristics:

POD-C528

Emulated I/O pins sink minimum 4 mA, and source (P2) minimum 4 mA for two clock cycles after an output low to high transition. As high outputs or inputs, P2 ports have a 22 kohm pull-up resistor.

Emulated outputs will change state three clock cycles later than a normal output.

The trace of accesses to the 83C528 AUX-RAM will contain correct addresses and data for writes. If your emulator has 128k RAM and data (MAPX) is mapped to emulator, the data will be written to the emulator RAM. For reads, the data displayed is the emulator RAM data if data is mapped to emulator. This data may NOT be the same as in the AUX-RAM unless it has already been written to the emulator RAM.

If your emulator board only has 32k RAM, data (MAPX) should normally be mapped to target.

P3.6 and P3.7 go through a 74HC4066 which adds about 100 ohms to their output impedance.

When accessing target memory, P2, P0, and ALE will be delayed approximately 25 ns.

The clock circuitry is emulated, but is very close to the processor clock.

The silk screen may be reversed for SY0 and SY1. SY1 is on the left.

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

If you break emulation using the 'Esc' key, and IDLE or PWD is lit, a number of "timeout error" messages will be displayed. This is normal, and you must issue the command RESET CHIP before you do anything else.

If you want to change processor, you have to use a Philips/Signetics 8XC528. Other bits do not affect the emulation mode.

POD-C528

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-C528

Board Layout

Figure 52 shows locations of jumpers and Figure 53 shows the jumper pins in detail.

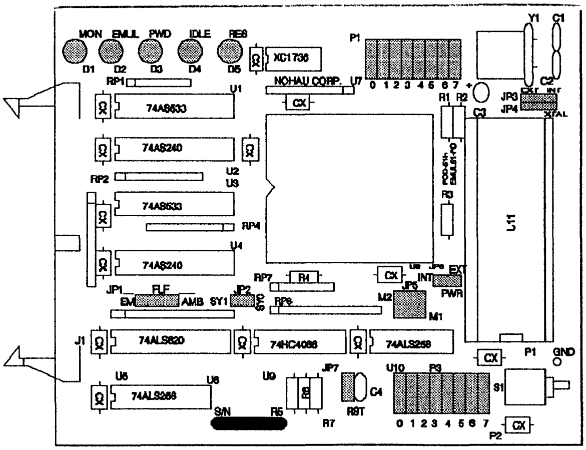


Figure 52. Locations of Jumpers on POD-C528

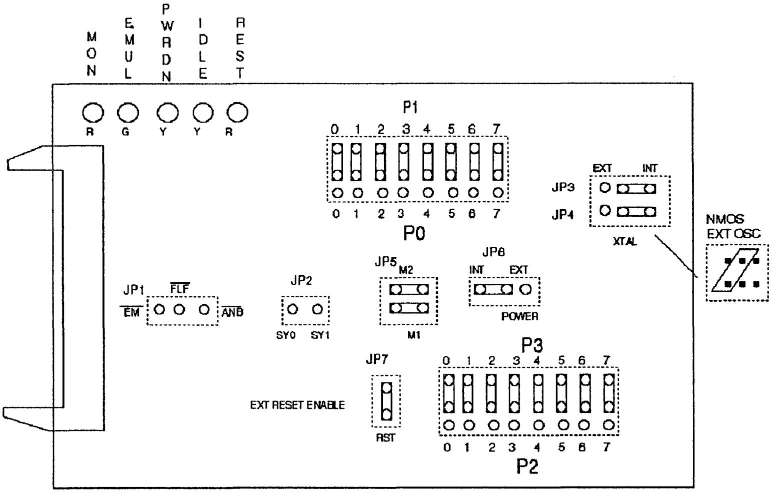


Figure 53. POD-C528 Jumper Diagram

POD-532-PGA, POD-535-PGA, POD-C535-PGA, Built from 535 Fab

POD TYPE: EXTERNAL MODE

Also higher frequency versions.

Identifying the POD

To use this description, the chip in the POD must be an 80532, 80535 or 80C535. If it is not, refer to the appropriate description for POD-C515A-PGA-18.

To use this description, the board fabrication must match the layout shown in Figures 54 and 55. If it does not match, your POD is built from a 515 fab, not a 535 fab. Refer to the section that does match.

The socket on this pod accepts a Siemens 80532, 80535 or 80C35 microcontroller with on-chip A/D converter with programmable reference voltages, watchdog timer, six 8-bit ports and 3 x 16 bit timer/counter. This pod works up to 12 MHz. Two jumpers make it possible to use an external crystal or clock. Power to the microcontroller chip can be taken either from the target system or from the pod. (The pod receives its power from the PC.) This pod emulates the 80535 only in its "external" mode. This means that P0 is used as a data port, P2 is used as an address port, P3.6 is used as WR, and P3.7 is used as RD.

Table 13. Designators for POD-532-PGA, POD-535-PGA, POD-C535-PGA

Designator	*Name*	*Description*
JB1	RST Selection	With JB1 installed, pin 10 on the POD board microprocessor (RST) is connected to the target system, which allows reset to be performed by S1 or the target system. With JB1 removed, RST is connected only to the S1 switch and the PC controlled reset line. This is useful if you have a software controlled watchdog, that keeps resetting the microprocessor.
JB2	POD Board Power	This jumper determines where the POD board gets its power. With the jumpers as preset by the factory (see figure), power is received from the PC. If the jumpers are changed to the opposite positions, power is taken from the target system. The B side of the jumpers is connected to Vpp (pin 4), and the A side is connected to Vcc (pin 68 of 80535).

(Table continued next page)

POD-532-PGA, POD-535-PGA, POD-C535-PGA, Built from 535 Fab

Designator	*Name*	*Description*
JB3-JB6	Crystal Selection	With JB4 and JB5 jumpers installed, the crystal on the POD board is connected to the microprocessor. When the two jumpers are installed in JB3 and JB6, the crystal of the target system is connected to the microprocessor.
JB7	PSEN Signal	As shipped, the PSEN signal only goes out to the target system when code is executed from target system PROM. In the other position PSEN always goes out to the target.
P1-0 thru P5-7	Port Connections	Ports P1-0 through P1-7 and P3-0 through P3-5 are connected to the trace board when the jumpers are in their preset positions. If the jumpers are removed, external signals to be traced can be connected to the middle pins with optional EZ-hooks. With the jumpers in the opposite position, ports P4 and P5 will be traced.
P4-6 P4-7	Port or External Connections	If jumpers are mounted here, P4-6 and P4-7 will be traced instead of E0 and E1. To trace external signals (E0, E1) attach EZ-hooks to the pins closest to connector. (Marked E0, E1 in the figure.)
SY0, SY1	External Signals	These two pins can also be connected to external signals with the optional EZ- hooks. No jumpers are used here. (SY0 may be used to trigger breakpoints.)
TP1	Trigger Point	This pin is low when the emulator is running. It goes high as soon as a break condition is detected. This may be used to trigger a memory oscilloscope.

POD-532-PGA, POD-535-PGA, POD-C535-PGA, Built from 535 Fab

Board Layout Figure 54 shows locations of jumpers and Figure 55 shows a diagram of the jumper pins.

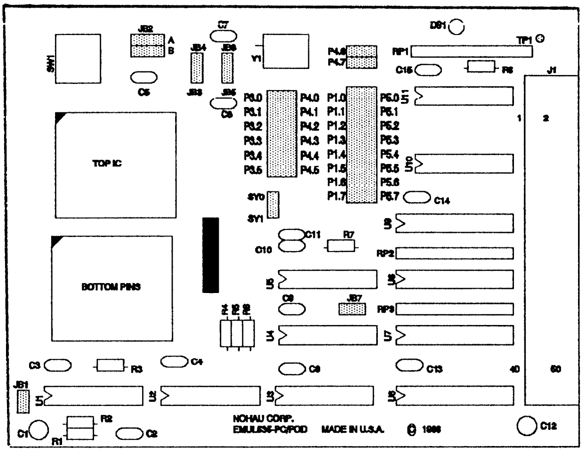


Figure 54. Locations of Jumpers on POD-532-PGA, POD-535-PGA, POD-C535-PGA Board

JB2 Pair: Power from POD shown

JB4 and JB5: POD crystal

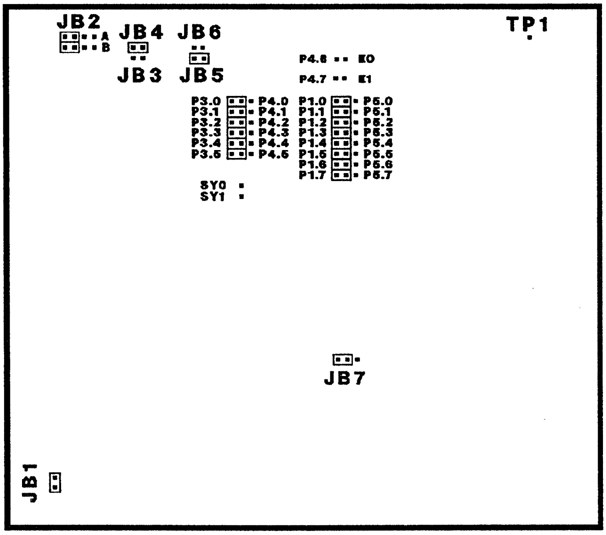


Figure 55. POD-532-PGA, POD-535-PGA, POD-C535-PGA Board Configured for power from POD and internal crystal

POD-532-PGA, POD-535-PGA, POD-C535-PGA, Built from 535 Fab

WATCHDOG APPLICATIONS If you enable the watchdog, use a special EPROM for the emulator board. Use software version 5.5D or higher. Use COM PROM 1.32.

Follow these instructions before you enable the watchdog in your application. Your application will need to service the watchdog.

ENABLE Watchdog control (Example):

1. Install COM EPROM 1.32
2. Rename 80535.STR 80535.OLD
3. Copy 80535WD.STR 80535.STR

DISABLE Watchdog control (Example):

1. Remove COM EPROM 1.32 and reinstall the regular EPROM
2. Delete 80535.STR
3. Rename 80535.OLD 80535.STR

POD-532-PGA, POD-535-PGA, POD-C535-PGA, Built from 515 Fab *POD TYPE: EXTERNAL MODE*

Also higher frequency versions.

Identifying the POD

To use this description, the chip in the POD must be an 80532, 80535 or 80C535. If it is not, refer to the appropriate description for POD-C515A-PGA-18.

To use this description, the board fabrication must match the layout shown in Figures 56 and 57. The silkscreen reads "POD-515A". If it does not match, your POD is built from a 535 fab, not a 515 fab. Refer to the section that does match.

Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

```
EMUL51 -p80535 -e110 -m128 -t100 -f16  ; typical invocation
```

The 80532, 80535 and 80C535 are invoked with -p80535.

This POD is used to emulate Siemens 80532, 80535 and 80C535 in external memory mode only.

The adapter goes into the target system replacing the microcontroller IC.

The solder side of the board has a 68 pin PGA plug. It is recommended to use a PGA socket in your target system for mechanical stability. If a 68 pin PLCC is used, an adapter can be supplied.

The POD has a regular Siemens 80532, 80535, or 80C535 chip. It can be replaced by the user if it gets damaged.

The pod board has three LEDs, named MON, EMUL, and RESET:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a

special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

RESET is red, and means that the processor RESET# input is low (ON).

Processors 80532, 80535, and 80C535 have different functions for pins 4, 37, and 68. This POD has jumpers to connect these pins in the appropriate way. The table below shows the function of the processor pins and the default jumper configuration to run the board stand alone (factory installed):

Pin	80532, 80535	80C535
4	Vpd - V(cc)	PE# - GND
37	Vbb - TARG	Vcc - VCC
68	Vcc - VCC	Vcc - VCC

When the pod is connected to the target system, it is recommended that all three jumpers are in the TARG position.

Jumpers XTAL determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the left two positions for the crystal on the pod board (INT), and the right two positions for target supplied crystal or clock (EXT).

The jumpers for XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

Jumper PSEN/ is normally in the left GT position. This means that PSEN/ is held high (inactive), when code in the emulator memory is accessed. This is to avoid collisions with memory on the target board. If the target is dependent on PSEN/ for timing or other purposes, the jumper can be moved to the right NG (non gated) position. However all target code memory has to be removed in this case to avoid collisions on the data bus.

Jumper RST connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulator in monitor mode and RST should be removed.

Jumper pins SY0 and SY1 are used to connect external signals to the emulator. They are used for the trace function. The left pin, SY0, can also be used in the breakpoint logic. The name of the right pin is SY1.

Jumpers P1/P5, upper row of pins are connected to port P1 in order with P1.0 on the left and P1.7 on the right. The lower row of pins are connected to port P5, with P5.0 on the left and P5.7 on the right. The middle row of pins are reflected on the trace display of P1. The board is delivered withjumpers so that P1 is traced. If the jumpers are moved to the lower row, P5 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumpers P3/P4, upper row of pins are connected to port P3 in order with P3.0 on the left and P3.5 on the right. The lower row of pins are connected to port P4, with P4.0 on the left and P4.7 on the right. The left six pins in the middle row are reflected on the trace display of P3. The right two pins are used as E0 and E1 in the trace. The board is delivered with jumpers so that P3 is traced. If the jumpers are moved to the lower row, P4 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumpers P6, are connected to P6 in order with P6.0 on the left, and P6.7 on the right.

The jumper pins between U4 and U5 carry signals to and from the emulator and trace boards. The left pin EM/ is high in monitor mode, and low in emulation mode. The middle pins are called FLF/ and ANB/ and are used with the Advanced Trace Board (ATR). See the documentation for the ATR. The right pin (BSW) is used for bank switching.

S1 push-button is used to reset the processor, and can be used instead of target system reset.

POD-532-PGA, POD-535-PGA, POD-C535-PGA, Built from 515 Fab

Note:
Processor pin 37 has a 0.1 uF decoupling capacitor (C13)to ground. If this is a problem for 80535 where this pin is used for VBB, capacitor C13 can be removed.

Processor pin 68 has a 100 kohm pullup resistor to Vcc (R9). The reason is to pull this pin high for 80C515A in order to avoid Hardware Power Down (HWPD#). This resistor may be removed if it is a problem.

P3.6 and P3.7 go through two 74AS32 gates to the target system. This may add an extra delay up to 10 ns.

In gated mode, PSEN/ goes trough a 74AS32 to the target system. This may add an extra delay up to 6 ns.

SY0, SY1, E0, and E1 have each a 100 kohm pullup resistor.

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-532-PGA, POD-535-PGA, POD-C535-PGA, Built from 515 Fab

Board Layout Figure 56 shows locations of jumpers and Figure 57 shows the jumper pins in detail.

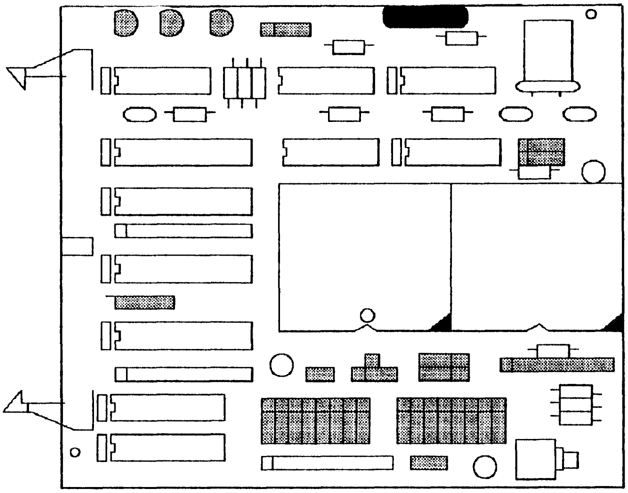


Figure 56. Locations of Jumpers on POD-532-PGA, POD-535-PGA, POD-C535-PGA, Built from 515 Fab

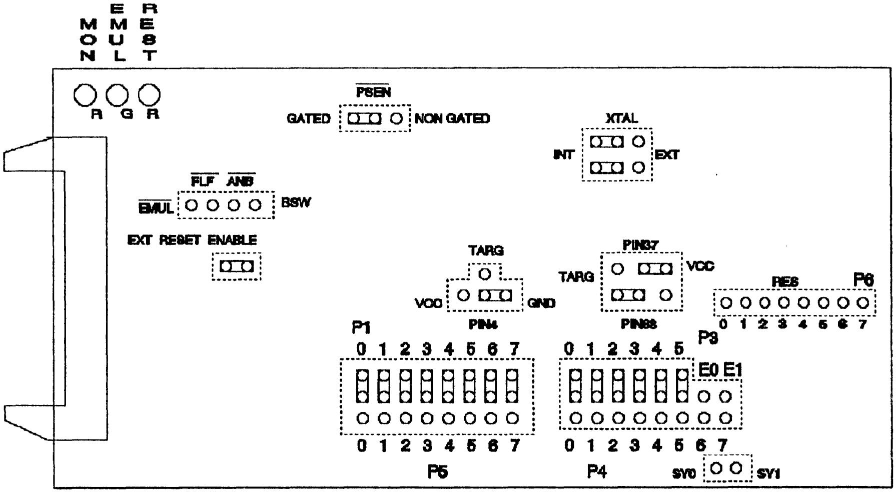


Figure 57. Jumper Diagram for POD-532-PGA, POD-535-PGA, POD-C535-PGA, Built from 515 Fab

POD-C537-PGA *POD TYPE: EXTERNAL MODE*

Also higher frequency versions.

The socket on this pod accepts a Siemens 80C537 PGA microcontroller. Two jumpers make it possible to use an external crystal or clock. Power to the microcontroller chip can be taken either from the target system or from the pod. (The pod logic receives its power from the PC.) This pod emulates the 80C537 only in its "external" mode. This means that P0 is used as a data port, P2 is used as an address port, p3.6 is used as WR, and P3.7 is used as RD.

Table 14. Designators for POD-C537-PGA

Designator	*Name*	*Description*
JP1	POWER	This jumper determines the source of power for the 80C537. If the slide switch is in the position away from the processor, the power to the processor is taken from the target system. If it is in the position toward the processor, power is taken from the pod board.
JP2	OWE	This jumper connects the OWE pin to ground if installed. If it is not installed, OWE is left open.
JP3	SY0 & SY1	These two pins are connected to the external signals SY0 and SY1 on the emulator and trace boards. SY1 is the pin closest to the crystal.
JP4, JP5	CRYSTAL Select	With both jumpers JP4 and JP5 in the position closest to the processor, the pod crystal is used. In the positions away from the processor, the target crystal is connected to the processor.

(Table continued next page)

POD-C537-PGA

Designator	*Name*	*Description*
JP6	RESET Enable	With this jumper in place, the target system's RESET will reach the processor. If the jumper is removed, it will not.
JP7	VAREF	With this jumper installed, VAREF is connected to the +5V of the pod board. If not installed VAREF is left open.
JP8	PE/SWD	With this jumper installed, PE/SWD is connected to GROUND. If not installed, PE/SWD is left open.
JP9	VAGND	VAGND is connected to ground when this jumper is installed. Otherwise it is left open.
JP10	PSEN/	This switch determines if PSEN/ comes out to the target during monitor mode. If the switch is in the position closer to the SIP resistor network, PSEN/ always goes out to the target In the other position PSEN/ will only reach the target during emulation.
JP11	Trace P3/P5	With jumpers installed toward the inside of the board, P3.0 through P3.5, P4.6 and P4.7 will be traced. With jumpers installed toward the edge of the board, P5.0 through P5.7 will be traced. If the jumpers are removed, external logic signals may be fed into the middle pins to be traced.
JP12	Trace P1/P6	With jumpers installed toward the inside of the board, P1.0 through P1.7 will be traced. With jumpers installed toward the edge of the board, P6.0 through P6.7 will be traced. If the jumpers are removed, external logic signals may be fed into the middle pins to be traced.

(Table continued on next page)

POD-C537-PGA

Designator	*Name*	*Description*
DS1	EMULATOR LED	DS1 lights when emulation mode is running.
S1	RESET Switch	Pressing this button resets the pod; same as if the command "RES CHIP" were issued.
GND	GROUND	Before plugging the pod board into the target system, connect a ground wire from the pod to the target system ground. This protects the components from damage due to static discharge.
TP1	TRIGGER Point	This pin is low when the emulator is running. It goes high as soon as a break condition is detected. May be used to trigger a memory oscilloscope.
TP2	RO/	Reset Output, pin 82 from 80C537.

POD-C537-PGA

Board Layout Figure 58 shows locations of jumpers and Figure 59 shows the jumper pins in detail.

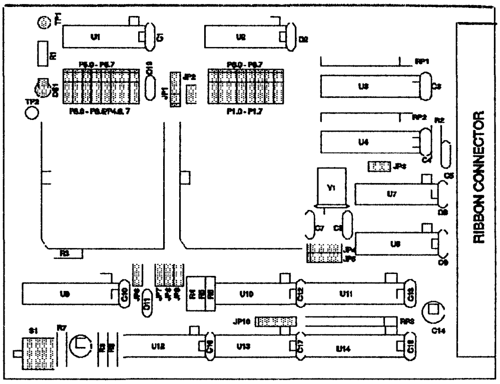


Figure 58. Locations of Jumpers on POD-C537-PGA

JP1: Power from PC shown

JP10: Gated PSEN/ selected JP4, JP5: Pod Crystal selected

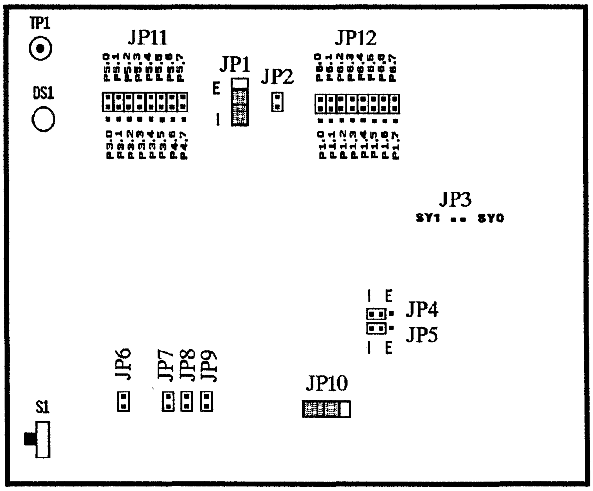


Figure 59. POD-C537-PGA Jumper Diagram

POD-C537-PGA

WATCHDOG APPLICATIONS If you enable the watchdog, use a special EPROM for the emulator board. Use software version 5.5D or higher. Use COM PROM 1.32.

Follow these instructions before you enable the watchdog in your application. Your application will need to service the watchdog.

ENABLE Watchdog control (Example):

1. Install COM EPROM 1.32
2. Rename 80535.STR 80535.OLD
3. Copy 80535WD.STR 80535.STR

DISABLE Watchdog control (Example):

1. Remove COM EPROM 1.32 and reinstall the regular EPROM
2. Delete 80535.STR
3. Rename 80535.OLD 80535.STR

POD-C550-PGA *POD TYPE: "HOOKS" MODE*

Also higher frequency versions.

The software must be 5.6A or later. Invoke the EMUL51-PC as in this example (please check Installation Chapter for option details):

```
EMUL51 -p80550 -e110 -m128 -t100 -f16   ; typical invocation
```

The POD-550 adapter is used to emulate Signetics microcontrollers 80C550, 83C550, and 87C550.

The EPROM on the emulator board must be version COM 1.4 or later. If it is older replace it with an EPROM enclosed with the pod.

The solder side of the board has a 44 pin PGA plug. It is recommended to use a PGA socket in you target system for mechanical stability. If a 44 pin PLCC is used, an adapter can be supplied.

The pod boards use a Signetics 80C550 chip in a special emulation mode. This emulation mode also uses a programmable gate array from XILINX (LCA).

The pod board has five LEDs, named MON, EMUL, PWD, IDLE, and RES:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

PWD is yellow, and means that the processor is in power down mode.

POD-C550-PGA

IDLE is yellow, and means that the microcontroller is in idle mode.

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RES is red, and means that the processor RESET input is high (ON).

JP4 determines the mode of operation of P0, P2, P3.6 and P3.7. If only internal memory is to be emulated, both jumper tops (M1 and M2) should be in. This enables P0, P2, P3.6 and P3.7 to be used as general purpose ports.

If the lower jumper (M1) is out, P0 will have address and data and P2 will have address or Port 2 data for external code fetches. PSEN will be active. The emulator mapping (MAPC) will determine whether code is read from the emulator or from the target system.

If the upper jumper (M2) is out, P0 will have address and data and P2 wil have address or Port 2 data for MOVX instructions. P3.6 and P3.7 will be active as RD and WR if data (MAPX) is mapped to the target system.

NOTE:
The mode jumpers M1, M2 can be changed at any time but you have to issue the command RES CHIP so that the LCA can properly switch operational modes.

The board has three jumpers for power and crystal connection to the bond out chip:

Jumper PWR, JP5, is used for power to the processor. Power should be supplied from the emulator and the jumper should be across the two right pins (INT). Taking power from the target system, with the jumper across the two left pins (EXT) is NOT RECOMMENDED. Power should be on the "internal" position, INT, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout PODs.) The processor should be powered from the emulator since its operation is closely connected with the LCA.

POD-C550-PGA

Jumpers XTAL determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the right two positions for the crystal on the pod board (INT), and the left two positions for target supplied crystal or clock (EXT). The oscillator is on the LCA, so do NOT drive the clock inputs when the emulator is powered off.

All jumpers for PWR and XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

JP2 (RST) connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulation and JP2 should be removed.

JP3 is used to connect external signals to the emulator. They are used for the trace function. The upper pin, SY0, can also be used in the breakpoint logic. The name of the lower pin is SY1. The silk screen may be wrong on early boards.

JP1 carries signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The other pins are not used currently, but may be used in the future.

Jumper P1/P0 upper pins carry the processor port 1 pins in order with P1.0 on the left, and P1.7 on the right. Port 1 is connected directly to the processor. The lower pins carry the port 0 with P0.0 on the left and P0.7 on the right. Port 0 is emulated in the LCA. This means that P0 has better sink capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P1/P0 is normally connected to P0 using jumper tops. The signal level on the middle pins will be reflected on the trace display. The board is delivered with jumpers, so that P1 will be traced. External signals can be connected to the middle pins if the jumpers to P0 are removed. The input load is one ALS input (74ALS258).

Jumper P3/P2 upper pins carry the processor port 3 pins in order with P3.0 on the left, and P3.7 on the right. Port 3 is connected directly to the processor, except for P3.6 and P3.7 that go through a 74HC4066 that has an approximately 100 ohm series resistance. The lower pins carry the port 2 with P2.0 on the left and P2.7 on the right. Port 2 is emulated in the LCA. This means that P2 has better sink and source capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P3/P2 is normally connected to P3 using jumper tops. It can be used for P2 or other external signals the same way as P1/P0. The input load is one ALS input (74ALS258).

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Note: Ports P0, and P2 are emulated, and have slightly different AC and DC characteristics:

Emulated I/O pins sink minimum 4 mA, and source (P2) minimum 4 mA for two clock cycles after an output low to high transition. As high outputs or inputs, P2 ports have a 22 kohm pull-up resistor.

Emulated outputs will change state three clock cycles later than a normal output.

If you use external clock, note that XTAL1 (pin 23) is input, and XTAL2 (pin 22) should be left open.

P3.6 and P3.7 go through a 74HC4066 which adds about 100 ohms to their output impedance.

When accessing target memory, P2, P0, and ALE will be delayed approximately 25 ns.

The clock circuitry is emulated, but is very close to the processor clock.

The silk screen may be reversed for P3/P2 and P1/P0. P3/P2 is in the upper right corner under the S/N block. P1/P0 is below the target PGA plug.

POD-C550-PGA

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

If you break emulation using the 'Esc' key, and IDLE or PWD is lit, a number of "timeout error" messages will be displayed. This is normal, and you must issue the command RESET CHIP before you do anything else.

If you use P1 pins only as digital inputs, AVcc (pin 1) should be connected to +5V and AVss (pin 4) should be connected to ground for proper operation.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-C550-PGA

Board Layout Figure 60 shows locations of jumpers and Figure 61 shows the jumper pins in detail.

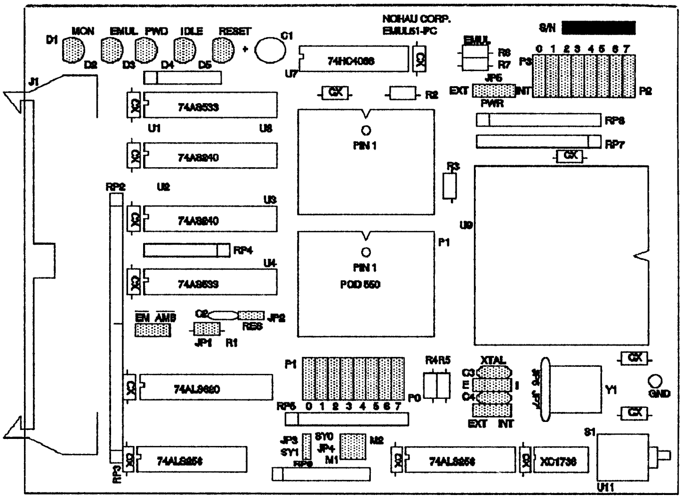


Figure 60. Locations of Jumpers on POD-C550-PGA

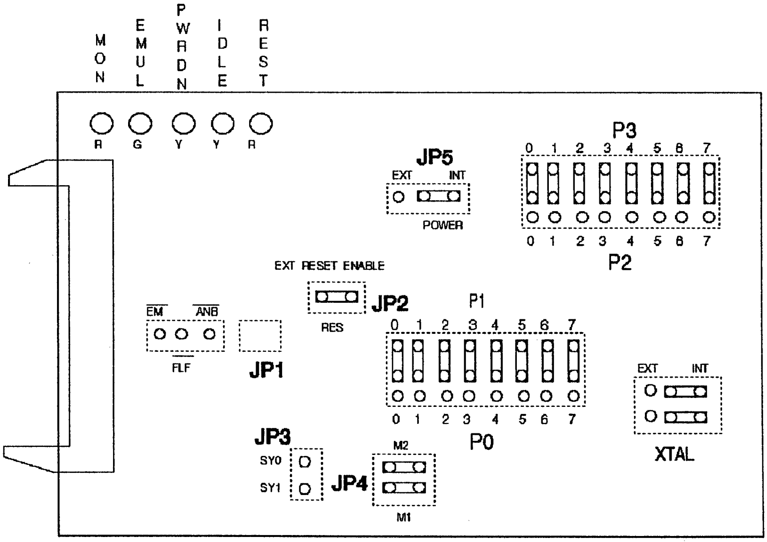


Figure 61. POD-C550-PGA Jumper Diagram

POD-C552-PGA, POD-C562-PGA *POD TYPE: EXTERNAL MODE*

Also higher frequency versions.

This pod has a socket that accepts an 80C552 or 80C562 microcontroller.

Table 15. Designators for POD-C552-PGA, POD-C562-PGA

Designator	*Name*	*Description*
J3	SY0-SY1	These are input signals that can be used for tracing external signals. SY0 can also be used as a Breakpoint.
J4	STADC/EW	These two signals are directly connected to the CPU. In order to trace one of them, you have to install a jumper cable between the desired J4 pin and a free trace input.
JB1	RESET	A jumper installed here will connect the POD-CPU-Reset and the TARGET-Reset signals.
JB2	PSEN	When this switch/jumper is in Position 1, the PSEN signal will only reach the Target system when target code is addressed. When in Position 3, PSEN will always reach the Target system.
JB3	Chip Power	When this switch/jumper is in Position I, the CPU chip will be powered from your PC. When in Position E, the CPU chip will get its power from the Target system.
JB4	Crystal Source	When this switch/jumper is in Position INT, the pod crystal will be connected to the CPU. When in Position EXT, the Target crystal or oscillator will be connected to the CPU.

(Table continued next page)

POD-C552-PGA, POD-C562-PGA

Designator	*Name*	*Description*
JB5 JB6	Trace P1/P4 Trace P3/P5	Installing the JB5 jumpers makes it possible to trace either P1 or P4. Installing the JB6 jumpers makes it possible to trace either P3 or P5. In each group of 3 pins, the middle pin is the trace input. By removing the jumper from this pin and connecting some external signal with a jumper cable (EZ hook), you will cause the external signal to be traced.
TP1	Emulating	This output signal goes high when a Breakpoint is executed or when the program is intercepted by the user. The signal at TP1 can be used to trigger an external instrument such as a logic analyzer.

POD-C552-PGA, POD-C562-PGA

Board Layout Figure 62 shows locations of jumpers and Figure 63 shows a diagram of jumper pins.

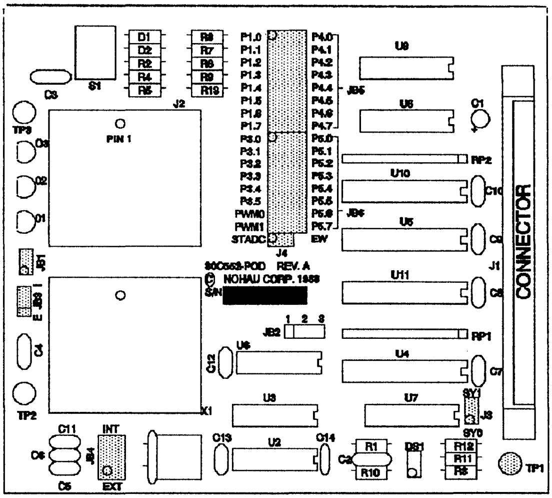
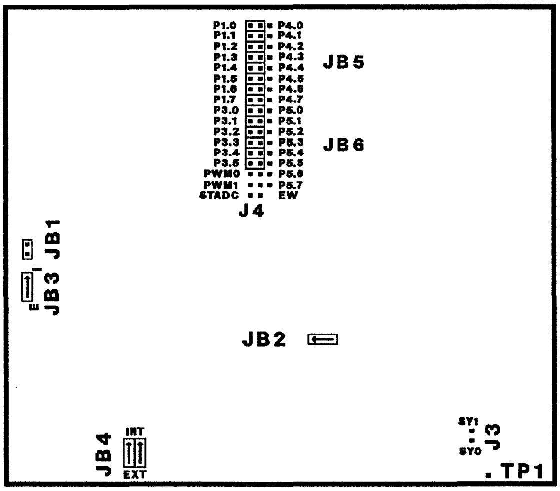


Figure 62. Locations of Jumpers on POD-C552-PGA, POD-C562-PGA



JB3: Shown switched to internal (POD) power

JB4 Pair: Shown switched to internal (POD) crystal

Figure 63. POD-C552-PGA, POD-C562-PGA Configured for internal (POD) power and internal crystal

WATCHDOG APPLICATIONS

PROM - STR FILE METHOD

If you enable the watchdog, use a special EPROM for the emulator board. Use software version 5.5D or higher. Use COM PROM 1.33.

Follow these instructions before you enable the watchdog in your application. Your application will need to service the watchdog.

ENABLE Watchdog control (Example):

1. Install COM EPROM 1.33
2. Rename 80552.STR 80552.OLD
3. Copy 80552WD.STR 80552.STR

DISABLE Watchdog control (Example):

1. Remove COM EPROM 1.33 and reinstall the regular EPROM
2. Delete 80552.STR
3. Rename 80552.OLD 80552.STR

HARDWARE ENABLE METHOD

You can use the 552's EW/ Enable Watchdog input to control the watchdog. The emulator has an output signal EM/ which goes low during emulation.

1. Isolate the EW/signal between your target system and the POD. You must disconnect the ground or low-going driver on the pin. One way is to remove a pin from an extra PGA socket for that purpose.

2. Connect the EM/ signal from TP1 to the EW/ pin on J4.

When you start emulation with a GO, STEP, LINESTEP or GO SLOW command, the signal will be low during emulation and will hardware enable the watchdog. When emulation breaks or is in between steps, the signal will be high and the watchdog will be disabled.

POD-C552B-PGA *POD TYPE: BONDOUT*

If the EPROM on the emulator is version 1.2 or higher there is no need to change it. Otherwise, replace it with the correct PROM.

Table 16. Designators for POD-C552B-PGA (80C/83C/87C552/652 Microcontrollers)

Designator	*Name*	*Description*
PSEN	PSEN Signal	PSEN gated/non-gated: In the non-gated position, the PSEN signal will be active to the user target during emulation if either the EA/ pin is low or the PSEN address is above the top of the memory boundary set with JB7, regardless of MAPC setting. A target EPROM might be unintentionally enabled in this case. The non-gated PSEN output is from the bond-out chip. In monitor (not emulating) mode, the signal will only be active for CBYTE and code window display if the MAPC setting is to target. In the gated PSEN position, during emulation the signal is only active if the MAPC is set to target and either the EA/ pin is low or the PSEN address is above the memory boundary. In monitor mode, the signal will be active to the target for CBYTE and code window display if MAPC is set to target and the EA/ pin is low or the address is above the boundary. The gated PSEN output is from a 74AS32 with a 330 ohm pullup.
RD WR	RD Signal WR Signal	WR & RD gated/non-gated: For P3.6 and P3.7 as ports, use the non-gated position. Non-gated will ignore MAPX settings during emulation. For monitor mode XBYTE commands and external data reads, the signals will be active if MAPX is set to target. The non-gated port lines are directly connected to the bond-out chip.

(Table continued next page)

POD-C552B-PGA

Jumper	*Name*	*Description*
		In the gated positions, the read and write signals will follow the setting of the MAPX in both emulation and monitor modes. Use gated if the lines are used as external memory write and read lines. ENABLE XDATA in the emulator to use the external data window. Each gated output is from a 74AS32 with a 330 ohm pullup.
JB4, JB5	CRYSTAL Selection	With both jumpers installed in the INT position (Figure 65), the pod crystal is connected to the bond-out chip. With jumpers installed in the EXT position, the target crystal or oscillator will be connected to the pod bond-out chip. When emulating NMOS external oscillator (not crystal), the X1 center pin is jumpered to the EXT pin of X2.
JB6	POWER	This jumper selects the source of power for the bond-out chip. If a jumper is installed for INT (Figure 65), the bond-out chip receives its power from the PC. If a jumper is installed in the opposite position (EXT), the bond-out chip will be powered from the target system.
JB7	MEMORY and CPU Selection	On JB7, the two jumpers nearest the ribbon connector select the emulated memory size. Figure 65 shows memory set to 8K. Select 4K by inserting a single jumper on the pins nearest the ribbon cable connector. Select 32K by installing both jumpers; 16K by removing both jumpers. The three pins on JB7 that are away from the ribbon cable select the CPU type. Select 8XC552 by installing jumpers on all three pins on JB7 farthest from the ribbon cable. Select 8XC562 by installing a jumper on the pin farthest from the cable and leaving the other two open. Select 8CX562 with EEPROM by installing jumpers on the two pins farthest from the cable.
RES	Ext Reset Enable	External reset is enabled with the jumper installed.

(Table continued next page)

POD-C552B-PGA

Designator	*Name*	*Description*
JB8	PORT 3 and 4	JB8 selects either Port 3 or 4 to be traced. If a jumper is installed in position P3.4 then port 3 bit 4 will be traced. If no jumper is installed, the middle pin (or pins) can be connected to external signals with the optional EZ-hooks. These signals will then be traced instead of the port signals.
JB9	PORT 1 and 2	These jumpers work exactly like JB8, but on ports 1 and 2
JB14, JB15	I^2C Selection	POD-C552B-PGA will emulate either an 8XC552 or 8XC562 microcontroller. The **8XC552** with I^2C ports is selected exactly as shown in Figure 65. Select **8XC562**, with non-I^2C ports, by inserting jumpers on JB14 and JB15 in the opposite position from that shown in Figure 65. (on the side nearest the center of the board).
JB16		Port 1 and Port 5 signals are available on these pins.
EM	Mode Signal	LOW = Emulation Mode; HIGH = Monitor Mode.
INTA	Interrupt Signal	This pin puts out a negative pulse at the start of an interrupt routine.
C1	Inst. Fetch Signal	The signal on this pin is high when the first byte of an instruction is fetched; it is multiplexed when ALE is high.
SY0, SY1	SY0 and SY1 Inputs	These 2 pins can be connected to trace external signals. SY0 may also be used to trigger breakpoints.
GND	Ground Wire	This point should be connected to the target system GROUND before plugging the pod board into the target system.
M	Monitor LED	Lights RED when in the Monitor mode.
E	Emulation LED	Lights GREEN when in Emulation mode.
P	Power-down LED	Lights ORANGE when the CPU is in Power-down mode.
I	Idle LED	Lights ORANGE when the CPU is in the Idle mode.
R	Reset LED	Lights RED when the CPU receives a RESET signal.

POD-C552B-PGA

"B" stands for Bond-Out

Limited Warranty on Bond-Out Chips: Damage to expensive bond-out chips is not covered under warranty if the proper power-up and power-down sequences are not followed. Please review the power sequencing instructions on Page 4-3.

The bond-out chip in the POD-C552B-PGA is a CMOS chip in a pin-grid-array (PGA) package.

Software Notes: (EMUL51-PC Software Version 5.4H or higher)

The POD-C552B must be invoked with -p80552b. Example:

```
EMUL51 -p80552b -e110 -m128 -t100 -f16   ;typical invocation
```

Internal ROM or EPROM must be mapped to the emulator memory. Due to the bondout chip used in the POD-C552B-PGA there is a quirk in the operation of the serial port. If you hit a breakpoint in the middle of a serial transmission, the character will be cut off. After resuming from the breakpoint, the flag telling that a new character can be sent will be inhibited, as well as the corresponding interrupt, if interrupts are used.

Timers are normally stopped at a breakpoint, and restarted when emulation is resumed. There are currently two exceptions to normal timer operation: 1. Setting a breakpoint at a MOVX instruction causes the timers to miss one tick. 2. When emulation is STARTED in TARGET, the timers will increment one extra tick at the next breakpoint. However, if the next break is at a MOVX instruction the two problems cancel and the timers will be correct.

The EA pin has a 220 k ohm pull up resistor, so if it is left open it will float high.

POD-C552B-PGA

Board Layout

Figure 64 shows locations of jumpers and Figure 65 shows the jumper pins and LED's in detail.

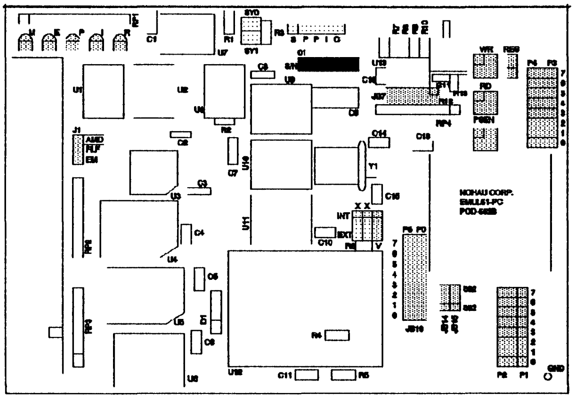
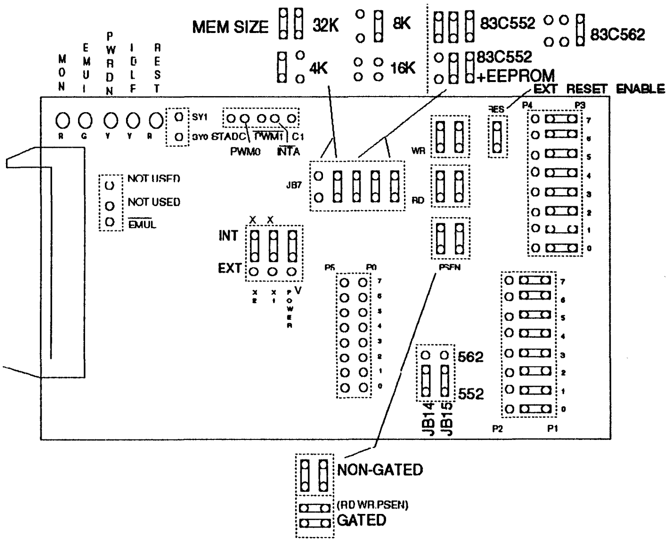


Figure 64. Locations of Jumpers on POD-C552B-PGA

JB7: MEM SIZE selected = 8K RES: External Reset Enable selected



WR, PSEN: shown NON-GATED

RD: shown GATED

JB4, JB5: POD Crystal selected

JB6: Power from PC selected

JB8,
JB9, and JB16: TRACE Ports 1, 3 and 5 (all bits) shown

JB14, JB15: 80C552

Figure 65. POD-C552B-PGA Jumper Diagram

POD-C558-16 *POD TYPE: "HOOKS" MODE*

The software must be 5.8A or later. Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

```
EMUL51 -p83558 -e110 -m128 -t100 -f16  ; typical invocation
```

The POD-C558 adapter is used to emulate Philips microcontrollers 83CE558, and 89CE558.

The EPROM on the emulator board must be version COM 1.4 or later.

The solder side of the board has two 16 pin headers and two 24 pin headers. They are designed to plug into this Emulation Technology adapter for surface mount pads (PQFP package): EPP- 080-QF08-LG.

The Emulation Technology adapters can be ordered from Nohau, or directly from Emulation Technology.

The adapter goes into the target system replacing the microcontroller IC.

The pod board uses a Philips 85C558 chip in a special emulation mode. This emulation mode also uses a programmable gate array from XILINX (LCA).

The pod board has five LEDs, named MON, EMUL, PWRDN, IDLE, and RST:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

POD-C558-16

PWRDN is yellow, and means that the processor is in power down mode.

IDLE is yellow, and means that the microcontroller is in idle mode.

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RST is red, and means that the processor RESET input is high (ON).

Jumper pins marked M1,M2 determine the mode of operation of P0, P2, P3.6 and P3.7. If only internal memory is to be emulated, both jumper tops (M1 and M2) should be in. This enables P0, P2, P3.6 and P3.7 to be used as general purpose ports.

If the upper jumper (M1) is out, P0 and P2 will contain address and data for external code fetches. PSEN will be active. The emulator mapping (MAPC) will determine whether code is read from the emulator or from the target system.

If the lower jumper (M2) is out, P0 and P2 will have address and data for MOVX instructions. P3.6 and P3.7 will be active as RD and WR if data (MAPX) is mapped to the target system.

NOTE:
The mode jumpers M1, M2 can be changed at any time but you have to issue the command RES CHIP so that the LCA can properly switch operational modes.

The board has four jumpers for power and crystal connection to the processor on the pod board:

Jumper PWR, JP6, is used for power to the processor. Power should be supplied from the emulator and the jumper should be across the two upper pins (I). Taking power from the target system, with the jumper across the two lower pins (E) is NOT RECOMMENDED. Power should be on the "internal" position, INT, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout

PODs.) The processor should be powered from the emulator since its operation is closely connected with the LCA.

The jumper marked XT/X1 determines the source of the pin SELXTAL1. If the jumper is in the top position (X1) SELXTAL1 is forced to select XTAL1. If it is in the lower (XT) position, SELXTAL1 comes from the target system.

Jumpers XTAL1 determine if the XTAL1 crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the upper two positions for the crystal on the pod board (INT), and the lower two positions for target supplied crystal or clock (EXT).

All jumpers for PWR and XTAL must be in internal (INT) positions and XT/X1 must be in the X1 position when no target system is connected. This is also the default position.

Rev A boards have a jumper marked X4GND. With this jumper installed pin X4 is grounded. Remove the jumper top if X4 is used.

Jumper pins marked RST connect the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulation and the jumper top should be removed.

SY0 and SY1 are used to connect external signals to the emulator. They are also used for the trace function. The right pin, SY0, can also be used in the breakpoint logic.

Jumper pins EM/, FLF/, and ANB/ carry signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The middle pin FLF/ and the right pin ANB/ and are used with the Advanced Trace Board (ATR). See the documentation for the ATR.

Jumper P1/P0 upper pins carry the processor port 1 pins in order with P1.0 on the left, and P1.7 on the right. Port 1 is connected directly to the processor. The lower pins carry the port 0 with P0.0 on the left and P0.7 on the right. Port 0 is emulated in the LCA. This means that P0 has better sink capability than a processor. Output signals also appear three clock cycles later than on a real part.

POD-C558-16

The middle row of jumper P1/P0 is normally connected to P1 using jumper tops. The signal level on the middle pins will be reflected on the trace display. The board is delivered with jumpers, so that P1 will be traced. External signals can be connected to the middle pins if the jumpers to P1 are removed. The input load is one ALS input (74ALS258).

Jumper P3/P2 upper pins carry the processor port 3 pins in order with P3.0 on the left, and P3.7 on the right. Port 3 is connected directly to the processor, except for P3.6 and P3.7 which go through a 74HC4066 that has an approximately 100 ohm series resistance. The lower pins carry the port 2 with P2.0 on the left and P2.7 on the right. Port 2 is emulated in the LCA. This means that P2 has better sink and source capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P3/P2 is normally connected to P3 using jumper tops. It can be used for P2 or other external signals the same way as P1/P0. The input load is one ALS input (74ALS258).

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Note:
Ports P0, and P2 are emulated, and have slightly different AC and DC characteristics:

Emulated I/O pins sink minimum 2 mA, and source (P2) minimum 4 mA for two clock cycles after an output low to high transition. As high outputs or inputs, P2 ports have a 22 kohm pull-up resistor.

Emulated outputs will change state three clock cycles later than a normal output.

P3.6 and P3.7 go through a 74HC4066 which adds about 100 ohms to their output impedance.

When accessing target memory, addresses on P2 and P0 will be delayed approximately one clock cycle. The falling edge of ALE will also be delayed about one clock cycle. This is caused by the special emulation mode of the 85CE558 chip.

Signals ADEXS and EW/ signals have a 220 kohm pullup resistor on the pod.

The silk screen may say STADC instead of ADEXS on some pods.

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

The trace of accesses to the 83C558 AUX-RAM will contain correct addresses when using DPTR, but only the low address will be correct using Ri. The high address will always show zero for any value of XRAMP. Data will be correct for writes but not for reads. If your emulator has 128k RAM and data (MAPX) is mapped to emulator, the data will be written to the emulator RAM. For reads, the data displayed is the emulator RAM data if data is mapped to emulator. This data may NOT be the same as in the AUX-RAM unless it has already been written to the emulator RAM. This method will not work if XRAMP is used since the high address may be incorrect.

If your emulator board only has 32k RAM, data (MAPX) should normally be mapped to target.

If you break emulation using the 'Esc' key, and IDLE or PWRDN is lit, a number of "timeout error" messages will be displayed. This is normal, and you must issue the command RESET CHIP before you do anything else.

Connecting the pod board to the target system:

Solder the Emulation Technology adapter to your target system instead of the microprocessor.

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

POD-C558-16

Plug the pod into the adapter. Make sure pin 1 is in the correct position.

VERY IMPORTANT: Turn on power to the PC first, and then to the target system.

VERY IMPORTANT: When turning off power, turn off the target first, then the PC.

POD-C558-16

Board Layout Figure 66 shows locations of jumpers and Figure 67 shows the jumper pins in detail.

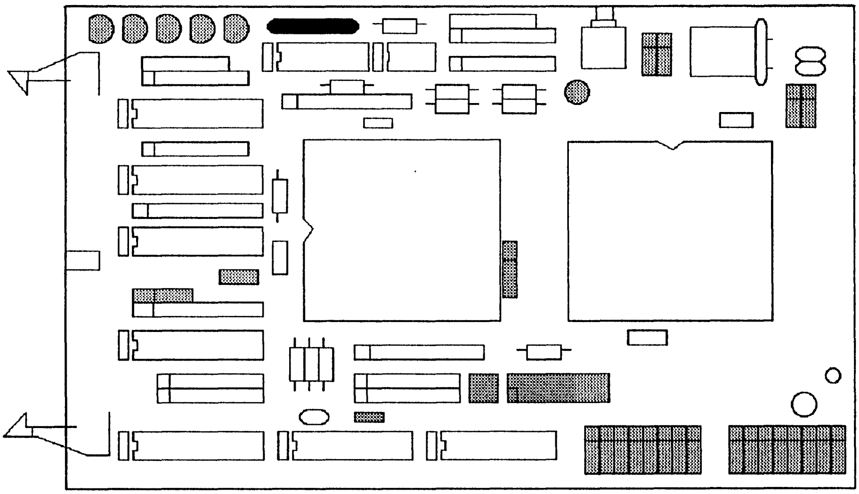


Figure 66. Location of Jumpers on POD-C558-16

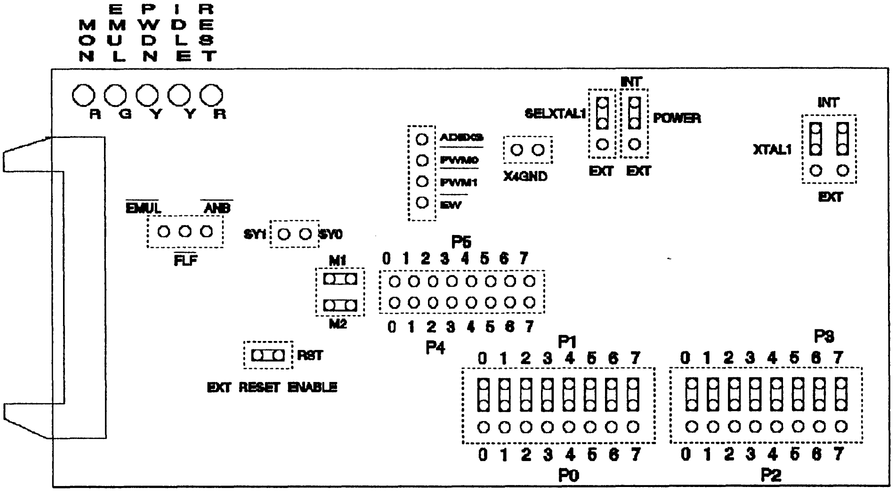


Figure 67. POD-C558-16 Jumper Diagram

POD-C575 *POD TYPE: "HOOKS" MODE*

The software must be 5.7N or later. Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

Example:

```
EMUL51 -p80575 -e110 -m128 -t100 -f16  ; typical invocation
```

The POD-575 adapter is used to emulate Philips / Signetics microcontrollers 80C575, 83C575, and 87C575.

The EPROM on the emulator board must be version COM 1.4 or later. If it is older replace it with the EPROM enclosed with the pod.

The solder side of the board has a 40 pin DIP plug. The plug goes into the target system replacing the microcontroller IC.

If a 44 pin PLCC is used, an adapter can be supplied.

The POD has a Philips/Signetics 87C575 chip using a special emulation mode. This emulation mode usesg a programmable gate array from XILINX (LCA).

The pod board has five LEDs, named MON, EMUL, PWD, IDLE, and RES:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

PWD is yellow, and means that the processor is in power down mode.

IDLE is yellow, and means that the microcontroller is in idle mode.

POD-C575

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RES is red, and means that the processor RESET input is high (ON).

JP5 determines the mode of operation of P0, P2, P3.6 and P3.7. If only internal memory is to be emulated, both jumper tops (M1 and M2) should be in. This enables P0, P2, P3.6 and P3.7 to be used as general purpose ports.

If the lower jumper (M1) is out, P0 will have address and data and P2 will have address or Port 2 data for external code fetches. PSEN will be active. The emulator mapping (MAPC) will determine whether code is read from the emulator or from the target system.

If the upper jumper (M2) is out, P0 will have address and data and P2 will have address or Port 2 data for MOVX instructions. P3.6 and P3.7 will be active as RD and WR if data (MAPX) is mapped to the target system.

NOTE:
The mode jumpers M1, M2 can be changed at any time but you have to issue the command RES CHIP so that the LCA can properly switch operational modes.

The board has three jumpers for power and crystal connection to the bond out chip:

Jumper PWR, JP6, is used for power to the processor. Power should be supplied from the emulator and the jumper should be across the two left pins (INT). Taking power from the target system, with the jumper across the two right pins (EXT) is NOT RECOMMENDED. Power should be on the "internal" position, INT, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout PODs.) The processor should be powered from the emulator since its operation is closely connected with the LCA.

POD-C575

Jumpers XTAL determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the right two positions for the crystal on the pod board (INT), and the left two positions for target supplied crystal or clock (EXT). The oscillator is on the LCA, so do NOT drive the clock inputs when the emulator is powered off.

If you use external clock, note that XTAL1 (pin 19) is input, and XTAL2 (pin 18) should be left open.

EXTERNAL NMOS DRIVE To accommodate external driving OSC signal (not a crystal) designed for NMOS parts, you can do either of the following: Change the target wiring so that STAL1 (pin 19) is the input and XTAL2 (pin 18) is open; or, you can cross-connect the XTAL jumpers to accomplish the same result. See the "NMOS external OSC connection" illustration in Figure 69. If you use a jumper, you will have to bend the pins slightly. The jumper block is 0.1 inches but the diagonal spacing between the pins in 0.14 inches. Alternatively, you can wire wrap a connection between the pins. Remember to not drive the clock input when the emulator is powered off.

All jumpers for PWR and XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

JP7 (RST) connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulation and JP7 should be removed.

JP2 is used to connect external signals to the emulator. They are used for the trace function. The right pin, SY0, can also be used in the breakpoint logic. The name of the left pin is SY1. The silk screen may be wrong on early boards.

JP1 carries signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The other pins are not used currently, but may be used in the future.

POD-C575

Jumper P1/P0 upper pins carry the processor port 1 pins in order with P1.0 on the left, and P1.7 on the right. Port 1 is connected directly to the processor. The lower pins carry the port 0 with P0.0 on the left and P0.7 on the right. Port 0 is emulated in the LCA. This means that P0 has better sink capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P1/P0 is normally connected to P1 using jumper tops. The signal level on the middle pins will be reflected on the trace display. The board is delivered with jumpers, so that P1 will be traced. External signals can be connected to the middle pins if the jumpers to P1 are removed. The input load is one ALS input (74ALS258).

Jumper P3/P2 upper pins carry the processor port 3 pins in order with P3.0 on the left, and P3.7 on the right. Port 3 is connected directly to the processor, except for P3.6 and P3.7 that go through a 74HC4066 that has an approximately 100 ohm series resistance. The lower pins carry the port 2 with P2.0 on the left and P2.7 on the right. Port 2 is emulated in the LCA. This means that P2 has better sink and source capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P3/P2 is normally connected to P3 using jumper tops. It can be used for P2 or other external signals the same way as P1/P0. The input load is one ALS input (74ALS258).

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Note:
Ports P0, and P2 are emulated, and have slightly different AC and DC characteristics:

Emulated I/O pins sink minimum 4 mA, and source (P2) minimum 4 mA for two clock cycles after an output low to high transition. As high outputs or inputs, P2 ports have a 22 kohm pull-up resistor.

Emulated outputs will change state three clock cycles later than a normal output.

POD-C575

If your emulator board only has 32k RAM, data (MAPX) should normally be mapped to target.

P3.6 and P3.7 go through a 74HC4066 which adds about 100 ohms to their output impedance.

When accessing target memory, P2, P0, and ALE will be delayed approximately 25 ns.

The clock circuitry is emulated, but is very close to the processor clock.

The silk screen may be reversed for SY0 and SY1. SY1 is on the left.

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

If you break emulation using the 'Esc' key, and IDLE or PWD is lit, a number of "timeout error" messages will be displayed. This is normal, and you must issue the command RESET CHIP before you do anything else.

If you for some reason want to change processor, you have to use a Philips / Signetics 87C575.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-C575

Board Layout

Figure 68 shows locations of jumpers and Figure 69 shows the jumper pins in detail.

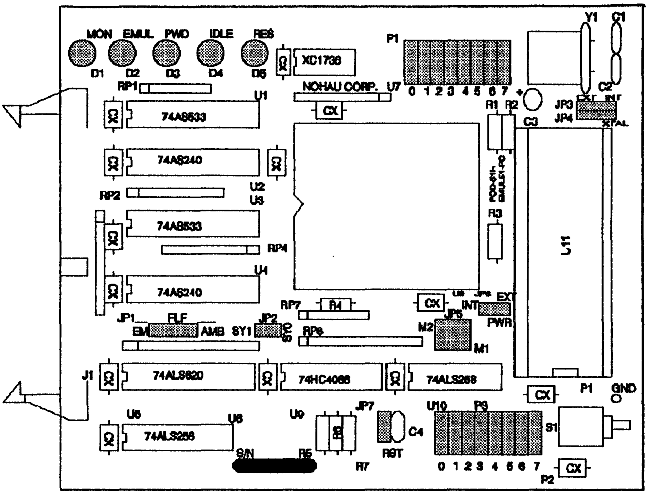


Figure 68. Locations of Jumpers on POD-C575

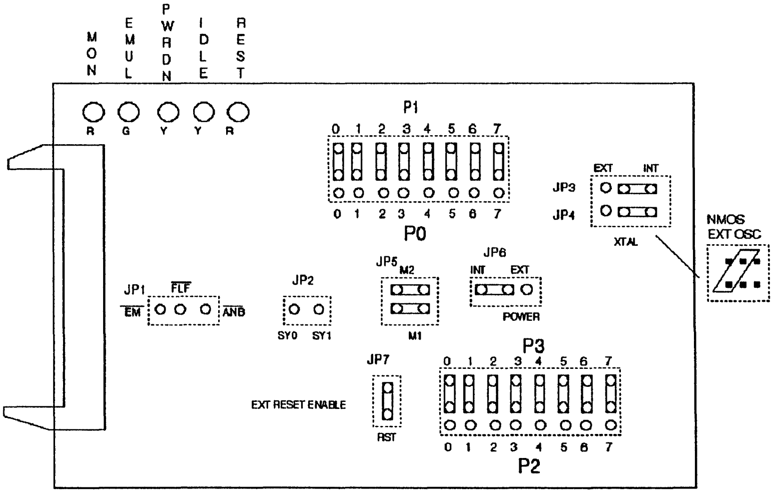


Figure 69. POD-C575 Jumper Diagram

POD-CL580 *POD TYPE: BONDOUT*

The software must be 5.5R or later. Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

Example:

```
EMUL51 -p83410 -e110 -m128 -t100 -f16        ; typcial invocation
```

The pod board is used to emulate Philips/Signetics' PCF83CL580.

The EPROM on the emulator board must be version COM 1.2 or later. If you have an older one, please request a new one from Nohau.

The solder side of the board has two 14 pin headers and two 20 pin headers. They are designed to plug into the Emulation Technology adapter EPP-064-QF09-LG for 64 pin surface mount pads.

The bottom portion of the adapter solders onto the target system replacing the microcontroller IC.

Emulation Technology adapters can be ordered from Nohau, or directly from Emulation Technology.

The pod board has a Philips PCF86C001WP bondout chip and a P83CL580/800 derivative chip to emulate the microcontroller chip.

The pod board has five LEDs marked MON, EMUL, PWRDN, IDLE, and RES:

MON is red, and if lit means that the system is in monitor mode. In monitor mode, the bondout chip is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and if lit means that the system is in emulation mode. In emulation mode the bondout chip is executing user code. This code is normally in a special RAM on the emulator board.

PWRDN is orange, and if lit means that the system is in power down mode.

POD-CL580

An interrupt or reset can take it out of this mode.

IDLE is orange, and if lit means that the system is in idle mode. An interrupt or reset can take it out of this mode.

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RES is red, and if lit means that the bondout chip RESET input is high (ON).

The top jumper marked POWER close to the crystal is used for power to the bondout chip. If the power is to be supplied from the emulator (5 volts), the jumper should be across the two right pins (INT). If power is to come from the target system (1.5 - 5 volts), the jumper should be across the two left pins (EXT).

The bottom jumpers marked XTAL close to the crystal determine if the crystal or clock is from the target system, or if the crystal on the pod is used. The jumper should be in the right two positions for the crystal on the pod board (INT), and the left two positions for target supplied crystal or clock (EXT).

The jumpers above must be in internal (INT) positions when no target system is connected. This is also the default position.

The jumper marked RES connects the target RESET pin to the emulator. If the target system has a watch dog, it will probably interfere with monitor mode and the jumper should be removed.

The jumper pins marked SY0 and SY1 close to the LEDs are used to connect external signals to the emulator. They are used for the trace function. The lower pin, SY0, can also be used in the breakpoint logic. The name of the upper pin is SY1.

In the left lower side of the board are three jumper pins: The left pin EM is high in monitor mode, and low in emulation mode. The other pins FLF, and ANB are used with the advanced trace board.

POD-CL580

For jumpers P2/P3: the top row of pins is connected to port P3, in order, with P3.7 at the right and P3.0 at the left. The bottom row of pins is connected to port P2, with P2.7 at the right and P2.0 at the left. The middle row of pins is reflected on the trace display of P3. The board is delivered with jumpers so that P3 is traced. If the jumpers are moved to the bottom row, P2 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

For jumpers P0/P1: the top row of pins is connected to port P1, in order, with P1.7 at the right and P1.0 at the left. The bottom row of pins is connected to port P0, with P0.7 at the right and P0.0 at the left. The middle row of pins is reflected on the trace display of P1. The board is delivered with jumpers so that P1 is traced. If the jumpers are moved to the bottom row, P0 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

On early versions of the board the text P0 and P1 is swapped.

Jumper pins P4 on top of the board carry the signals of P4 in order with P4.0 on the right. The very left pin is the signal PWM0.

Jumper INTA/C1 just below the bondout chip has the bondout chip signals INTA (interrupt acknowledge) and C1 (first op code fetch).

If you want to trace C1 or INTA the corresponding pins must be connected to the "middle row" of P2/P3 jumper rows thereby replacing those ports in the trace display.

JB1 close to the pushbutton is used to set the amount of internal memory emulated. It is also used to emulate the "ROM PROTECT" feature. See Figure 71.

JB2 close to the crystal should be in the left position for the current version of the derivative chip.

S1 pushbutton is used to reset the bondout chip, and can be used instead of the target system reset.

Connecting the pod board to the target system:

The adapter consists of two subassemblies. Solder in the lower part of the adapter onto the target board replacing the microcontroller. Please observe the instructions enclosed with the adapter.

Plug the upper part of the adapter on top of the soldered-in adapter.

Make sure that the power is turned off in both the host PC, and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Plug the board carefully into the second adapter. With the pod board oriented so the long ribbon cable goes out to the left. Make sure that the target system pin 1 is towards the top of the board.

Turn on power to the PC first, and then to the target system.

When turning power off, turn the target off first, then the PC.

Note:
If the target system is running at a voltage lower than 5 volts, the power jumper should be in the EXT position. In this situation the bondout chip and the derivative chip will be powered from the target system. Since the voltage level translation circuitry uses some power from the bondout chip signal lines, the current consumption will be higher than for a "real" chip. The current may be up to 10 mA.

Since the P83CL580 processor is mainly intended for ROM code with P0 and P2 used as port pins, and because of the amount of circuitry involved for voltage level translation, the emulation support for external memory access is limited.

POD-CL580

MOVX instructions are not fully supported. XDATA can not be mapped to the emulator. Do not use breakpoints on MOVX read or write cycles. The address and data for MOVX instructions is not correct in the trace display. CODE can be executed in the target system, but breakpoints will not work properly. The trace will show addresses correct, but the actual code bytes read from the target system will be incorrect. So CODE should normally be mapped to the emulator memory.

Bondout chip and derivative chip bugs:

Early versions of the bondout chip may have the following problems:

Writing a zero to a port 2 pin that already is low will generate a positive pulse, one clock cycle wide.

Writing a one to a port 0 pin that was zero will generate a positive pulse, two clock cycles wide before the pin goes into high impedance state.

Power down locks up the system for early versions of the derivative chip. The target and PC has to be powered of to restart the system.

POD-CL580

Board Layout

Figure 70 shows locations of jumpers and Figure 71 shows the jumper pins in detail.

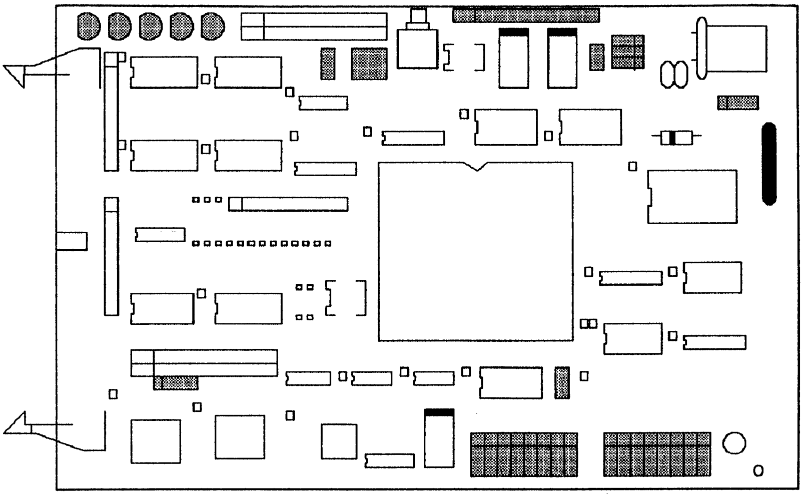


Figure 70. Locations of Jumpers on POD-CL580

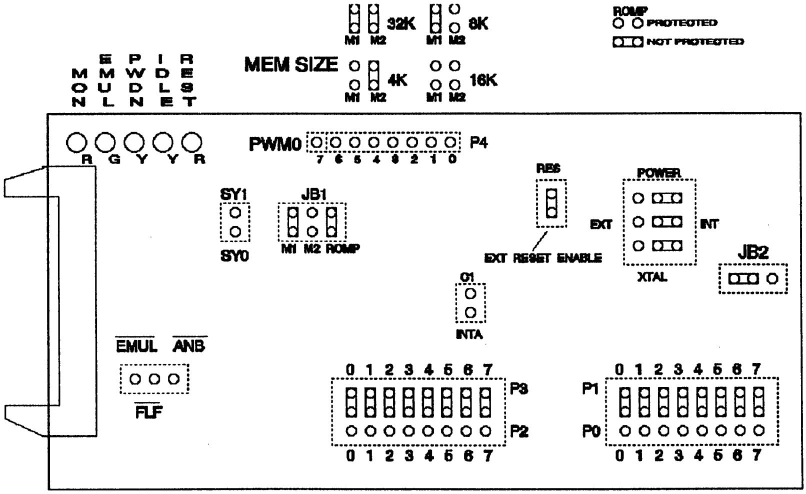


Figure 71. POD-CL580 Jumper Diagram

POD-C592-PGA *POD TYPE: "HOOKS" MODE*

Also higher frequency versions.

The software must be 5.6A or later. Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

Example:

```
EMUL51 -p80592 -e110 -m128 -t100 -f16     ; typical invocation with 128k
                                          emulator
EMUL51 -p80592 -e110 -m32 -t100 -f16      ; typical invocation with 32k
                                          emulator
```

The POD-C592-PGA adapter is used to emulate Philips microcontrollers 80C592, 83C592, and 87C592.

The EPROM on the emulator board must be version COM 1.4 or later. If it is older replace it with an EPROM enclosed with the pod.

The solder side of the board has a 68 pin PGA plug. It is recommended that you use a PGA socket in your target system for mechanical stability. If a 68 pin PLCC is used, an optional adapter can be supplied.

The pod board uses a Signetics 80C592 chip in a special emulation mode. This emulation mode also uses a programmable gate array from XILINX (LCA).

The pod board has five LEDs, named MON, EMUL, PD, IDLE, and RST:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

PD is yellow, and means that the processor is in power down mode.

POD-C592-PGA

IDLE is yellow, and means that the microcontroller is in idle mode.

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RST is red, and means that the processor RESET input is high (ON).

Jumper pins marked M1 and M2 determine the mode of operation of P0, P2, P3.6 and P3.7. If only internal memory is to be emulated, both jumper tops (M1 and M2) should be in. This enables P0, P2, P3.6 and P3.7 to be used as general purpose ports.

If the left jumper (M1) is out, P0 and P2 will contain address and data for external code fetches. PSEN will be active. The emulator mapping (MAPC) will determine whether code is read from the emulator or from the target system.

If the right jumper (M2) is out, P0 and P2 will have address and data for MOVX instructions. P3.6 and P3.7 will be active as RD and WR if data (MAPX) is mapped to the target system.

NOTE:
The mode jumpers M1, M2 can be changed at any time but you have to issue the command RES CHIP so that the LCA can properly switch operational modes.

The board has three jumpers for power and crystal connection to the bond out chip:

Jumper PWR, JP6, is used for power to the processor. Power should be supplied from the emulator and the jumper should be across the two upper pins (INT). Taking power from the target system, with the jumper across the two lower pins (EXT) is NOT RECOMMENDED. Power should be on the "internal" position, INT, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout PODs.) The processor should be powered from the emulator since its operation is closely connected with the LCA.

POD-C592-PGA

Jumpers XTAL determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the right two positions for the crystal on the pod board (INT), and the left twopositions for target supplied crystal or clock (EXT). The oscillator is on the LCA, so do NOT drive the clock inputs when the emulator is powered off.

All jumpers for PWR and XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

Jumper pins marked RST connect the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulation and the jumper top should be removed.

SY0 and SY1 are used to connect external signals to the emulator. They are also used for the trace function. The lower pin, SY0, can also be used in the breakpoint logic.

Jumper pins EM/, FLF/, and ANB/ carry signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The other pins are not used currently, but may be used in the future.

Jumper P1/P0 upper pins carry the processor port 1 pins in order with P1.0 on the left, and P1.7 on the right. Port 1 is connected directly to the processor. The lower pins carry the port 0 with P0.0 on the left and P0.7 on the right. Port 0 is emulated in the LCA. This means that P0 has better sink capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P1/P0 is normally connected to P1 using jumper tops. The signal level on the middle pins will be reflected on the trace display. The board is delivered with jumpers, so that P1 will be traced. External signals can be connected to the middle pins if the jumpers to P1 are removed. The input load is one ALS input (74ALS258).

POD-C592-PGA

Jumper P3/P2 upper pins carry the processor port 3 pins in order with P3.0 on the left, and P3.7 on the right. Port 3 is connected directly to the processor, except for P3.6 and P3.7 which go through a 74HC4066 that has an approximately 100 ohm series resistance. The lower pins carry the port 2 with P2.0 on the left and P2.7 on the right. Port 2 is emulated in the LCA. This means that P2 has better sink and source capability than a processor. Output signals also appear three clock cycles later than on a real part.

The middle row of jumper P3/P2 is normally connected to P3 using jumper tops. It can be used for P2 or other external signals the same way as P1/P0. The input load is one ALS input (74ALS258).

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Note:
Ports P0, and P2 are emulated, and have slightly different AC and DC characteristics:

Emulated I/O pins sink minimum 4 mA, and source (P2) minimum 4 mA for two clock cycles after an output low to high transition. As high outputs or inputs, P2 ports have a 22 kohm pull-up resistor.

Emulated outputs will change state three clock cycles later than a normal output.

If you use external clock, note that XTAL1 (pin 34) is input, and XTAL2 (pin 33) should be left open.

P3.6 and P3.7 go through a 74HC4066 which adds about 100 ohms to their output impedance.

When accessing target memory, P2, P0, and ALE will be delayed approximately 25 ns.

The clock circuitry is emulated, but is very close to the processor clock.

POD-C592-PGA

The timer/counters are stopped at breakpoints. This usually also means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

If you break emulation using the 'Esc' key, and IDLE or PWD is lit, a number of "timeout error" messages will be displayed. This is normal, and you must issue the command RESET CHIP before you do anything else.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

VERY IMPORTANT: Turn on power to the PC first, and then to the target system.

VERY IMPORTANT: When turning off power, turn off the target first, then the PC.

POD-C592-PGA

Board Layout Figure 72 shows locations of jumpers and Figure 73 shows the jumper pins in detail.

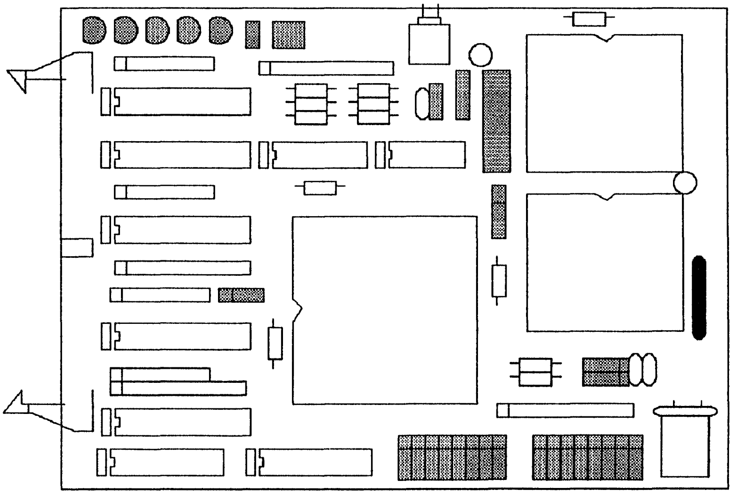


Figure 72. Locations of Jumpers on POD-C592-PGA

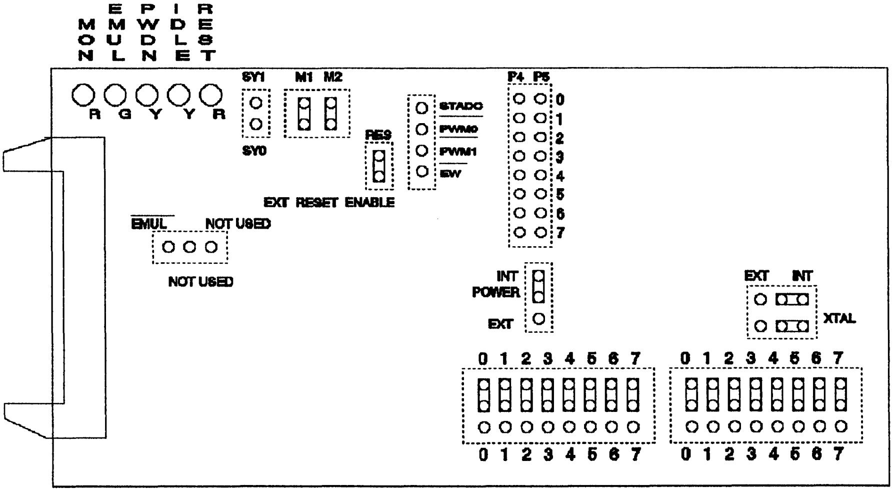


Figure 73. POD-C592-PGA Jumper Diagram

POD-C598-KIT

This "kit" to emulate the 8XCE598 consists of an existing POD plus two adapters. The POD-C592-PGA or POD-C592-PGA-16 can be used to emulate the 8XCE598. The adapters enable you to connect the POD to the 80-pin quad flat pack (QFP) surface mount pattern.

The adapters connect to each other. Both are required.

EDI/80-ET/68PG-592-598

This first adapter is an intermediate assembly. The 68-pin socket on top accepts the pin grid array (PGA) plug from the POD-C592-PGA. This unit plugs into the bottom adapter.

ET/EPP-080-QF08-LG

This second adapter connects to the user's board. It solders onto the 80-pin QFP pattern on the board. Then the top adapter plugs into it.

Consult the data sheets for the 8XC592 and the 8XCE598 for differences between the parts. For example, the 8XCE598 has a feature that allows software to disable the ALE signal. Since the 8XC592 and the POD-C592-PGA do not have this feature, this feature is not emulated.

Refer to the POD-C592-PGA description for operation of the POD.

POD-C751 *POD TYPE: BONDOUT*

Also higher frequency versions.

POD-C751 is used to emulate Signetics microcontrollers 83C751 and 87C751. The EPROM on the emulator board must be version 1.2 or later. If it is older, replace it.

The processor type parameter for invocation is -p80751.

The solder side of the board has a 24 pin DIP plug which is inserted in the target board in place of the microcontroller IC. If a 28 pin PLCC is used, an adapter can be supplied. If the pod board can not be attached directly to the target system, a 24-pin DIP extender cable is available. However, in this case using a target system crystal for the oscillator may be unreliable.

Limited Warranty on Bond-Out Chips: POD-C751 uses a Signetics 85C751 bond out chip to emulate the microcontroller. Damage to expensive bond-out chips is not covered under warranty if the proper power-up and power-down sequences are not followed. Please review the power sequencing instructions on Page 4-3.

Note: The 83C751 does NOT support MOVX instructions. The emulator may get confused, trying to execute a MOVX instruction, and display a time out error message, or just lock up. Often both the MON and EML LED's will come on in this case. If there is no keyboard response, the reset button can be used to get the prompt back. Several resets may be necessary. Use the RESET CHIP command to establish communication again. Examine the code for illegal instructions. The trace information may have frames with address 3000 - 37FF. They should be ignored.

POD-C751

Table 17. Designators for POD-C751

Designator	Name	Description
JB1	SY0 & SY1	These pins are used to connect external signals to the emulator. They are used for the trace function. SY0 can also be used in the breakpoint logic.
JB2	EMUL Signal	The EM pin (bottom one of three) is LOW in emulation mode, and high in monitor mode. The other two pins in JB2 are not currently used.
JB3	POWER	The source of power to the bondout chip is selected by JB3. With the jumper in the "I" position, power is supplied by the emulator; with it in the "E" position, power from the target system is used.
JB4	Port 3	With all eight jumpers in JB4, Port 3 bits 0-7 are traced. With the jumpers removed, external signals can be traced by connecting to the row of pins farthest from the processor. Input load in this case is one ALS input (74ALS258).
JB5	Port 1	JB5 is used in the same manner as JB4, but for tracing Port 1 (or external signals).
JB6	Port 0	The three pins farthest from the ribbon connector provide access to the bond out chip Port 0 pins P0.0 - P0.2. The JB3 jumpers on the three low order pins may be moved so that P0 is traces instead of P3.0 - P3.2.
JB6	INITA/	The second pin from the left on JB6 carries the INITA/ signal from the bondout chip. This signal goes low for two frames while the chip internally calls the interrupt vector.
JB6	C1	The left pin in JB6 is C1, which goes high when the chip fetches the first byte of an instruction. It is also high in the first frame of an interrupt internal call.
JB7, JB8	Crystal Select	JB7 & JB8 determine if the crystal or clock is from the target system, or if the crystal on the pod is used. The jumper should be in the upper two positions for the crystal on the pod board (I), and the lower positions for target supplied crystal or clock (E).
JB9	Ext. RESET Enable	JB9 connects the target RESET pin to the emulator. If the target system has a watch dog, it will probably interfere with the emulation and JB 9 should be removed.
S1	RESET	The S1 push button is used to reset the boundout chip; can be used instead of target system reset.

(Table continued next page)

POD-C751

Designator	*Name*	*Description*
MON	Monitor LED	Lights red when in the monitor mode.
EML	Emulation LED	LIghts green when in emulation mode.
PD	Power Down LED	Lights yellow while in the power down mode. To restart, a reset must be issued, either from the target system, from the PC keyboard, or by pressing the S1 button.
IDL	Idle Mode LED	Lights yellow when the microcontroller is in the idle mode. An interrupt or reset will take the chip out of this mode.
RST	Reset LED	Lights red when the microcontroller RESET input is high (active).

All jumpers in JB3, JB7, and JB8 must be in the internal (I) position when no target system is connected.
This is also the default position.

4

POD-C751

Board Layout Figure 74 shows locations of jumpers and Figure 75 shows the jumper pins in detail.

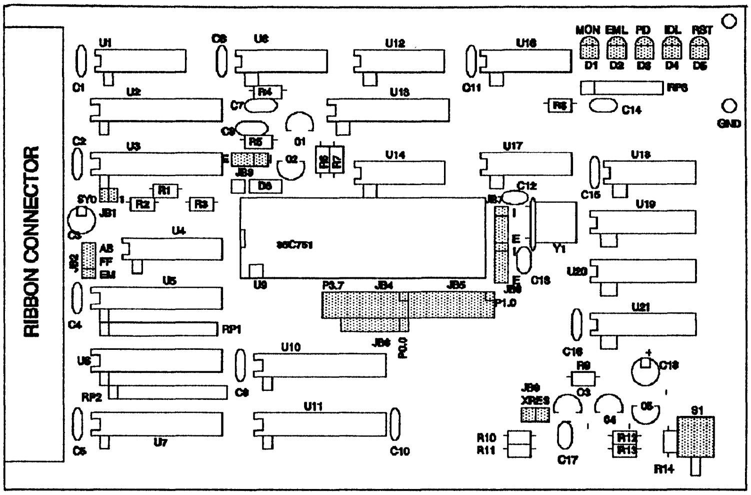


Figure 74. Locations of Jumpers on POD-C751

JB3: Power from PC Selected

JB7, JB8: Internal Crystal selected

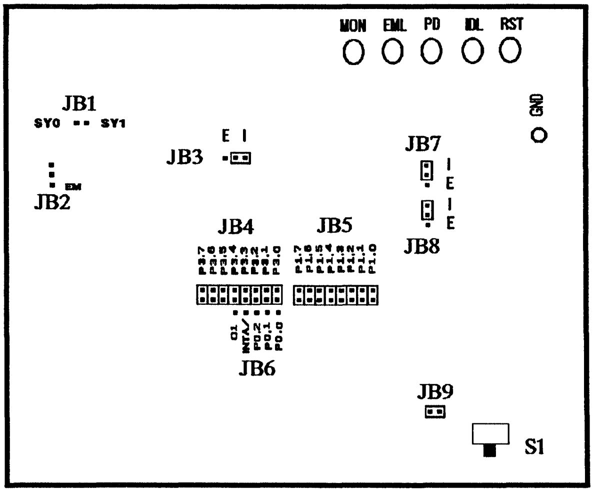


Figure 75. POD-C751 Jumper Diagram

POD-C752 *POD TYPE: "HOOKS" MODE*

The software must be 5.49C or later. Invoke the EMUL51-PC as in this example:

```
EMUL51 -p80752 -e110 -m32 -t100 -f16   ;typical invocation
```

The pod board is used to emulate Signetics microcontrollers 83C752 and 87C752.

The EPROM on the emulator board must be version COM 1.3 or later. If it is older replace it with the EPROM enclosed with the pod.

The solder side of the board has a 28 pin DIP plug. The plug goes into the target system replacing the microcontroller IC.

If a 28 pin PLCC is used, an adapter can be supplied.

If the pod board can not be attached directly to the target system, an extender cable 28 pin DIP to 28 pin DIP can be supplied. In this case, using a target system crystal for the oscillator may be unreliable.

The pod board has a Signetics 87C752 chip using a special emulation mode. This emulation mode is also using a programmable gate array from XILINX (LCA).

The pod board has five LEDs named MN, EM, PD, ID, and RST:

MN is red, and means that the system is in monitor mode. In monitor mode, the 87C752 is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EM is green, and means that the system is in emulation mode. In emulation mode the 87C752 is executing user code. This code is in a special RAM on the emulator board.

PD is yellow, and means that the 87C752 is in power down mode. To get out of this mode, a reset has to be issued. Reset can be issued from the target system, from the reset button (S1) or from the PC keyboard.

ID is yellow, and means that the microcontroller is in idle mode. Only a reset will take the chip out of this mode. Because of limitations in the 87C752, the emulation can not continue after interrupt.

RST is red, and means that the 87C752 RESET input is high (ON).

The board has three jumpers for power and crystal connection to the bond out chip:

Jumper PWR, JP6, is used for power to the processor. Power should be supplied from the emulator and the jumper should be across the two lower pins (I). Taking power from the target system, with the jumper across the two upper pins (EXT) is NOT RECOMMENDED. Power should be on the "internal" position, INT, even if the POD is connected to a user target system. (Note that this selection is the opposite of that for the bondout PODs.) The processor should be powered from the emulator since its operation is closely connected with the LCA.

Jumpers XTAL determine if the crystal or clock is from the target system, or if the crystal on the pod is used. The jumpers should be in the upper two positions for the crystal on the pod board (INT), and the lower two positions for target supplied crystal or clock (EXT). The oscillator is on the LCA, so do not drive the clock inputs, when the emulator is powered off.

All jumpers for PWR and XTAL must be in internal (I) positions when no target system is connected. This is also the default position.

JP13 connects the target RESET pin to the emulator. If the target system has a watch dog, it will probably interfere with the emulation and JP13 should be removed.

JP14 is used to connect external signals to the emulator. They are used for the trace function. The right pin, SY0, can also be used in the breakpoint logic. The name of the left pin is SY1.

POD-C752

The jumper pins between U2 and U3 carry signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The other pins are not used currently, but may be used in the future.

Jumper P3 left row of pins carry the emulated port 3 pins in order, with P3.0 at the bottom and P3.7 at the top. The signal level on the right pins will be reflected on the trace display. The board is delivered with jumpers, so that P3 will be traced. External signals can be connected to the right pins if the jumpers to P3 are removed. The input load is one ALS input (74ALS258).

Jumper P1/P0 upper pins carry the 80C752 port 1 pins in order with P1.0 on the left, and P1.7 on the right. P1.0 - P1.4 are connected directly to the 87C752, while P1.5 - P1.7 are emulated using the LCA. The lower pins carry the P0.0 - P0.4 with P0.0 on the left. P0.3 is emulated, the others are connected to the 87C752 directly.

The right part of jumper P1/P0 lower pins consists of some non port signals: R is the reset signal going to the 87C752. C is a signal indicating first opcode fetch. It is high at first opcode fetch, and low at other code fetches. E is high in emulation mode, and low in monitor mode.

The middle row of jumper P1/P0 is normally connected to P1 using jumper tops. It can be used for P0 or other external signals the same way as P3. The input load is one ALS input (74ALS258).

JP15 and JP16 are factory set and should not be changed.

S1 push-button is used to reset the 87C752, and can be used instead of target system reset.

Note:
Idle mode is not fully supported. Reset has to be used to get out of idle mode.

Ports P3, P1.5 - P1.7, and P3.0 are emulated, and have slightly different AC and DC characteristics:

POD-C752

Emulated I/O pins sink minimum 4 mA, and source minimum 4 mA for two clock cycles after an output low to high transition. As high outputs or inputs they have a 220 kohm pull-up resistor.

Emulated outputs will change state three clock cycles later than a normal output. INT0, INT1, and T0 inputs will be sensed two clock cycles later that normally.

The clock circuitry is emulated, but is very close to the 80C752 clock.

The delay from reset low to processor start is longer than in the normal part. The delay is about 80 clock cycles.

The timer/counter is stopped at breakpoints, but counts one step extra, when used with internal clock.

A reset during emulation will generate about five "garbage" frames in the trace buffer. Idle mode will generate about 16 "garbage" frames.

If you break emulation using the 'Esc' key, and ID or PD is lit, a number of "timeout error" messages will be displayed. This is normal, and you must issue the command RESET CHIP before you do anything else.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Plug the board carefully into the target system. With the pod board oriented so the long ribbon cable goes out to the left, make sure that the target system ground is on the lower right pin, and +5V is on the upper left pin of the 24 pin DIP.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-C752

Note: The 87C752 does NOT support MOVX, LJMP, and LCALL instructions. The emulator should behave reasonably if you try to execute an illegal instruction, but it is not guaranteed.

POD-C752

Board Layout

Figure 76 shows locations of jumpers and Figure 77 shows the jumper pins in detail.

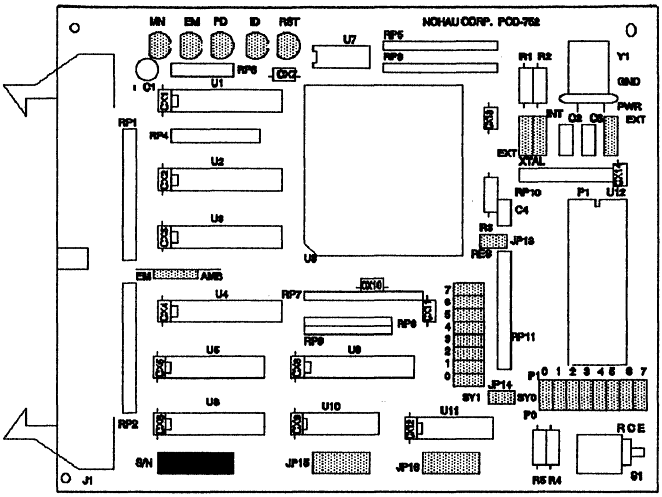


Figure 76. Locations of Jumpers on POD-C752

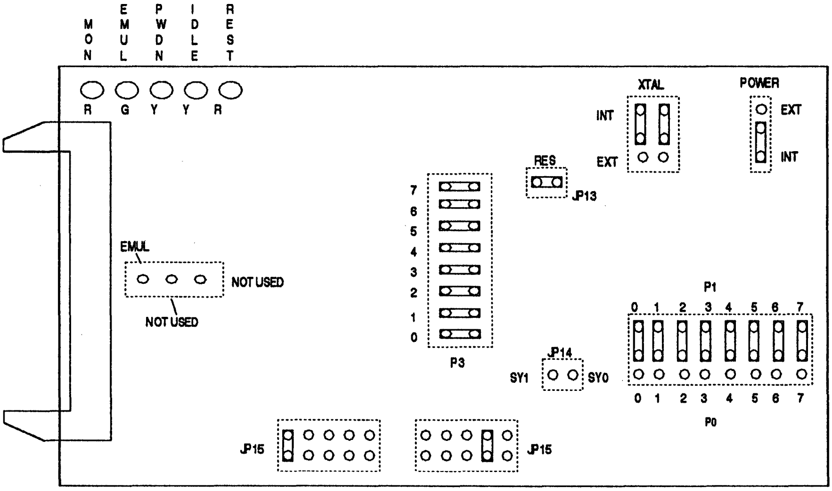


Figure 77. POD-C752 Jumper Diagram

POD-CL782 *POD TYPE: BONDOUT*

The software must be 5.8D or later. Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

```
EMUL51 -p83782 -e110 -m128 -t100 -f16        ;typical invocation
```

The pod board is used to emulate Philips/Signetics' P83CL781 and P83CL782.

The EPROM on the emulator board must be version COM 1.2 or later. If it is older replace it with the EPROM enclosed with the pod.

The bottom side of the board has a 40 pin DIP plug. The pod board goes into the target system, replacing the microcontroller IC. An adapter is available for 44 pin QFP.

The pod board has a Philips P85C001HFA bondout chip and a P83CL782/800 derivative chip to emulate the microcontroller chip.

The pod board has five LEDs marked MON, EMUL, PWRDN, IDLE, and RES:

MON is red, and if lit means that the system is in monitor mode. In monitor mode, the bondout chip is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and if lit means that the system is in emulation mode. In emulation mode the bondout chip is executing user code. This code is normally in a special RAM on the emulator board.

PWRDN is orange, and if lit means that the system is in power down mode. An interrupt or reset can take it out of this mode.

IDLE is orange, and if lit means that the system is in idle mode. An interrupt or reset can take it out of this mode.

POD-CL782

If you break in Idle or Power Down mode, you must issue "reset chip" to regain communication with the processor.

RES is red, and if lit means that the bondout chip RESET input is high (ON).

The jumper marked POWER close to the pushbutton is used for power to the bondout chip. If the power is to be supplied from the emulator (5 volts), the jumper should be across the two left pins (INT). If power is to come from the target system (1.5 - 5 volts), the jumper should be across the two right pins (EXT).

The jumpers marked XTAL close to the crystal determine if the crystal or clock is from the target system, or if the crystal on the pod is used. The jumper should be in the upper two positions for the crystal on the pod board (INT), and the lower two positions for target supplied crystal or clock (EXT).

The jumpers above must be in internal (INT) positions when no target system is connected. This is also the default position.

The jumper marked RES between the two big chips on the board connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with monitor mode and the jumper should be removed.

The jumpers marked SY0 and SY1 close to the ribbon cable connector are used to connect external signals to the emulator. They are used for the trace function. The left pin, SY0, can also be used in the breakpoint logic. The name of the right pin is SY1.

Jumper pins EM/, FLF/, and ANB/ carry signals from the emulator. The left pin EM/ is high in monitor mode, and low in emulation mode. The middle pin FLF/ and the right pin ANB/ and are used with the Advanced Trace Board (ATR). See the documentation for the ATR.

For jumpers P2/P3: the top row of pins is connected to port P3, in order, with P3.0 at the left and P3.7 at the right. The bottom row of pins is connected to port P2, with P2.0 at the left and P2.7 at the right. The middle row of pins is reflected on the trace display of P3. The board is delivered with jumpers so that P3 is traced. If the jumpers are moved to the bottom row, P2 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

For jumpers P0/P1: the top row of pins is connected to port P1, in order, with P1.0 at the left and P1.7 at the right. The bottom row of pins is connected to port P0, with P0.0 at the left and P0.7 at the right. The middle row of pins is reflected on the trace display of P1. The board is delivered with jumpers so that P1 is traced. If the jumpers are moved to the bottom row, P0 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

Jumper INTA/C1 has the bondout chip signals INTA (interrupt acknowledge) and C1 (first op code fetch).

If you want to trace C1 or INTA the corresponding pins must be connected to the "middle row" of P2/P3 jumper rows thereby replacing those ports in the trace display.

JB1 close to the crystal is used to set the amount of internal memory emulated. It is also used to emulate the "ROM PROTECT" feature. See Figure 79.

S1 pushbutton is used to reset the bondout chip, and can be used instead of the target system reset.

Connecting the pod board to the target system:

Make sure that the power is turned off in both the host PC or box, and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Plug the board carefully into the target system, with the pod board oriented so the long ribbon cable goes out to the left. Make sure that the target system pin 1 is towards the bottom of the board.

POD-CL782

Turn on power to the PC or box and then to the target system.

When turning power off, turn the target off first, then the PC.

Note:
If the target system is running at a voltage lower than 5 volts, the power jumper should be in the EXT position. In this situation the bondout chip will be powered from the target system. Since the voltage level translation circuitry uses some power from the bondout chip signal lines, the current consumption will be higher than for a "real" chip. The current may be up to 10 mA.

Since the P83CL782 processor is mainly intended for ROM code with P0 and P2 used as port pins, and because of the amount of circuitry involved for voltage level translation, the emulation support for external memory access is limited.

MOVX instructions are not fully supported. XDATA can not be mapped to the emulator. Do not use breakpoints on MOVX read or write cycles. The address and data for MOVX instructions is not correct in the trace display.

CODE can be executed in the target system, but breakpoints will not work properly. The trace will show correct addresses, but the actual code bytes read from the target system will be incorrect. So CODE should normally be mapped to the emulator memory.

Bondout chip bug:

Early versions of the bondout chip may have the following problems:

Writing a one to a port 0 pin that was zero will generate a positive pulse, two clock cycles wide before the pin goes into high impedance state.

POD-CL782

Board Layout Figure 78 shows locations of jumpers and Figure 79 shows the jumper pins in detail.

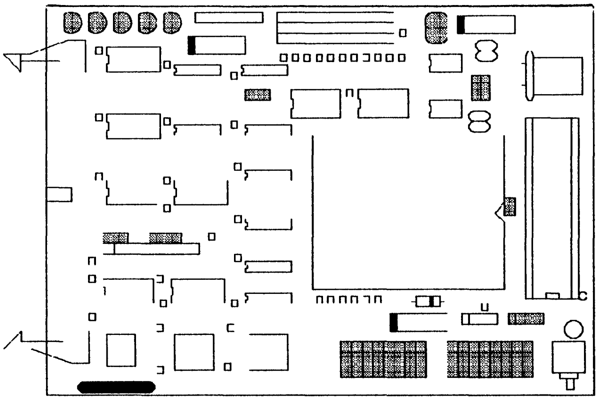


Figure 78. Locations of Jumpers on POD-CL782

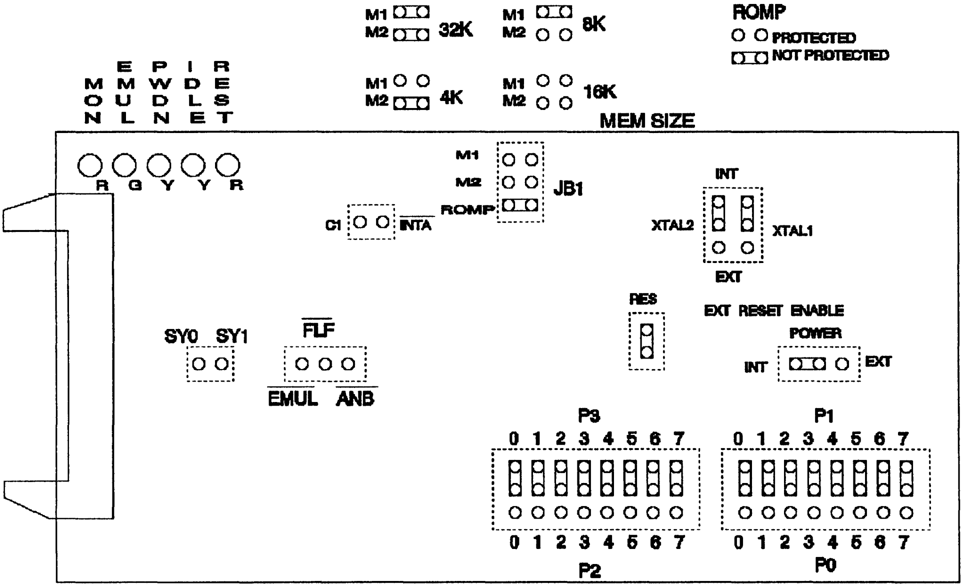


Figure 79. POD-CL782 Jumper Diagram

POD-5001-16 *POD TYPE: BONDOUT*

The software must be 5.8C or later. Invoke the EMUL51-PC as in this example (please see Installation Chapter for option details):

```
EMUL51 -p5001 -e110 -m128 -t100 -f16  ; typical invocation
```

The POD-5001 adapter is used to emulate Dallas Semiconductor DS5000, DS5000T, DS5000FP, DS5001FP, and DS5002FP.

The EPROM on the emulator board must be version COM 1.43. The EPROM should be enclosed with the pod if it is not already on the emulator board. This EPROM is a special version for the POD-5001. This EPROM will not work well with regular external mode pods.

The solder side of the board has two 16 pin headers and two 24 pin headers. They are designed to plug into this Emulation Technology adapter for surface mount pads (FP package): EPP-080-QF08-LG. The pod will also plug into the included Nohau adapter ADAP5001-DIP40 to emulate the 40 pin DIP.

The Emulation Technology adapter can be ordered from Nohau or directly from Emulation Technology.

The adapter goes into the target system replacing the microcontroller IC.

The pod board has a special Dallas Semiconductor bondout chip to emulate the microcontrollers.

The pod board has three LEDs, named MON, EMUL, and RESET:

MON is red, and means that the system is in monitor mode. In monitor mode, the processor is executing code that is internal to the emulator. This code is not user code, and is used to communicate with the host PC, to set up breakpoints etc.

EMUL is green, and means that the system is in emulation mode. In emulation mode the processor is executing user code. This code is in a

POD-5001-16

special RAM on the emulator board, or on the target PROM depending on how code is mapped (MAPC).

RESET is red, and means that the processor RESET input is high (ON).

Jumpers XTAL determine if the crystal or clock is taken from the target system, or if the crystal on the pod is used. The jumpers should be in the lower two positions for the crystal on the pod board (INT), and the upper two positions for target supplied crystal or clock (EXT).

Jumper POWER is used for power to the processor. If the power is to be supplied from the emulator, the jumper should be across the two left pins (INT). If power is to come from the target system, the jumper should be across the two right pins (EXT).

All jumpers for POWER and XTAL must be in internal (INT) positions when no target system is connected. This is also the default position.

Jumper RST connects the target RESET pin to the emulator. If the target system has a watchdog, it will probably interfere with the emulator in monitor mode and RST should be removed.

Jumper pins SY0 and SY1 are used to connect external signals to the emulator. They are used for the trace function. The lower pin, SY0, can also be used in the breakpoint logic.

Jumper PROC determine which processor to emulate. In the left position 5001 and 5002 will be emulated, and in the right position 5000 will be emulated. All power, including the computer (or box), must be off when the jumper is moved.

Jumper CLOCK must be in the right (EN) position to enable the DS1215 chip to emulate DS5000T. In all other cases it should be in the left (DIS) position.

Jumper RD/WR must be in the left (DIS) position if P3.6 and P3.7 are used as I/O pins. If P3.6 and P3.7 are used as /RD and /WR signals the jumper must be in the right (EN) position.

POD-5001-16

Jumper MEM must be in the left (TARG) position if the code is mapped to the target. It should be in the right (EMUL) position if the code is mapped to the emulator.

Jumper SDI should be removed for emulation of DS5002FP.

For jumpers P2/P3: the top row of pins is connected to port P3, in order, with P3.7 at the right and P3.0 at the left. The bottom row of pins is connected to port P2, with P2.7 at the right and P2.0 at the left. The middle row of pins is reflected on the trace display of P3. The board is delivered with jumpers so that P3 is traced. If the jumpers are moved to the bottom row, P2 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

For jumpers P0/P1: the top row of pins is connected to port P1, in order, with P1.7 at the right and P1.0 at the left. The bottom row of pins is connected to port P0, with P0.7 at the right and P0.0 at the left. The middle row of pins is reflected on the trace display of P1. The board is delivered with jumpers so that P1 is traced. If the jumpers are moved to the bottom row, P0 will be traced. External signals or signals from other ports can be connected to the middle row of pins if the jumpers are removed. The input load is one ALS input (74ALS258).

The jumper pins between U4 and U5 carry signals to and from the emulator and trace boards. The left pin EM/ is high in monitor mode, and low in emulation mode. The middle pin FLF/ and the right pin ANB/ and are used with the Advanced Trace Board (ATR). See the documentation for the ATR.

S1 push-button is used to reset the processor, and can be used instead of target system reset.

Note:
The signals /CE1 (/ECE1), /CE2, /CE3. /CE4, /PE1 (/ECE2), /PE2, /PE3, and /PE4 go through a 74ACT32 to the target system. This will add an extra delay up to 9 ns.

POD-5001-16

SY0, and SY1 have each a 100 kohm pullup resistor.

The timer/counters are normally stopped at breakpoints. This usually means that the serial port also stops at breakpoints. If a character is received or sent at this moment it will be distorted.

To enable timers to run in breakpoints, issue this command:

```
NB E512 = FF
```

The emulator will NOT set up MCON automatically when the program is loaded. The user must set MCON to an appropriate value before a program is loaded.

MCON register will be initialized to F8, and RPCTL register will be initialized to 01 when the emulator is started and at the command RESET EMULATOR.

The least significant bit in MCON is the security lock bit SL.

By setting the PAA bit in 5000 mode, or the AE bit in 5001/5002 mode, code can be loaded to all the target program memory including the lower 2k or 4k.

Example of program load into target program memory:

```
Make sure .MCON is set up correctly
MAPC = EMUL                             ; map code to emulator
LOAD user.prg                           ; load your program
RB .RPCTL = 11 or RB .MCON = FA         ; set AE or PAA bit
MAPX = USER                             ; map data to target
XB 0 = CBY 0 TO FFF                     ; move 1000 bytes (rather slow)
MAPC = USER                             ; map code to target
RB .RPCTL = 1 OR RB .MCON = F8          ; clear AE or PAA
```

If P2 and P0 are used as I/O ports, make sure the XDATA window is disabled (Command: DISABLE XDATA).

POD-5001-16

When single stepping through a program with many interrupts (for example TEST.A03), the background program should execute one instruction between every interrupt. Because of the design of this particular bondout chip, we can not emulate this, and no instructions from the background program will be executed if an interrupt is pending (when single stepping).

Memory accessed with PSEN can not be accessed in monitor mode and can therefore not be displayed or disassembled. This memory can however be executed and traced in emulation mode.

Connecting the pod board to the target system:

For 40 pin DIP connect the included adapter to the pod. Note the position of pin 1.

For 80 pin PQFP, solder the Emulation Technology adapter to your target system instead of the microprocessor.

Make sure that the power is turned off in both the host PC and the target system.

Connect the black cable with the EZ hook to ground on the target system.

Plug the pod into the target or the adapter. Make sure pin 1 is in the correct position.

Turn on power to the PC first, and then to the target system.

When turning off power, turn off the target first, then the PC.

POD-5001-16

Board Layout Figure 80 shows the locations of jumpers and Figure 81 shows the jumper pins in detail.

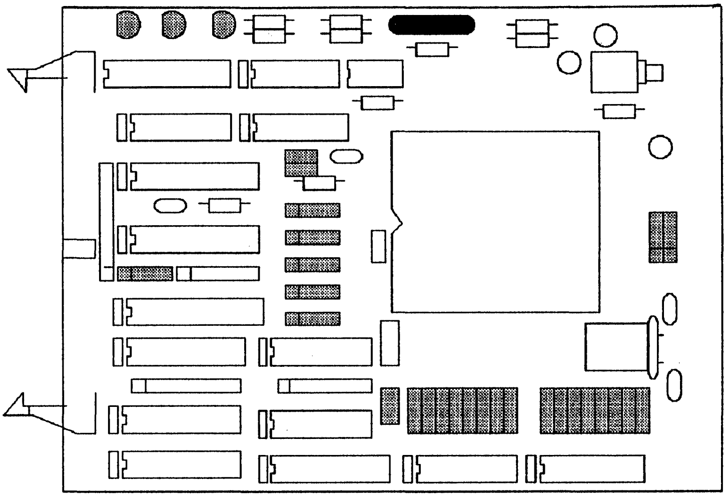


Figure 80. Locations of Jumpers on POD-5001-16

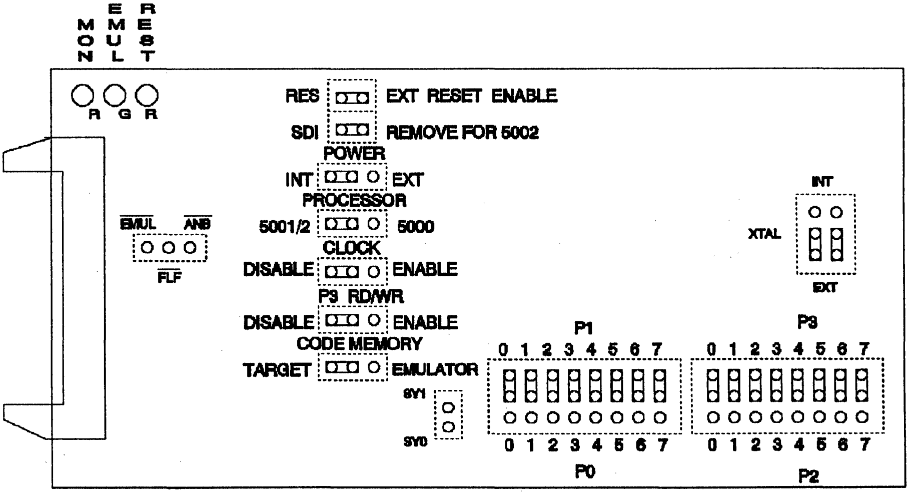


Figure 81. POD-5001-16 Jumper Diagram

5 TROUBLESHOOTING

Basic Standalone Troubleshooting 5-3
Error Messages 5-8
Serial Box 5-8
Problems Operating in Target Systems 5-8
Source Code Display Problems 5-11

5 TROUBLESHOOTING

ALWAYS TURN POWER OFF BEFORE YOU PLUG IN, OR UNPLUG BOARDS, RIBBON CABLES OR POD BOARD!

Troubleshooting Sections:

1. **Standalone Problems**
2. **Error Messages**
3. **Serial Box**
4. **Problems Operating in Target Systems**
5. **Source Code Display Problems**

The first question is whether the emulator and POD together operate by themselves without being connected to the target system. If you can start EMUL51 and do simple commands, you can go on to the paragraphs about problems operating with target systems.

BASIC STANDALONE TROUBLESHOOTING

"Standalone" means that the bottom of the POD is not connected to any target system. Remove anything from the bottom plug of the POD. Make sure that there is not any conductive foam (usually black) on the bottom of the POD. If there is an adapter, remove it to be sure it is not causing a problem.

Power And Crystal

The power or voltage "V" jumper(s) must be set to POD power (internal, "INT" or "I"). The clock or crystal jumpers ("XTAL", "XTL" or "X") must be set for POD crystal (internal, "INT" or "I").

If you can start the emulator, you should not get any error messages. You should be able to execute simple commands such as CBYTE 0 = AA or RESET CHIP with no error messages. If you get errors, continue troubleshooting standalone.

If you have a valid instruction loaded at address 0000, you should be able to execute a STEP command and have the chip execute as expected.

Things to look for if you get error messages:

CABLING

Check that the ribbon cable is connected to the emulator on the back of the PC. Check that the POD is connected to the other end of the ribbon cable.

You can plug either end of the ribbon cable into either the POD or the emulator; there is no "POD end". The ribbon cable is keyed to indicate which direction it plugs into its connector. However, it is not impossible to force it in incorrectly, so check for proper orientation.

CONNECTIONS

Check that the board is securely plugged into the PC's system board connector. The bracket should be screwed into the computer's chassis.

PARAMETERS

Check the startup parameters.

PROCESSOR PARAMETER

a. If the processor parameter, "-p" does not agree with the POD type, you will get errors and faulty operation. You can identify the POD type from the illustrations in the POD chapter, or from invoicing information. For example, the only POD that uses "-p8051" or "-p8052" has a '535 chip on it. There are tables of -p parameters in the Installation Chapter.

MEMORY PARAMETER

b. The emulator memory size parameter, "-m" is important. The memory parameter is either -m32 or -m128. Usually it matches the emulator board memory size. Verify the emulator memory size by comparing the installed memory chips and jumper settings to the settings shown in the Installation Chapter.

If you get a memory failure readback check error, the parameter may be incorrect. Note that the 128 k board can be set in a mode to use the -m32 parameter. Also, the 128k bankswitch modified board uses the -m32 parameter.

FRAME PARAMETER

c. An incorrect trace frame size parameter, " -f " will not normally cause startup problems or error messages. The parameter is important for correct display of the order of the frames.

If you have the trace board installed, you must have the frame parameter and trace address parameter, " -t ". Otherwise, you may get constant breakpointing upon GO commands because of uninitialized trace hardware.

SOFTWARE AND EPROM

Make sure the software revision and PROM are recent enough to run the POD. External mode PODs generally can run with any version software and PROM. Most bondout and hooks PODs specify the required software version.

The "COM" EPROM is on the emulator board. COM 1.0 (labeled "COM") is adequate for most external PODs. Most bondout and hooks PODs specify the required PROM level. If the emulator can start, the VERSION command reports the COM PROM version.

FURTHER STANDALONE TROUBLESHOOTING

BOARD SWAPPING

a. If you have other EMUL51-PC systems, you can swap PODs or emulator boards around to isolate the unit with the problem. If you use a different model POD, be sure to use the correct -p.

VOLTAGE CHECKING

b. You can check for an acceptable 5 volt level on one of the regular logic chips on the POD. If it is low, such as 4.5 volts or lower, the system is not adequately supplying the emulator and POD.

We have noticed that especially on some older PC clones the power distribution is very poor. Even if the voltage is 5 volts at the power supply, it may be down to less than 4.6 volts when it reaches some of the I/O boards. Then there is an additional voltage drop from the emulator to the pod board. Even if the voltage is less than 4.5 volts on the pod board, it usually works, but the whole system becomes much more sensitive to noise.

If the voltage is marginal, one tactic is to move the emulator board to a slot closer to the power supply end of the computer's backplane.

CHIP SWAPPING

c. Turn power off before replacing any chips.

(1) *Bondout PODs.* This option does not apply to bondout PODs. Replacement bondout chips are not available separately.

(2) *External mode PODs.* If you have an external mode DIP POD, you can simply change the microcontroller in the socket. If your POD takes a PLCC chip, it is best to use a chip extractor tool (available from Nohau or other suppliers) to change chips. Ideally the chip should be exactly the same type.

(3) *Hooks mode PODs.* Some hooks mode PODs use readily available standard microcontroller chips and some do not. Consult the specific POD description.

The Xilinx logic array chip on a hooks mode POD can be changed if the speed of the replacement is as fast. The logic in the Xilinx is not pre-programmed; it gets loaded from the small PROM on the POD upon power-up.

(4) *Port Substitution.* The chip is a standard 80535 or 80C535 and can be swapped.

ADDRESSING

d. Change addresses on the emulator board. See the installation chapter for jumper settings. The " -e " parameter must agree with the emulator address. The " -t " parameter is the trace address.

Try different addresses that you don't think are occupied. The emulator uses 8 consecutive addresses. Make sure that the emulator and trace boards don't have the same or overlapping addresses. The trace board uses 16 consecutive addresses. If the emulator does not work, you still may have a conflict on the I/O bus. If you have boards in your PC, and you are not sure which I/O addresses they use, pull these boards out to see if that helps.

ERROR MESSAGES

The error message "WRITE TIMEOUT ERROR" followed by "READ TIMEOUT ERROR" and then "Can't find emulator at address xxx" means that communication between the emulator and the PC does not work. The POD must operate for the software to find the emulator.

The numbers in the error messages allow the EMUL51 program to look up the text of the error message. For example, "165" is the reference that allows the program to look up the text "READ TIME OUT ERROR" and "219" looks up the text "Can't find emulator board on address".

There is no list of emulator error messages. The text of the message describes the error.

SERIAL BOX TROUBLESHOOTING

All the information about standalone troubleshooting applies to systems in the external serial box. The system is linked through all the parts, and if any link fails, there will be error messages.

Refer to the BOX-S section in the installation chapter. That section lists baud rate settings, parameters and box-specific errors.

If other box systems are available, troubleshoot by swapping different items in the sets. If necessary and feasible, take the emulator out of the box and troubleshoot the emulator and POD in PC plug-in configuration.

PROBLEMS OPERATING IN TARGET SYSTEMS

This section deals with problems that do not exist when you operate the emulator and POD standalone. Standalone troubleshooting is covered earlier in this chapter.

Some errors will prevent the emulator from starting.

Emulator Software Does Not Start

If power is jumpered for target system, make sure the power is available from the target.

Check whether RESET is active on the target system. The POD has a jumper to allow isolating the RESET line between your target system and the POD. If there is an external watchdog circuit or if RESET is held active, the POD will be prevented from starting.

Try using internal crystal. If internal crystal works but external crystal or oscillator does not, investigate the circuitry. In some cases it may be necessary to compensate for differences between the POD oscillator circuit and the chip's. Ideally, there should be no difference, but a pragmatic approach may be required.

If you use a target crystal that is much slower than the POD crystal and you get errors, see the "-d" parameter in the Installation Chapter.

EXTERNAL MODE PODS

Is the "PSEN jumper" in the factory-installed position? If it is not, PSEN will go out on the target even in monitor mode. If it is used to decode a PROM address for example, it might conflict with the internal operation of the emulator.

Do you have anything on your target system that may drive the data bus, and which is decoded without using PSEN, RD, or WR? In that case it may interfere with the internal operation of the emulator.

You can use an isolator to find out if there is a conflict between the target system and the POD. Note that such conflicts can exist even though the target system operates okay without the emulator.

The DIP40-ISO is such an isolator. Others are available for other pin counts. Use it to turn off switches to isolate the conflict. The Port 0 data bus, AD0 through AD7 are a good place to start. You could also try turning off the Port 2 address bus, A8 through A15.

Another technique is to remove devices from your target system. Pull out all devices on your target system that can drive the data bus. If it still does not work, you most likely have some other problem on the P0 or P2 lines. You may have a short or maybe excessive capacitance on one or more of the signals in your target system.

HOOKS MODE PODS

If the POD addresses external memory, use the same troubleshooting techniques as for external PODs.

Check that the memory jumpers are configured correctly. See the specific POD documentation.

BONDOUT PODS

If the POD addresses external memory, use the same troubleshooting techniques as for external PODs.

Check that the processor jumpers are configured correctly. See the specific POD documentation.

PROGRAM COUNTER PROBLEMS

If the program counter jumps off to some unusual address, or if a step of a simple instruction does not operate correctly, use the above methods. If you do not have the problem standalone, use the same techniques as for troubleshooting when the emulator software does not start.

EXTERNAL READ-WRITE PROBLEMS

Improve the ground connection between the pod and the target system (especially if you are using an extender cable).

Diagnose the problems by using the XBYTE command to write and read external memory.

You can assemble small assembly language programs to execute external reads and writes. A tight loop can provide a high duty cycle program. That makes it easier to trigger an oscilloscope or analyzer on the read or write signal for troubleshooting.

TRACE PROBLEMS

If the frames of the trace seem non-contiguous, check the jumper settings, the " -f " frame parameter and the memory sizes according to the Installation Chapter.

If data is missing and is shown by dashes (- -), it means the filter term (for example, "TRACE A", or "Record: Yes, if A") is suppressing the information that would be shown where the dashes are.

SOURCE CODE DISPLAY PROBLEMS

Follow the guidelines for Source Level Debugging in the Orientation Chapter. Use the procedure that is specific for your compiler. Pay particular attention to compile and link switches.

Normally, you can display source code if you follow these steps:

Load your program.

Enable the code window.

RESET CHIP, if you have not already.

Type LINE (or L) for LINE step. You may need to LINE step more than once.

Emulation should stop. If it does not, line symbols may be missing.

If emulation stops, press the F2 function key, or type ENABLE SOURCE. LINE step again if necessary.

If emulation does not break, use ESC to stop it manually. To see if you have line symbols loaded, type "SYM ? ? LINE". If there are no symbols of type "line", there is a compile or link problem.

6 TUTORIAL

User Interface 6-3
Emulator Commands 6-15
Standard Trace Commands 6-22
Advanced Trace Commands 6-41
Advanced Trace Features 6-56
Search Function 6-72

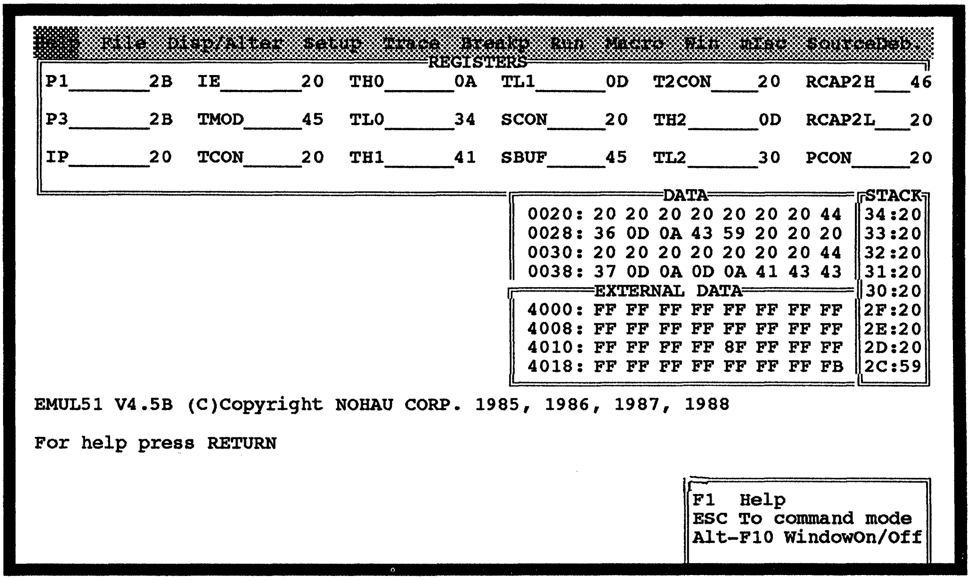
6 TUTORIAL

The short, interactive tutorial modules in this section of the manual provide a quick and convenient way for you to become familiar with the EMUL51. All necessary files are included on the diskettes supplied by Nohau Corp. You can work on the Tutorial anytime after you've installed the EMUL51. To start, merely invoke the emulator in the usual way from the DOS command line, by typing the appropriate path specification and name of the batch file for your hardware configuration.

User Interface

The goal of this module is to introduce you to the EMUL51 user interface. Approximate time for working through this practice is 15 to 20 minutes.

First Screen Figure 1 shows the first screen that appears after you invoke the emulator. Note that the entire first line, called the *Top Menu Line*, is highlighted. To indicate the initial position of the keyboard cursor, the first item on the line, *Help*, is highlighted with more intensity than the other items. (If you have a mouse connected, a second mouse cursor also appears on the screen. That cursor will be discussed later in this module.)



```
[illegible] File Disp/Alter Setup Trace Breakp Run Macro Win mIsc SourceDeb.
                                   REGISTERS
P1________2B  IE________20  TH0_______0A  TL1_______0D  T2CON____20  RCAP2H___46

P3________2B  TMOD______45  TL0_______34  SCON______20  TH2_______0D  RCAP2L___20

IP________20  TCON______20  TH1_______41  SBUF______45  TL2_______30  PCON_____20

                                   DATA                        STACK
                         0020: 20 20 20 20 20 20 20 44        34:20
                         0028: 36 0D 0A 43 59 20 20 20        33:20
                         0030: 20 20 20 20 20 20 20 44        32:20
                         0038: 37 0D 0A 0D 0A 41 43 43        31:20
                               EXTERNAL DATA                  30:20
                         4000: FF FF FF FF FF FF FF FF        2F:20
                         4008: FF FF FF FF FF FF FF FF        2E:20
                         4010: FF FF FF FF 8F FF FF FF        2D:20
                         4018: FF FF FF FF FF FF FF FB        2C:59
EMUL51 V4.5B (C)Copyright NOHAU CORP. 1985, 1986, 1987, 1988

For help press RETURN

                                             F1  Help
                                             ESC To command mode
                                             Alt-F10 WindowOn/Off
```

Top Menu Line }

Registers Window }

Scroll Window }

Help Line }

{Data, Stack, External Data Windows

{ Help Window

Figure 1. First Screen After Invoking Emulator

The next lines on the screen are enclosed in a rectangular window labeled *REGISTERS*. Listed here are various registers in the POD processor and their current contents (in hexadecimal). Below this box are three additonal windows (labeled *DATA, STACK* and *EXTERNAL DATA*) which show contiguous memory addresses and the current contents of these addresses.

The line near the middle of the screen shows the EMUL-51 software version number and copyright date lies in the *Scroll Window*. All other windows on the screen are displayed on top of the Scroll Window. The window near the lower right corner of the screen is called the *Help Window*.

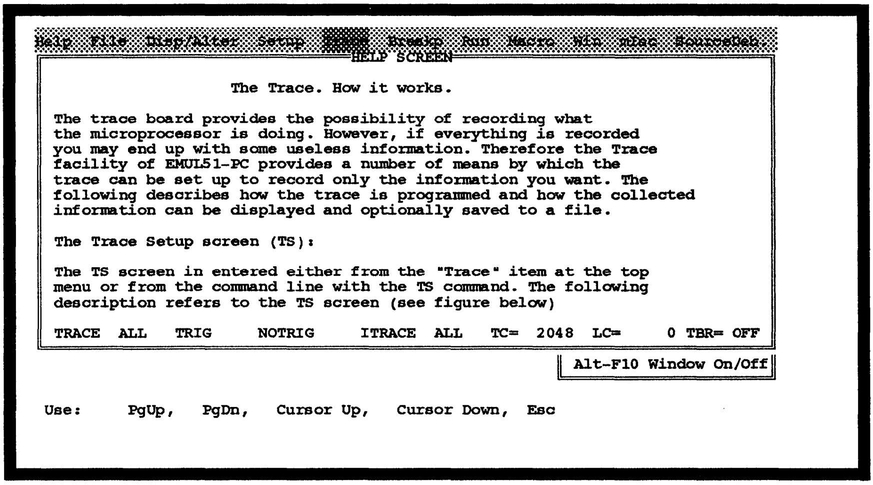
The last line on the screen is called the *Help Line*. It shows a message about the current choice on the Top Menu Line.

Moving the Cursor To move the cursor along the Top Menu Line, press the Right or Left Arrow cursor key. As you do this, note that the message on the Help Line at the bottom of the screen changes when a different item is selected. Also note that when you are at the *Help* item, pressing Left Arrow will "wrap" the cursor around to the *SourceDeb* item at the end of the line.

You can also move from one item to another by pressing the ALT key and the upper case character in the desired item. For example, pressing ALT + T moves the cursor to the *Trace* item, pressing ALT + I moves cursur to *misc* item, and so forth.

Exiting to DOS After moving the cursor to the *mIsc* item, you can return to DOS by pressing the ENTER key three times. (If you do this, you must invoke the emulator again to continue with the Tutorial.)

Help Paragraphs Some items on the Top Menu Line have general nature "help" paragraphs associated with them. To observe this feature, move the cursor to the *Trace* item and then press the F1 Function Key. Doing this brings up the screen shown in Figure 2.


Help File Disp/Alter Setup Breakp Run Macro Win misc SourceDeb.

```
                     HELP SCREEN

                The Trace. How it works.

The trace board provides the possibility of recording what
the microprocessor is doing. However, if everything is recorded
you may end up with some useless information. Therefore the Trace
facility of EMUL51-PC provides a number of means by which the
trace can be set up to record only the information you want. The
following describes how the trace is programmed and how the collected
information can be displayed and optionally saved to a file.

The Trace Setup screen (TS):

The TS screen in entered either from the "Trace" item at the top
menu or from the command line with the TS command. The following
description refers to the TS screen (see figure below)

TRACE  ALL   TRIG    NOTRIG    ITRACE  ALL   TC=  2048  LC=     0 TBR= OFF
                                                  Alt-F10 Window On/Off

Use:     PgUp,   PgDn,   Cursor Up,   Cursor Down,  Esc
```

Figure 2. Help Screen for Trace

Text inside the box describes the Trace feature. The line at the bottom of the screen identifies cursor keys that can be used to move new text into the box:

- PgDn to scroll the next box of text onto the screen
- PgUp to scroll the previous box of text onto the screen
- Down Arrow to scroll text up one line
- Up Arrow to scroll text down one line.

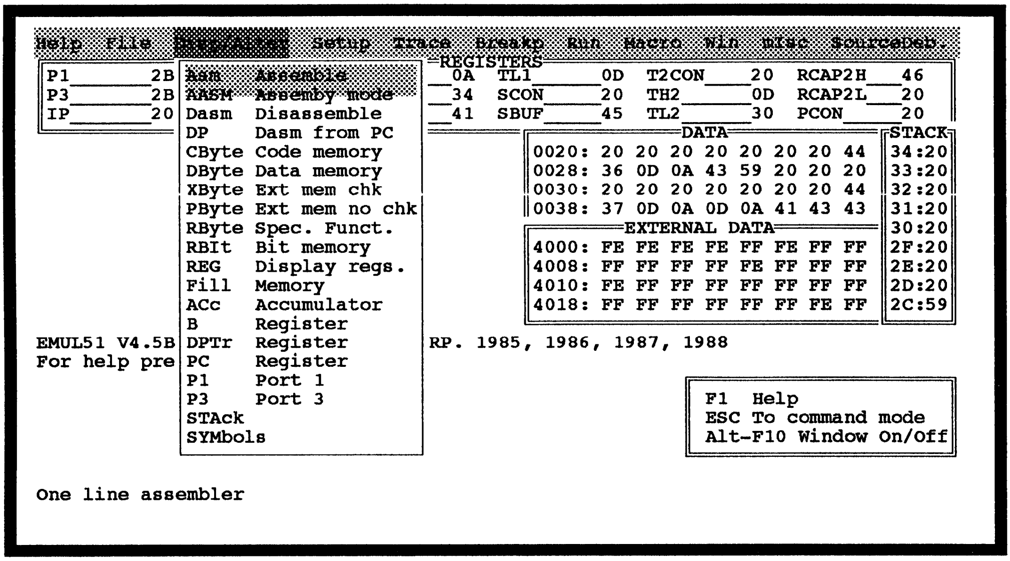
Esc at the end of the line at bottom of the screen tells you that pressing the ESC key will take you back to the original menu.

Use these cursor keys to move around in the text, then press ESC to continue with the tutorial.

Pull-Down Menus

The items on the Top Menu line identify major groups or categories of EMUL-51 commands. Individual commands within a given category can be accessed through pull-down menus from this line.

To see how this feature works, move the cursor to the *Display/Alter* item, then press the ENTER key. Figure 3 shows the pull-down menu that appears after you do this.



```
Help  File  Disp/Alter  Setup  Trace  Breakp  Run  Macro  Win  misc  SourceDeb.
P1_______2B Asm   Assemble      =REGISTERS=
P3_______2B AASM  Assemby mode  __0A  TL1______0D  T2CON____20  RCAP2H___46
IP_______20 Dasm  Disassemble   __34  SCON_____20  TH2______0D  RCAP2L___20
            DP    Dasm from PC  __41  SBUF_____45  TL2______30  PCON_____20
            CByte Code memory          =========DATA==========  STACK
            DByte Data memory          0020: 20 20 20 20 20 20 20 44  34:20
            XByte Ext mem chk          0028: 36 0D 0A 43 59 20 20 20  33:20
            PByte Ext mem no chk       0030: 20 20 20 20 20 20 20 44  32:20
            RByte Spec. Funct.         0038: 37 0D 0A 0D 0A 41 43 43  31:20
            RBIt  Bit memory           =====EXTERNAL DATA=====        30:20
            REG   Display regs.        4000: FE FE FE FE FF FE FF FF  2F:20
            Fill  Memory               4008: FF FF FF FF FE FF FF FF  2E:20
            ACc   Accumulator          4010: FE FF FF FF FF FF FF FF  2D:20
            B     Register             4018: FF FF FF FF FF FF FE FF  2C:59
EMUL51 V4.5B DPTr Register       RP. 1985, 1986, 1987, 1988
For help pre PC   Register
            P1    Port 1                         F1  Help
            P3    Port 3                         ESC To command mode
            STAck                                Alt-F10 Window On/Off
            SYMbols

One line assembler
```

Figure 3. Pull-Down Menu for Display/Alter Commands

Each line on the pull-down menu gives the name of a command. The line that is higlighted indicates the current position of the cursor. Use the cursor Up Arrow and Down Arrow keys to move between commands. As you do this, note that the Help line (at bottom of screen) describes use of the currently highlighted command.

By pressing the F1 Function Key, you can bring up more detailed information into the Help Window.

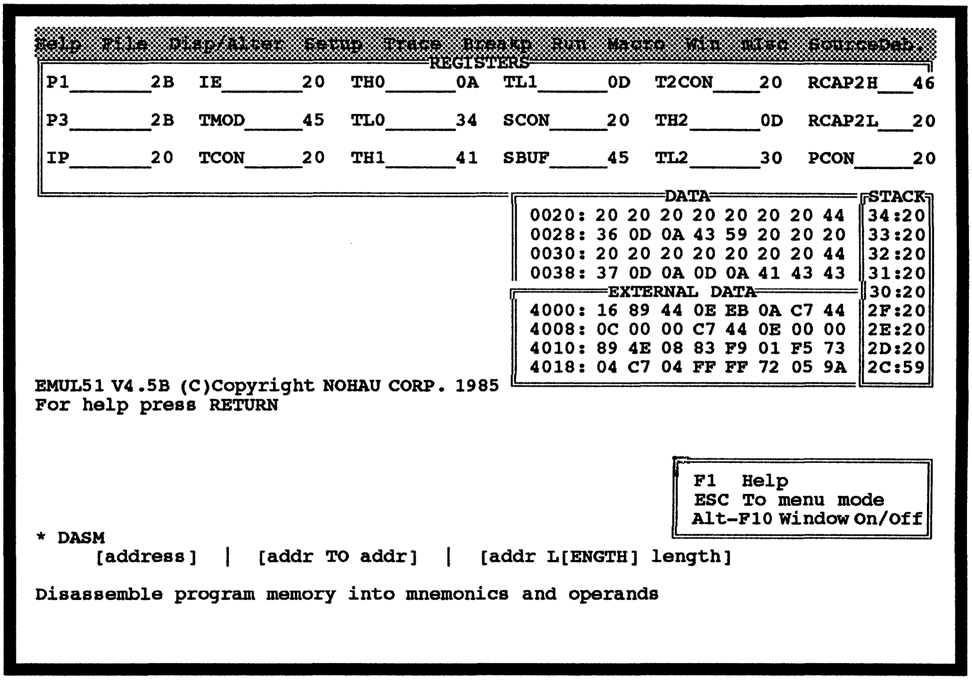
On the pull-down menu note that abbreviations for the commands begin with one or more capital letters. The capital letters denote the minimum form of the command.

For faster cursor movement on a pull-down menu, press the ALT key plus a number key or a letter key *A, B, C,* etc. For example, to move the cursor from the top line (line #1) to the ninth line, press ALT + 9. To move to the tenth line, press ALT + A, and so forth.

If you want to remove the pull-down without making a selection, press the ESC key.

Command Line

To continue with the Tutorial, on the pull-down menu for *Disp/Alter* move the cursor to *Dasm*, (which is the second command down from the top). The Help line now tells you that *Dasm* is a command that disassembles the contents of user program memory into assembler mnemonics and operands. Next press ENTER. Doing this changes the screen as shown in Figure 4.


Help File Disp/Alter Setup Trace Breakp Run Macro Win Misc SourceDeb.

```
                                   REGISTERS
P1_______2B  IE_______20  TH0______0A  TL1______0D  T2CON____20  RCAP2H___46

P3_______2B  TMOD_____45  TL0______34  SCON_____20  TH2______0D  RCAP2L___20

IP_______20  TCON_____20  TH1______41  SBUF_____45  TL2______30  PCON_____20

                                                  DATA                  STACK
                                 0020: 20 20 20 20 20 20 20 44   34:20
                                 0028: 36 0D 0A 43 59 20 20 20   33:20
                                 0030: 20 20 20 20 20 20 20 44   32:20
                                 0038: 37 0D 0A 0D 0A 41 43 43   31:20
                                         EXTERNAL DATA            30:20
                                 4000: 16 89 44 0E EB 0A C7 44   2F:20
                                 4008: 0C 00 00 C7 44 0E 00 00   2E:20
                                 4010: 89 4E 08 83 F9 01 F5 73   2D:20
                                 4018: 04 C7 04 FF FF 72 05 9A   2C:59
EMUL51 V4.5B (C)Copyright NOHAU CORP. 1985
For help press RETURN

                                              F1  Help
                                              ESC To menu mode
                                              Alt-F10 Window On/Off
* DASM
    [address]  |  [addr TO addr]  |  [addr L[ENGTH] length]

Disassemble program memory into mnemonics and operands
```

Com-mand Line }

Help Line }

Figure 4. Screen Showing Command Line for DASM Command

The pull-down menu has disappeared, * DASM is written on a Command Line near the bottom of the screen, and the keyboard cursor becomes a blinking underline that is positioned after DASM. On the next line after that you are prompted with a list of parameters defined for this command.

As a further reminder that the keyboard cursor has moved, on the Top Menu line you see that no item is highlighted more brightly than any other.

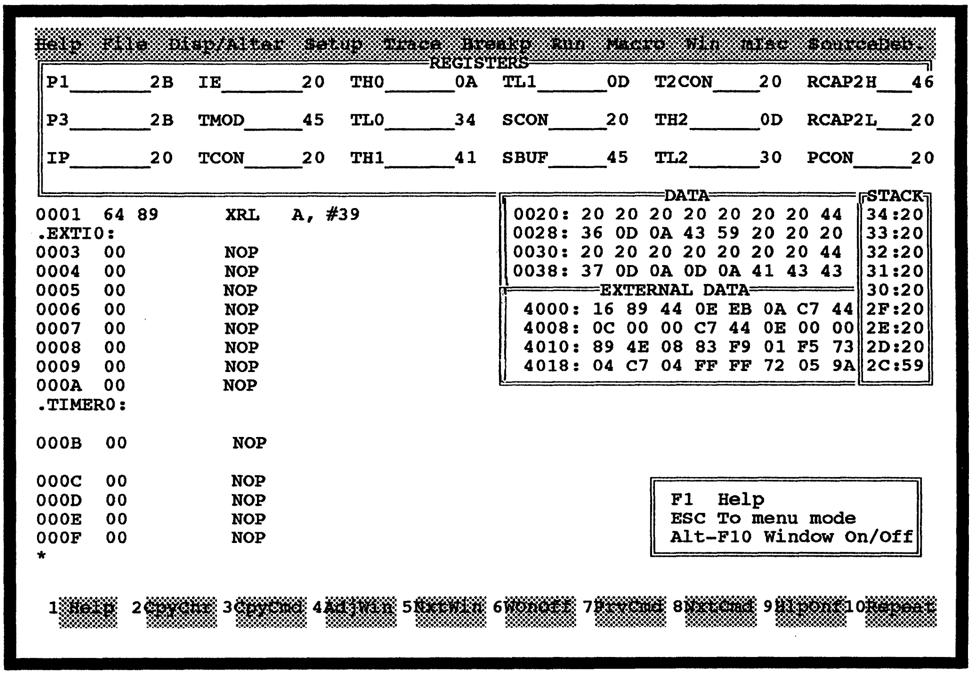
Note the contents of the Help Window at the lower right corner of the screen. Then press the ESC key to confirm that doing this causes the cursor to move back up to the Top Menu line, and that pressing it again moves the cursor directly to the Command Line. This toggling action is handy when you want to move down to the Command Line without using a pull-down menu. The Help Window also serves to remind you that pressing the F1 Function Key will bring context-sensitive help information onto the screen. Finally, confirm that pressing the ALT plus F10 key combination toggles the Help Window on and off.

Command Execution After practicing the keystrokes for cursor movement between the Top Menu line and Command line, and within the pull-down menus, restore DASM to the Command Line as shown in Figure 4. For DASM the parameters may be omitted, so to execute the command in this case you only need to press ENTER. Doing this causes a disassembly of 16 to 18 bytes to appear in a "scroll window" on the screen. After the command has been completed, the cursor remains at the Command line and the emulator is ready for you to type in a new command, (see Figure 5).

Editing Keys Editing of characters on the Command line can be accomplished as follows. Press Left Arrow or Right Arrow to move the cursor. To replace a character, merely type the new character over the old one. To delete a character, press the DEL key. Pressing the INS key toggles you between "overtype" mode and "insert" mode. The shape of the cursor changes to a lighted rectangle when in "overtype," and back to a blinking underline when in "insert."

Function Keys At the bottom of the screen note the contents of the Help line. The numbers 1 through 10 refer to the Function Keys. Press F1, look at the help window that appears after you do this, then press ESC to restore the screen shown in Figure 5.

On the Command line type *REGISTERS* and press ENTER. Doing this causes contents of the most common registers to be displayed, then the cursor returns to the Command line. Now press F2. You will see the characters of the previous command, (REGISTERS), copied one by one onto the new Command line.


Help File Disp/Alter Setup Trace Breakp Run Macro Win misc SourceDeb.
1Help 2CpyChr 3CpyCmd 4AdjWin 5NxtWin 6WonOff 7PrvCmd 8NxtCmd 9HlpOnf10Repeat

Scroll Window }

Command Line }

Help Line }

```
                                  REGISTERS
P1_______2B  IE_______20  TH0______0A  TL1______0D  T2CON____20  RCAP2H___46

P3_______2B  TMOD_____45  TL0______34  SCON_____20  TH2______0D  RCAP2L___20

IP_______20  TCON_____20  TH1______41  SBUF_____45  TL2______30  PCON_____20

                                                         DATA               STACK
0001  64 89       XRL   A, #39          0020: 20 20 20 20 20 20 20 44   34:20
.EXTIO:                                 0028: 36 0D 0A 43 59 20 20 20   33:20
0003  00          NOP                   0030: 20 20 20 20 20 20 20 44   32:20
0004  00          NOP                   0038: 37 0D 0A 0D 0A 41 43 43   31:20
0005  00          NOP                          EXTERNAL DATA            30:20
0006  00          NOP                    4000: 16 89 44 0E EB 0A C7 44  2F:20
0007  00          NOP                    4008: 0C 00 00 C7 44 0E 00 00  2E:20
0008  00          NOP                    4010: 89 4E 08 83 F9 01 F5 73  2D:20
0009  00          NOP                    4018: 04 C7 04 FF FF 72 05 9A  2C:59
000A  00          NOP
.TIMER0:

000B  00           NOP

000C  00           NOP
000D  00           NOP                           F1  Help
000E  00           NOP                           ESC To menu mode
000F  00           NOP                           Alt-F10 Window On/Off
*
```

Figure 5. Screen After Execution of DASM Command

Use the editing keys to change characters on the Command line. An advisory message appears on the Help line if your new command is not valid. Then press ENTER. The command will be executed if it was valid, and the cursor will reappear after the * prompt character on a new line. If the command you typed was not valid, a SYNTAX error will result.

Press F3. Doing this causes the entire previous command to be copied onto the Command line and makes it available for editing.

The Command line entries are saved in a buffer that can hold up to 30 entries. You press F7 to move backward in the buffer, and press F8 to move forward. You may edit any of these commands using the cursor arrows, INS and DEL keys.

To familiarize yourself with the operation of this buffer, type in 10 random commands and experiment. (For this practice, it does not matter whether the commands are syntactically correct or not.)

F9 serves as a toggle for turning the Help line off and on. (After you become very familiar with the emulator commands, you may not want help information showing on the screen all the time.)

Pressing F10 copies the previous command onto the Command line again, (like F3), and also causes the command to be executed again. To see how this works, type *STEP* on the Command line, press ENTER, and then press F10.

Uses of the remaining Function Keys, (F4, F5 and F6), are described below.

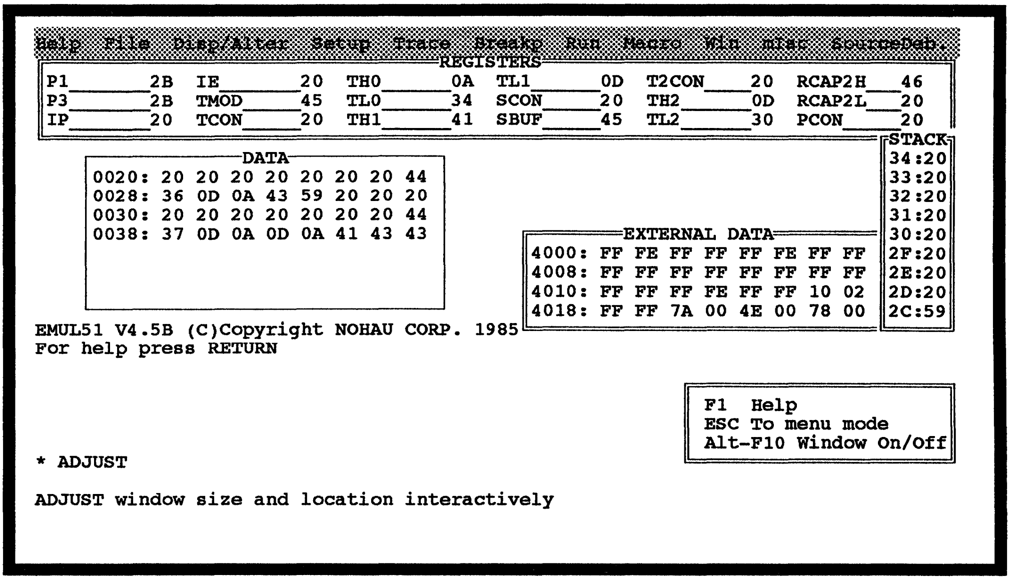
Manipulating the Windows

From the keyboard you can turn the Registers, Data, Stack and External Data windows on and off, move them to other locations on the screen, and change their size.

With the cursor on the Top Menu line, proceed as follows: Move cursor to the *Win* item, press ENTER (to activate the pull-down menu), move cursor to the *ADjust Windows* selection, then press ENTER again. (To invoke this command from the Command line, type *ADJUST* or just *AD*, then press ENTER.)

Press F6 twice. Note that this Function Key toggles all the windows off and on. Press F5. Note that the border around one of the windows changes from a double line to a single line, but that borders around the other windows are unaffected. Press F5 again and note that the border changed back to a double line and that the border of a different window was changed to a single line. The use of F5 is to select the "active" window for subsequent moving and sizing. Press F5 to make Data the "active" Window.

Press F4 and note new contents of Help line: "Use Arrows to Move Window -- Press End to Change Size -- Return to Quit." Press Left and Down Arrow keys to move the Data window above Copyright line, then press END key. After that, press Down Arrow to make the window larger, and press ENTER (see Figure 6).



```
Help  File  Disp/Alter  Setup  Trace  Breakp  Run  Macro  Win  misc  SourceDeb.
                                   REGISTERS
P1_______2B  IE_______20  TH0______0A  TL1______0D  T2CON___20  RCAP2H___46
P3_______2B  TMOD_____45  TL0______34  SCON_____20  TH2______0D  RCAP2L___20
IP_______20  TCON_____20  TH1______41  SBUF_____45  TL2______30  PCON_____20
                                                                        STACK
                    DATA                                                34:20
0020: 20 20 20 20 20 20 20 44                                           33:20
0028: 36 0D 0A 43 59 20 20 20                                           32:20
0030: 20 20 20 20 20 20 20 44                                           31:20
0038: 37 0D 0A 0D 0A 41 43 43           EXTERNAL DATA                   30:20
                                 4000: FF FE FF FF FF FE FF FF          2F:20
                                 4008: FF FF FF FF FF FF FF FF          2E:20
                                 4010: FF FF FF FE FF FF 10 02          2D:20
                                 4018: FF FF 7A 00 4E 00 78 00          2C:59
EMUL51 V4.5B (C)Copyright NOHAU CORP. 1985
For help press RETURN

                                                    F1  Help
                                                    ESC To menu mode
                                                    Alt-F10 Window On/Off
* ADJUST

ADJUST window size and location interactively
```

Figure 6. Screen Showing New Position of Data Window

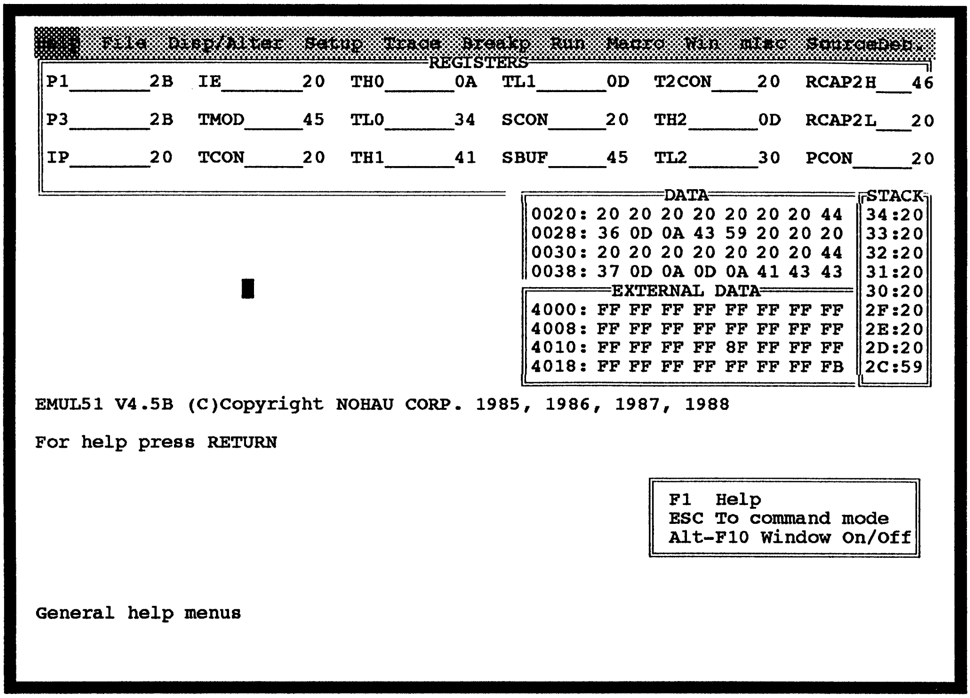
Changing Contents of Windows Whenever the cursor is on the Command line, with the cursor arrow keys you can move the keyboard cursor into any window (other than the Help Window). After that, to change a value within the window, position the cursor over the value and then type in the desired new value. In the Data Window and External Data Window you can cause the contents to scroll by pressing the PgUp and PgDn keys.

For more about how to edit contents of the Registers Window, and about how to save your window setups, move the cursor to the *Win* item on the Top Menu line and then press F1 to bring up "help" information.

Operation With a Mouse If your PC is equipped with a mouse pointing device, on the first screen you will see both the keyboard cursor (whose initial position is over the *Help* item on the Top Menu line), and a mouse cursor (in the form of a lighted solid rectangle), (see Figure 7).

Key-board Cursor }

Mouse Cursor }


File Disp/Alter Setup Trace Breakp Run Macro Win misc SourceDeb.

```
                              REGISTERS
P1_______2B  IE_______20  TH0_______0A  TL1______0D  T2CON____20  RCAP2H___46

P3_______2B  TMOD_____45  TL0_______34  SCON_____20  TH2______0D  RCAP2L___20

IP_______20  TCON_____20  TH1_______41  SBUF_____45  TL2______30  PCON_____20

                                  DATA                     STACK
                          0020: 20 20 20 20 20 20 20 44   34:20
                          0028: 36 0D 0A 43 59 20 20 20   33:20
                          0030: 20 20 20 20 20 20 20 44   32:20
                          0038: 37 0D 0A 0D 0A 41 43 43   31:20
                               EXTERNAL DATA              30:20
                          4000: FF FF FF FF FF FF FF FF   2F:20
                          4008: FF FF FF FF FF FF FF FF   2E:20
                          4010: FF FF FF FF 8F FF FF FF   2D:20
                          4018: FF FF FF FF FF FF FF FB   2C:59
EMUL51 V4.5B (C)Copyright NOHAU CORP. 1985, 1986, 1987, 1988

For help press RETURN

                                              F1  Help
                                              ESC To command mode
                                              Alt-F10 Window On/Off


General help menus
```

Figure 7. Typical First Screen on PC Equipped with Mouse

Operation with a mouse is faster, because the mouse cursor can be moved more quickly than the keyboard cursor, and because the left and right mouse buttons can be used in place of the ENTER and ESC keys, respectively.

To demonstrate this capability, move the mouse cursor to the Top Menu line. (You can do this directly by moving the mouse. Or, if already on the the Command line, merely press the right mouse button. Or, if not already on the Command Line, press the right button twice.) Then move the mouse to positon the cursor over the *misc* item and press the left button. Doing this causes a pull-down menu to appear. Now move the cursor to the *Version Number* selection, then press the left button. Doing this puts VER and the cursor on the Command line. To execute this command, press the left button. The program version number will be printed on the screen.

Next move the mouse to position the cursor somewhere inside the DATA Window and over some byte value that you decide to change. Then press the left button and type in a new value. (You can continue to type in consecutive new values if you wish.) When you are done, move the cursor out of the window and press the left button.

This completes the User Interface module. To continue the Tutorial, go on to the next page.

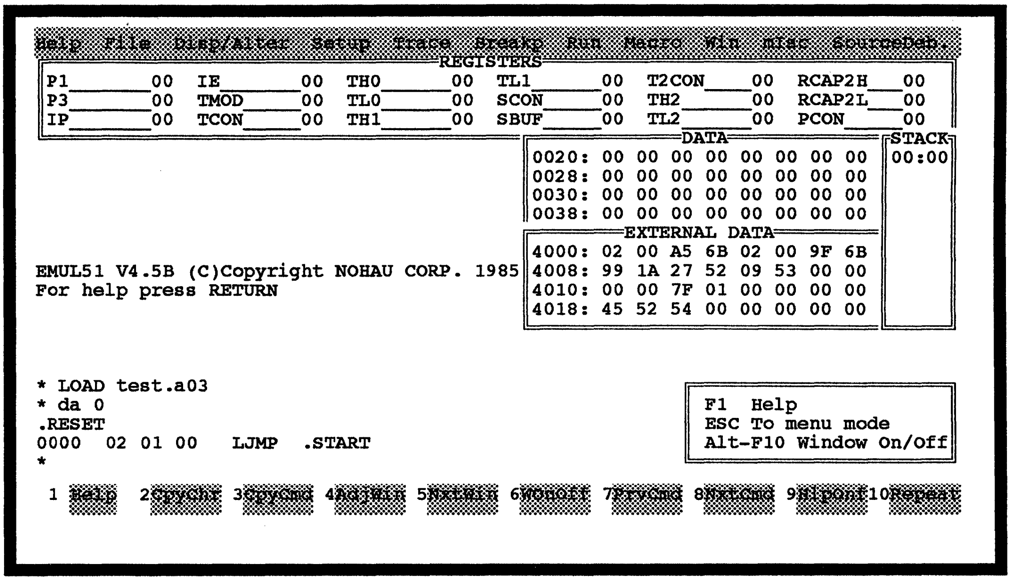
Emulator Commands

The goal of this module is to familiarize you with the basic EMUL51 emulator commands. Approximate time for working through this practice is 10 minutes.

Loading a File

Among the disk files supplied by Nohau Corp. are a sample source file *TEST.SRC* and object file *TEST.A03* including symbols. During installation these files are normally put in the same directory with *EMUL51.EXE*. Starting on the Menu Line, move the cursor to the *File* item and then press ENTER to bring up the pull down menu for that item. Press ENTER again to choose *LOAD*, (which is the command on the top line). Note that doing this moves the cursor down to the Command Line near the bottom of the screen and prints the name of the selected command *(LOAD)* there also.

After *LOAD* type *TEST.A03* and press ENTER. After the emulator carries out this command, the asterisk prompt will reappear on a new line. You can now use the Disassemble command to verify that the test file has been loaded. To do this, on the Command Line type *DASM 0* and press ENTER. (DASM and DA are short forms for Disassemble, and 0 is the reset location for this test program.) A line of disassembled code should appear near the bottom of the screen as shown in Figure 8.



```
Help File Disp/Alter Setup Trace Breakp Run Macro Win misc SourceDeb.
=====================================REGISTERS=====================================
P1_________00  IE_________00  THO________00  TL1________00  T2CON____00  RCAP2H___00
P3_________00  TMOD_______00  TLO________00  SCON_______00  TH2______00  RCAP2L___00
IP_________00  TCON_______00  TH1________00  SBUF_______00  TL2______00  PCON_____00
                                             ==================DATA================ STACK
                                             0020: 00 00 00 00 00 00 00 00         00:00
                                             0028: 00 00 00 00 00 00 00 00
                                             0030: 00 00 00 00 00 00 00 00
                                             0038: 00 00 00 00 00 00 00 00
                                             ============EXTERNAL DATA=============
                                             4000: 02 00 A5 6B 02 00 9F 6B
EMUL51 V4.5B (C)Copyright NOHAU CORP. 1985   4008: 99 1A 27 52 09 53 00 00
For help press RETURN                        4010: 00 00 7F 01 00 00 00 00
                                             4018: 45 52 54 00 00 00 00 00


* LOAD test.a03                                                  F1  Help
* da 0                                                           ESC To menu mode
.RESET                                                           Alt-F10 Window On/Off
0000  02 01 00   LJMP  .START
*
 1 Help  2CpyChr 3CpyCmd 4AdjWin 5NxtWin 6WOnOff 7PrvCmd 8NxtCmd 9HlpOnf10Repeat
```

Figure 8. Screen After Disassembly of One Line of Code

The label *.RESET* is "built into" the emulator, as are all pre-defined symbols. The symbol *.START*, (which was loaded from the *TEST.A03* file), is equivalent to 100H.

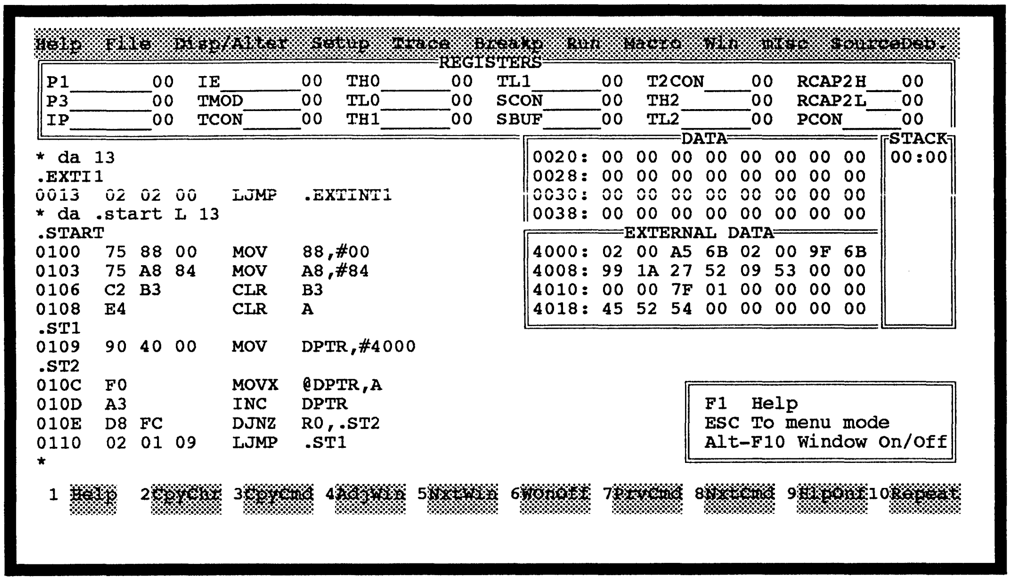
Next take a look at location 13H. To do this, on the Command Line type *DASM 13* and press ENTER. Doing this causes the following new line to appear on the screen:

```
.EXT1
0013    02  02  00    LJMP    .EXTINT1
```

This is the location for the "external interrupt 1" jump vector. As you can see, it is located at 200H.

Disassembling Code

Next take a look at the main program (at 100H or .START) and the 13H locations that follow. To do this, on the Command Line type *DASM .START L 13* and press ENTER. Doing this should cause the lines of disassembled code to scroll up the screen as shown in Figure 9.


Help File Disp/Alter Setup Trace Breakp Run Macro Win misc SourceDeb.
1 Help 2CpyChr 3CpyCmd 4AdjWin 5NxtWin 6WOnOff 7PrvCmd 8NxtCmd 9HlpOnf10Repeat

```
                              REGISTERS
P1_______00  IE_______00  TH0______00  TL1______00  T2CON____00  RCAP2H___00
P3_______00  TMOD_____00  TL0______00  SCON_____00  TH2______00  RCAP2L___00
IP_______00  TCON_____00  TH1______00  SBUF_____00  TL2______00  PCON_____00
                                                  DATA                    STACK
* da 13                                   0020: 00 00 00 00 00 00 00 00   00:00
.EXTI1                                    0028: 00 00 00 00 00 00 00 00
0013  02 02 00   LJMP  .EXTINT1           0030: 00 00 00 00 00 00 00 00
* da .start L 13                          0038: 00 00 00 00 00 00 00 00
.START                                         EXTERNAL DATA
0100  75 88 00   MOV   88,#00             4000: 02 00 A5 6B 02 00 9F 6B
0103  75 A8 84   MOV   A8,#84             4008: 99 1A 27 52 09 53 00 00
0106  C2 B3      CLR   B3                 4010: 00 00 7F 01 00 00 00 00
0108  E4         CLR   A                  4018: 45 52 54 00 00 00 00 00
.ST1
0109  90 40 00   MOV   DPTR,#4000
.ST2
010C  F0         MOVX  @DPTR,A                       F1  Help
010D  A3         INC   DPTR                          ESC To menu mode
010E  D8 FC      DJNZ  R0,.ST2                       Alt-F10 Window On/Off
0110  02 01 09   LJMP  .ST1
*
```

Figure 9. Screen After Disassembly of Several More Lines of Code

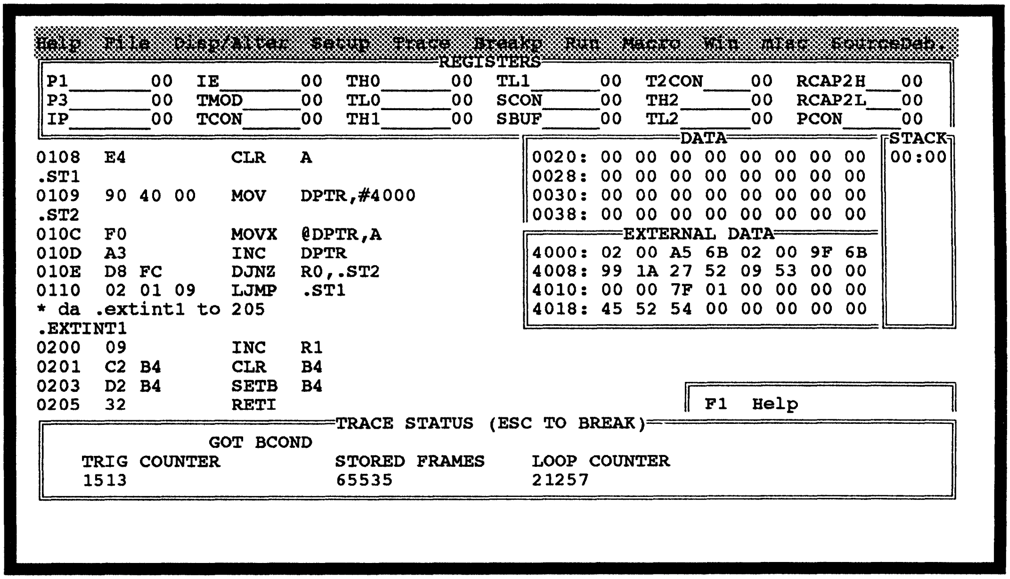
From the disassembly you can see that this part of the program activates external interrupt 1, (which is P3.3), and then pulls it low. This means that the main program will be interrupted after each instruction in the main program. The rest of the main program clears the memory starting at 4000H over and over again.

Next disassemble the external interrupt 1 service routine by typing *DASM .EXTINT1 TO 205* and pressing RETURN. Doing this produces the following:

```
.EXTINT1
0200  09       INC    R1
0201  C2 B4    CLR    .P3+4(T0)
0203  D2 B4    SETB   .P3+4(T0)
0205  32       RETI
```

From the disassembly you can see that the service routine increments R1 and sends out a low pulse on the P3.4 pin. This pulse can be used to check real-time speed by measuring the P3.4 pin with an oscilloscope.

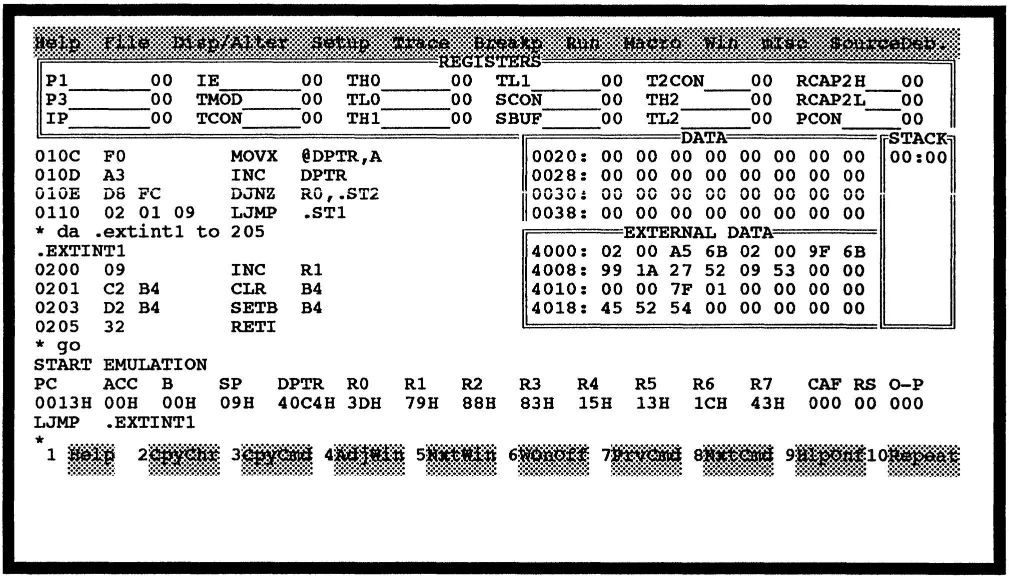
Running the Program Now you're ready to run the program: type *GO* and press ENTER. If a trace board is connected, you'll now see the TRACE STATUS window appear on the screen as shown in Figure 10. (If no trace board is connected, you'll see only the cursor.)


Help File Disp/Alter Setup Trace Breakp Run Macro Win mIsc SourceDeb.

```
                                REGISTERS
P1________00  IE________00  TH0________00  TL1________00  T2CON____00  RCAP2H___00
P3________00  TMOD______00  TL0________00  SCON_______00  TH2______00  RCAP2L___00
IP________00  TCON______00  TH1________00  SBUF_______00  TL2______00  PCON_____00
                                                         DATA                    STACK
0108   E4          CLR   A                 0020: 00 00 00 00 00 00 00 00   00:00
.ST1                                       0028: 00 00 00 00 00 00 00 00
0109   90 40 00    MOV   DPTR,#4000        0030: 00 00 00 00 00 00 00 00
.ST2                                       0038: 00 00 00 00 00 00 00 00
010C   F0          MOVX  @DPTR,A                  EXTERNAL DATA
010D   A3          INC   DPTR              4000: 02 00 A5 6B 02 00 9F 6B
010E   D8 FC       DJNZ  R0,.ST2           4008: 99 1A 27 52 09 53 00 00
0110   02 01 09    LJMP  .ST1              4010: 00 00 7F 01 00 00 00 00
* da .extint1 to 205                       4018: 45 52 54 00 00 00 00 00
.EXTINT1
0200   09          INC   R1
0201   C2 B4       CLR   B4
0203   D2 B4       SETB  B4
0205   32          RETI                              F1  Help
                   TRACE STATUS (ESC TO BREAK)
               GOT BCOND
   TRIG COUNTER           STORED FRAMES     LOOP COUNTER
   1513                   65535             21257
```

Figure 10. Screen After Executing GO Command

Next manually break execution by pressing the ESC key. Doing this causes the contents of the most common registers to appear on the screen as shown in Figure 11.


Help File Disp/Alter Setup Trace Breakp Run Macro Win misc SourceDeb.
1 Help 2CpyChr 3CpyCmd 4AdjWin 5NxtWin 6WOnOff 7PrvCmd 8NxtCmd 9HlpOnf10Repeat

```
                                   REGISTERS
P1_______00  IE_______00  TH0_______00  TL1_______00  T2CON____00  RCAP2H___00
P3_______00  TMOD_____00  TL0_______00  SCON______00  TH2______00  RCAP2L___00
IP_______00  TCON_____00  TH1_______00  SBUF______00  TL2______00  PCON_____00
                                          DATA                       STACK
010C  F0          MOVX  @DPTR,A     0020: 00 00 00 00 00 00 00 00   00:00
010D  A3          INC   DPTR        0028: 00 00 00 00 00 00 00 00
010E  D8 FC       DJNZ  R0,.ST2     0030: 00 00 00 00 00 00 00 00
0110  02 01 09    LJMP  .ST1        0038: 00 00 00 00 00 00 00 00
* da .extint1 to 205                     EXTERNAL DATA
.EXTINT1                            4000: 02 00 A5 6B 02 00 9F 6B
0200  09          INC   R1          4008: 99 1A 27 52 09 53 00 00
0201  C2 B4       CLR   B4          4010: 00 00 7F 01 00 00 00 00
0203  D2 B4       SETB  B4          4018: 45 52 54 00 00 00 00 00
0205  32          RETI
* go
START EMULATION
PC    ACC  B    SP   DPTR  R0  R1  R2  R3  R4  R5  R6  R7  CAF RS O-P
0013H 00H  00H  09H  40C4H 3DH 79H 88H 83H 15H 13H 1CH 43H 000 00 000
LJMP  .EXTINT1
*
```

Figure 11. Typical Screen After Breaking Execution

Note that the next instruction to be executed is shown in disassembled form.

Setting a Breakpoint

To set a breakpoint at address 200H, on the Command Line you type *G T 200* and press ENTER. (*G T 200* is short for *GO TILL 200.*) Doing this causes the register contents at the bottom of the screen to to change as follows:

```
PC    ACC B   SP  DPTR  R0  R1  R2  R3  R4  R5  R6  R7  CAF RS O-P
0201H 00H 00H 09H 40C4H 3DH 7AH 88H 83H 15H 13H 1CH 43H 000 00 000
CLR   .P3+4(T0)
```

From this you can see that the instruction at 200H has been executed, and the next instruction to be executed is at address 201H.

Single Step You can use the STEP command when you want to watch the program execution instruction by instruction. Now type *STEP* and press ENTER. The register contents at the bottom of the screen will change as follows:

```
PC    ACC B   SP  DPTR  R0  R1  R2  R3  R4  R5  R6  R7  CAF RS O-P
0203H 00H 00H 09H 40C4H 3DH 7AH 88H 83H 15H 13H 1CH 43H 000 00 000
SETB .P3+4(T0)
```

Typing *STEP* and pressing ENTER several more times, (or merely pressing the F10 Function Key several more times), will produce the following changes in the contents of PC:

```
PC
0205H
RETI

PC
010EH
DJNZ  R0,.ST2

PC
0013H
LJMP  .EXTINT1

PC
0200H
INC   R1
```

RESET Commands These commands can be used to reset different functions in the emulator. The most common is RESET CHIP, which causes a hardware reset by putting out a high pulse on the Reset pin of the processor. Type *RES CHIP* and press ENTER. Then type *DASM PC* and press ENTER. The following new line will appear on the screen:

```
0000  02 01 00  LJMP  .START
```

As you can see, a line of code is disassembled starting where the Program Counter is pointing, (which is 0 after the processor had been reset).

Mapping By default, the Code Memory is mapped to the emulator and the External Data Memory is mapped to the target system. However, with the MAP command both code and external data are mappable to the target system. To look at the mapping of External Data, now type *MAPX* and press ENTER. Doing this will cause the following lines to scroll up the screen:

```
* mapx
 0K -  4K  = TARGET          4K -  8K = TARGET
 8K - 12K = TARGET          12K - 16K = TARGET
16K - 20K = TARGET          20K - 24K = TARGET
24K - 28K = TARGET          28K - 32K = TARGET
32K - 36K = TARGET          36K - 40K = TARGET
40K - 44K = TARGET          44K - 48K = TARGET
48K - 52K = TARGET          52K - 56K = TARGET
56K - 60K = TARGET          60K - 64K = TARGET
```

If you wanted to map 4K block randomly, you would need to type something like the following: *MAPX=e,t,t,e,e,t* where *e* stands for emulator and *t* stands for target.

You can however map all 64k at once with a single simple command, either *MAPX=EMUL* or *MAPX=TARGET*. To demonstrate this, type *MAPX=EMUL* and press ENTER, then type *MAPX* and press ENTER again. Doing this will cause the following lines to scroll up the screen:

```
* mapx=emul
* mapx
 0K -  4K  = EMULATOR          4K -  8K = EMULATOR
 8K - 12K = EMULATOR          12K - 16K = EMULATOR
16K - 20K = EMULATOR          20K - 24K = EMULATOR
24K - 28K = EMULATOR          28K - 32K = EMULATOR
32K - 36K = EMULATOR          36K - 40K = EMULATOR
40K - 44K = EMULATOR          44K - 48K = EMULATOR
48K - 52K = EMULATOR          52K - 56K = EMULATOR
56K - 60K = EMULATOR          60K - 64K = EMULATOR
```

The MAPC command works like the MAPX command, except the MAPC maps the Code Memory instead of the External Data Memory.

Trace Commands

The goal of this module is to familiarize you with the basic EMUL51 trace commands. Approximate time for working through this practice is 10 minutes.

Loading a File Among the disk files supplied by Nohau Corp. are a sample source file *TEST.SRC* and object file *TEST.A03* including symbols. During installation these files are normally put in the same directory with *EMUL51.EXE*. Starting on the Menu Line, move the cursor to the *File* item and then press ENTER to bring up the pull down menu for that item. Press ENTER again to choose *LOAD*, (which is the command at top of pull down menu). Note that doing this moves the cursor down to the Command Line near the bottom of the screen and prints the name of the selected command *(LOAD)* there also.

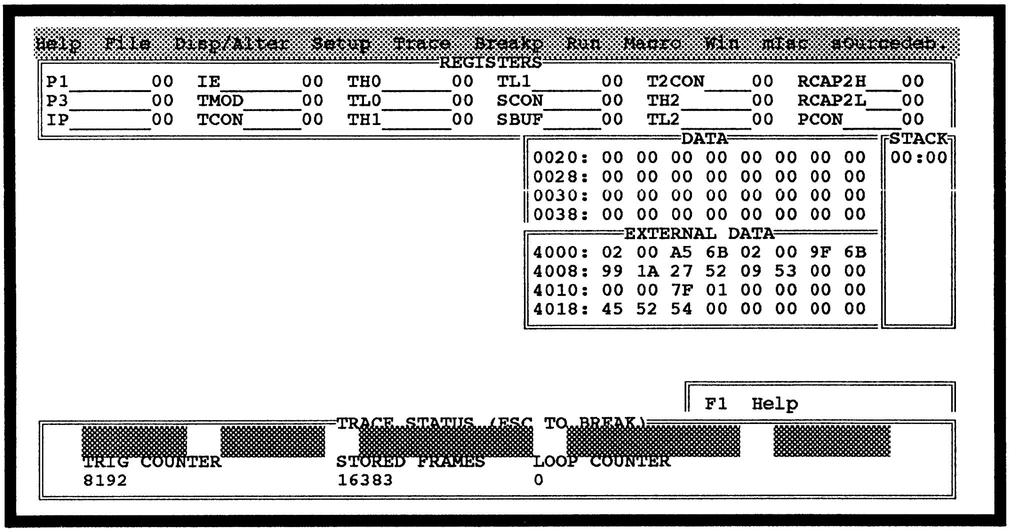
After *LOAD* type *TEST.A03* and press ENTER. After the emulator carries out this command, the asterisk prompt will reappear on a new line.

For convenience, during the rest of this module you will enter commands directly at the command line: Type in what is shown after the prompt (*) and press ENTER. (You could also "fetch" commands from the top menu as in the "User Interface" module.)

Starting Emulation and Trace After loading the test file *TEST.A03*, you first reset the processor chip (to ensure that you will start at 0), and then you start emulation with the GO command:

```
* RES CHI
* G
```

Doing this causes the Trace Window to become visible at the bottom of the screen as shown in Figure 12. The trace starts automatically when the emulation is started.


Help File Disp/Alter Setup Trace Breakp Run Macro Win mIsc sOurcedeb.

```
                              REGISTERS
P1________00  IE________00  TH0_______00  TL1_______00  T2CON____00  RCAP2H___00
P3________00  TMOD______00  TL0_______00  SCON______00  TH2______00  RCAP2L___00
IP________00  TCON______00  TH1_______00  SBUF______00  TL2______00  PCON_____00
                                          DATA                          STACK
                              0020: 00 00 00 00 00 00 00 00             00:00
                              0028: 00 00 00 00 00 00 00 00
                              0030: 00 00 00 00 00 00 00 00
                              0038: 00 00 00 00 00 00 00 00
                                      EXTERNAL DATA
                              4000: 02 00 A5 6B 02 00 9F 6B
                              4008: 99 1A 27 52 09 53 00 00
                              4010: 00 00 7F 01 00 00 00 00
                              4018: 45 52 54 00 00 00 00 00

                                                      F1  Help
                     TRACE STATUS (ESC TO BREAK)
TRIG COUNTER             STORED FRAMES       LOOP COUNTER
8192                     16383               0
```

Figure 12. Screen After Executing GO Command

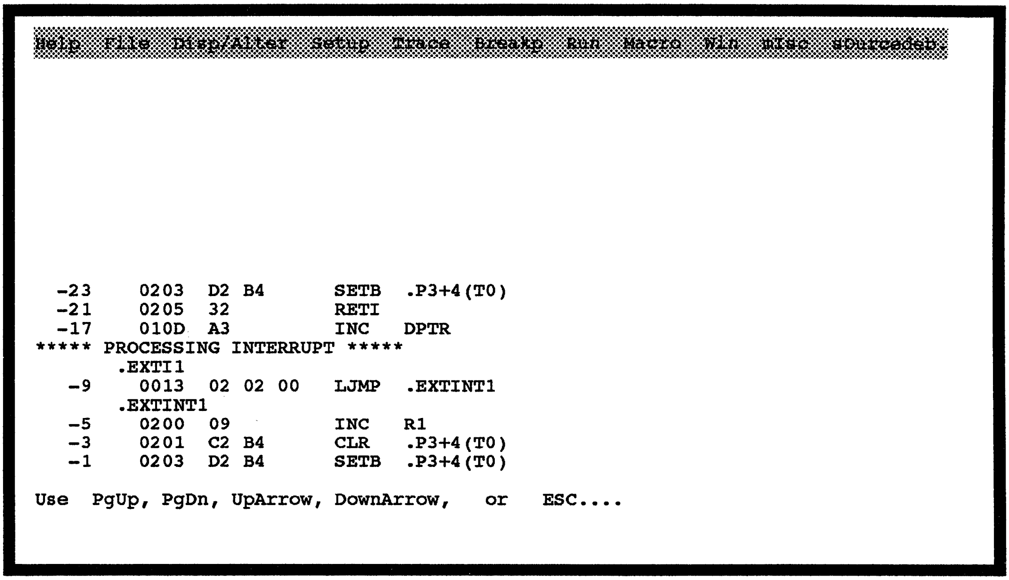
Trace Buffer To look at the contents of the trace buffer, you first have to stop the trace. This can be done in four different ways:

- Press the ESC key, (which also stops the emulation)
- Set a BREAKPOINT
- Use the TE command, (Trace End)
- When the trace buffer fills up after a trace trigger occurs.

This module will demonstrate the first three of these ways. Press ESC and note that doing this also brings back the prompt (*). Then enter the TD command (Trace Display), which causes buffer contents to be displayed in disassembled form.

```
*TD
```

Now your screen should look something that shown in Figure 13.


Help File Disp/Alter Setup Trace Breakp Run Macro Win misc sOurcedeb

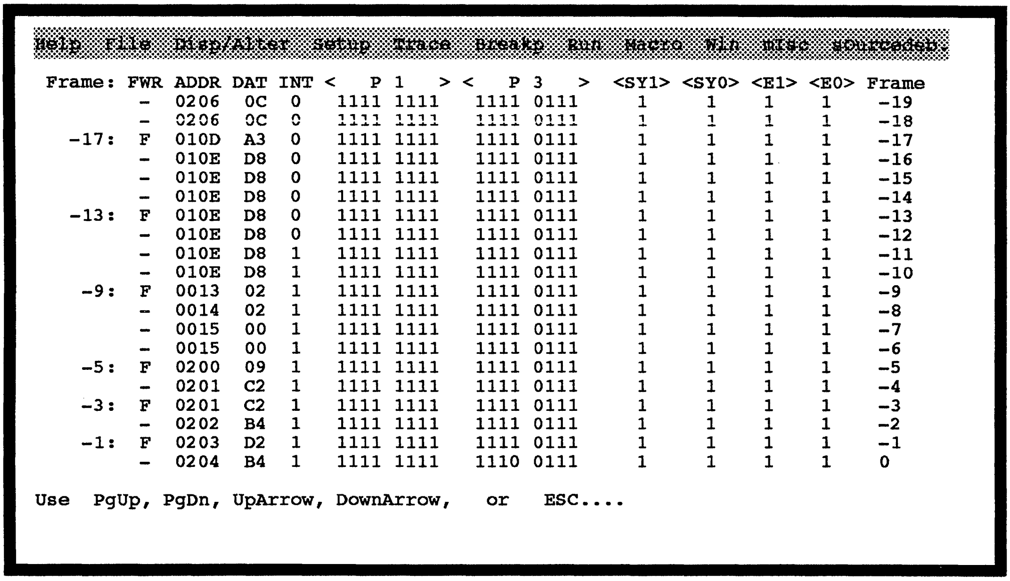
```
  -23     0203  D2 B4      SETB   .P3+4(T0)
  -21     0205  32         RETI
  -17     010D  A3         INC    DPTR
***** PROCESSING INTERRUPT *****
         .EXTI1
   -9     0013  02 02 00   LJMP   .EXTINT1
         .EXTINT1
   -5     0200  09         INC    R1
   -3     0201  C2 B4      CLR    .P3+4(T0)
   -1     0203  D2 B4      SETB   .P3+4(T0)

Use  PgUp, PgDn, UpArrow, DownArrow,   or   ESC....
```

Figure 13. Screen After Executing TD Command

Press ESC. Doing this clears the trace display screen and restores the former screen, which has the prompt (*). Now try the TDF command (Trace Display Frame), which also displays frames collected in the trace buffer, but in frame form.

```
        * TDF
```

Now your screen should look something like that shown in Figure 14.


Help File Disp/Alter Setup Trace Breakp Run Macro Win mIsc sourcedeb.

```
Frame: FWR ADDR DAT INT <   P 1    > <   P 3    >  <SY1> <SY0> <E1> <E0> Frame
        -  0206  0C  0    1111 1111    1111 0111     1     1     1    1    -19
        -  0206  0C  0    1111 1111    1111 0111     1     1     1    1    -18
  -17:  F  010D  A3  0    1111 1111    1111 0111     1     1     1    1    -17
        -  010E  D8  0    1111 1111    1111 0111     1     1     1    1    -16
        -  010E  D8  0    1111 1111    1111 0111     1     1     1    1    -15
        -  010E  D8  0    1111 1111    1111 0111     1     1     1    1    -14
  -13:  F  010E  D8  0    1111 1111    1111 0111     1     1     1    1    -13
        -  010E  D8  0    1111 1111    1111 0111     1     1     1    1    -12
        -  010E  D8  1    1111 1111    1111 0111     1     1     1    1    -11
        -  010E  D8  1    1111 1111    1111 0111     1     1     1    1    -10
   -9:  F  0013  02  1    1111 1111    1111 0111     1     1     1    1    -9
        -  0014  02  1    1111 1111    1111 0111     1     1     1    1    -8
        -  0015  00  1    1111 1111    1111 0111     1     1     1    1    -7
        -  0015  00  1    1111 1111    1111 0111     1     1     1    1    -6
   -5:  F  0200  09  1    1111 1111    1111 0111     1     1     1    1    -5
        -  0201  C2  1    1111 1111    1111 0111     1     1     1    1    -4
   -3:  F  0201  C2  1    1111 1111    1111 0111     1     1     1    1    -3
        -  0202  B4  1    1111 1111    1111 0111     1     1     1    1    -2
   -1:  F  0203  D2  1    1111 1111    1111 0111     1     1     1    1    -1
        -  0204  B4  1    1111 1111    1110 0111     1     1     1    1    0

Use  PgUp, PgDn, UpArrow, DownArrow,   or   ESC....
```

Figure 14. Screen After Executing TDF Command

The trace buffer can hold many more frames than will fit on the screen at any one time. The line at the bottom of the screen in Figure 13 and Figure 14 identifies the keys you can press to scroll through the buffer contents. The numbers along the left side of the screen give the frame numbers: pre-trigger frames have negative numbers, the trigger frame is number zero, and post-trigger frames have positive numbers.

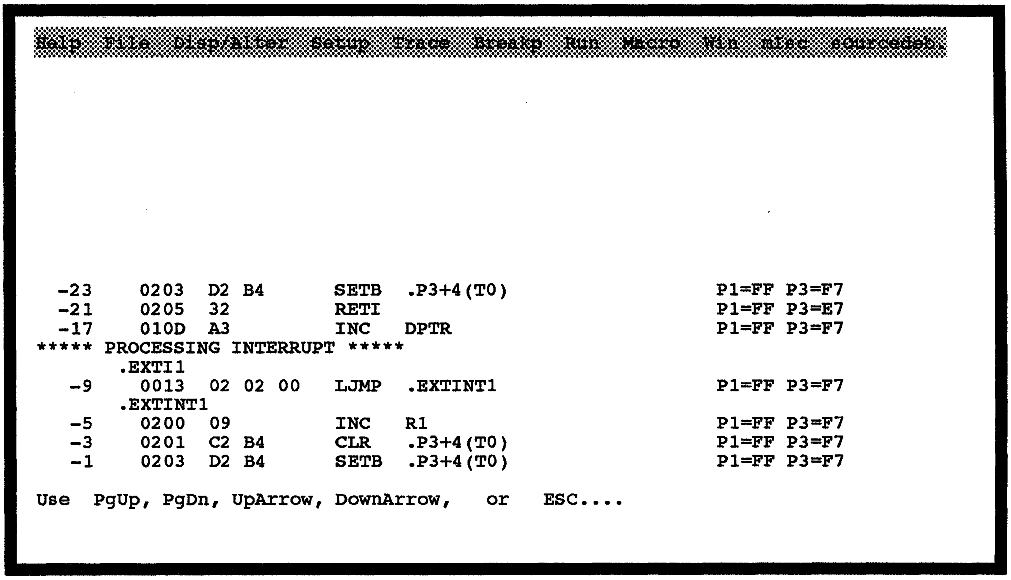
In Figure 14 the column labeled *FWR* tells whether the frame was an instruction fetch (F), a memory write (W), or a memory read (R). The column titles *ADDR* and *DAT* stand for Address and Data. *INT* stands for Interrupt Level, (which is 0 for the Main Program, 1 for a Level 1 interrupt, etc.). The entries under the *P1* and *P3* columns give the states of Ports P1 and P3, expressed in binary form. The entries under *SY1* and *SY0* give the states of the external signals SY1 and SY0. Likewise the entries under *E0* and *E1* give the states of the two extra signals marked E0 and E1 on the POD board. For more details about the format of the frame data, see *How Trace Works* in

Chapter 1.

Press ESC. Now try the TDP command (Trace Display with Ports), which displays trace buffer contents (like the TD command) as well as Ports P1 and P3 (in hexadecimal).

```
* TDF
```

Now your screen should look something like that shown in Figure 15.


Help File Disp/Alter Setup Trace Breakp Run Macro Win misc sourcedeb.

```
  -23     0203  D2 B4      SETB   .P3+4(T0)                 P1=FF P3=F7
  -21     0205  32         RETI                             P1=FF P3=E7
  -17     010D  A3         INC    DPTR                      P1=FF P3=F7
***** PROCESSING INTERRUPT *****
       .EXTI1
   -9     0013  02 02 00   LJMP   .EXTINT1                  P1=FF P3=F7
       .EXTINT1
   -5     0200  09         INC    R1                        P1=FF P3=F7
   -3     0201  C2 B4      CLR    .P3+4(T0)                 P1=FF P3=F7
   -1     0203  D2 B4      SETB   .P3+4(T0)                 P1=FF P3=F7

Use  PgUp, PgDn, UpArrow, DownArrow,   or   ESC....
```

Figure 15. Screen After Executing TDP Command

Trace Window Start the emulation and trace again by entering the following commands:

```
* RES CHI
* G
```

Now we'll take a closer look at the Trace Window that appears at the bottom of the screen when the trace is invoked, (see Figure 12). Labeled fields in the TRACE WINDOW have the following meanings:

TRIG COUNTER: Tells how many frames are stored after trace trigger.
STORED FRAMES: Tells how many frames are stored before trace trigger.
LOOP COUNTER: Tells how many Trace loops have occurred.

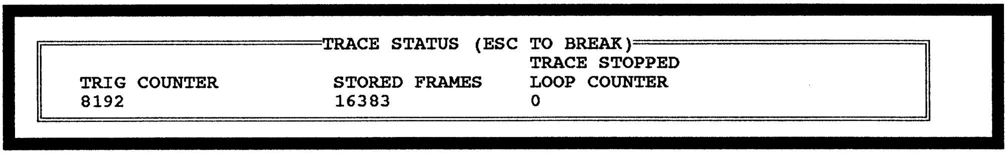
The highlighted boxes above these fields are for the trace conditions; from left to right the definitions for the boxes are:

GOT ACOND
GOT BCOND
TRACE TRIGGERED
LOOP COUNT DONE / TRACE STOPPED
TRACE DONE.

Stopping Trace Only To stop the trace without stopping emulation, next enter the TE command (Trace End):

```
> TE
```

Note that the prompt has changed to a different character. The message *TRACE STOPPED* now appears inside the Trace Window, (Figure 16), and on the POD board the LED continues to be lit, (which indicates that the emulation is still active). At this point you could restart the trace by entering the TB command, (Trace Begin).



```
                    TRACE STATUS (ESC TO BREAK)
                                      TRACE STOPPED
TRIG COUNTER            STORED FRAMES LOOP COUNTER
8192                    16383         0
```

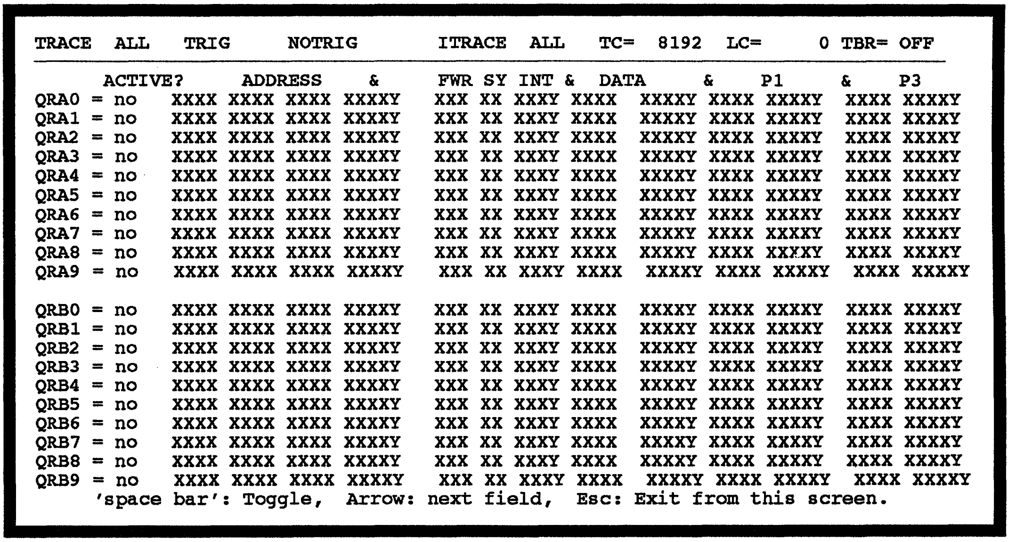
Figure 16. Trace Window After Executing TE Command

So far you have seen how the trace always starts automatically when emulation is started, and have "taken snapshots" of what happened just before the ESC key was pressed or before you entered the TE command. The trace buffer will always show the last 16K frames, (or last 4K frames if you have a 4K trace).

Next we'll discuss how to stop the trace in a more controlled way. You will be working with the TS window, which is accessed by entering the TS command (Trace Setup). Press ESC and enter the following:

```
*TS
```

Doing this will bring up the TS window as shown in Figure 17.



```
TRACE  ALL   TRIG     NOTRIG      ITRACE  ALL   TC=  8192  LC=     0 TBR= OFF

       ACTIVE?     ADDRESS    &        FWR SY INT &  DATA      &     P1      &    P3
QRA0 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA1 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA2 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA3 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA4 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA5 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA6 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA7 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA8 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA9 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY

QRB0 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB1 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB2 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB3 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB4 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB5 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB6 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB7 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB8 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB9 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
     'space bar': Toggle,  Arrow: next field,  Esc: Exit from this screen.
```

Figure 17. Screen After Executing TS Command

Given below for the TS window are explanations of the functions of fields in the lines at the top of the screen.

TRACE This field is for filtering of your trace captures. The available functions are a TRUE condition relative to the qualifying registers (QRA or QRB).

TRIG This field lets you select which function, (if any), that you want to trigger the trace capture.

ITRACE This field is another filter, but is for filtering the interrupt tracing.

TC= This if the field that you would use to adjust the amount of PRE/POST trigger that you wish to see. The default is set to the half-way mark.

LC= This field, LOOP COUNT, is used to selectively choose after which triggered pass to start your trace capture.

TBR= This field is used to turn ON/OFF all of the active qualifying registers as breakpoints.

ACTIVE? These fields are use to activate each of the qualifying registers.

ADDRESS In this field you can specify what address or address range you wish to use to qualify your trace capture.

MISC This is the most complex field in the trace setup because it allows you to specify some very specific conditions to qualify the condition. We will examine each of the bit placements under this field.

BIT#	DESCRIPTION
15	Valid fetch (VF): This is the first opcode fetch of an instuction (start of instruction).
14	Write (WR): This specifies a write function to the external data area only.
13	Read (RD): This specifies a read function to the external data area only.
12	(SY1): This is one of the probes on the pod board that you can use to trigger the trace from an external source.
11	(SY0): This is another of the probes on the pod board that you can use to trigger the trace from an external source.
10-8	(I2,I1,I0): Interrupt level.

DATA 7-0 (DATA): Value of data desired.

P1 This field normally traces port 1, but you can hook up wires to the input pin on the pod and trace other external signals.

P3 This field is like P1, except for bits 6 and 7 on some PODs (which are for the inputs E0 and E1, respectively).

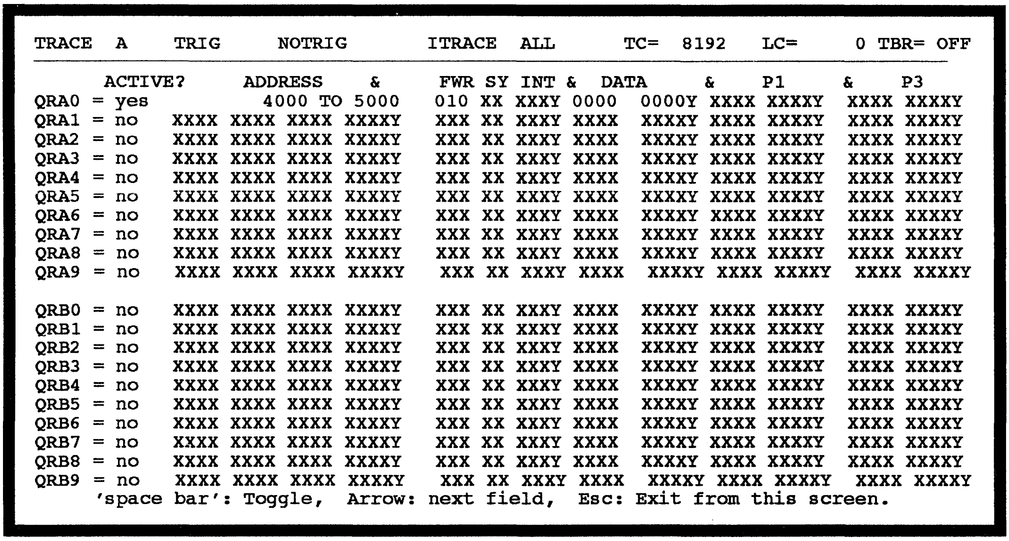
QRA0 through QRB9 ,(along left side of the screen), stand for Qualification Registers A0-A9 and Qualification Registers B0-B9. Contents of fields for the QRA and QRB registers are interpreted as follows:

Y = Binary pattern
X = Don't care
1 = True or High signal level
0 = False or Low signal level.

Several examples are given next to demonstrate how changes to the trace setup will have an affect on what is saved in the trace buffer.

Setup Example 1 By performing the following steps you create a trace setup that will yield a trace capture of ONLY the write to address range 4000h to 5000h with the data of 00h.

Starting from the screen shown in Figure 17, use the cursor arrow keys to move the cursor into the highlighted field after the word *TRACE* on the top line of the screen, (if cursor is not already there). Next press the SPACE bar; note that the contents of this field change each time you do this. Keep pressing and releasing the SPACE bar until *A* appears in the *TRACE* field. After that, move the cursor down into the first field after *QRA0 =* and press the SPACE bar to bring *YES* into this field. Move the cursor into the next field, (*ADDRESS* for QRA0), and type *4000 TO 5000.* Move the cursor into the next field, (*MISC* for QRA0), and type *010X XXXXY.* Move the cursor into the next field, (*DATA* for QRA0) and type *0000 0000Y.* Your screen should now like the Trace Setup screen shown in Figure 18.

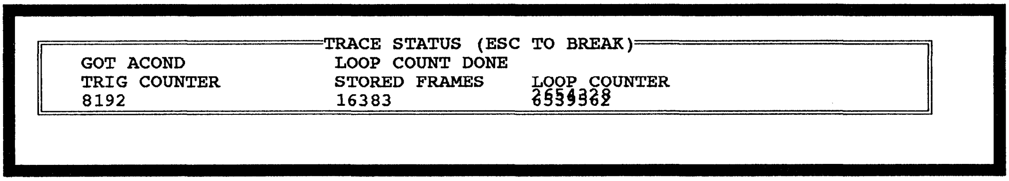


```
TRACE  A    TRIG      NOTRIG       ITRACE  ALL      TC=  8192   LC=      0 TBR= OFF

     ACTIVE?      ADDRESS     &       FWR SY INT &  DATA      &      P1       &     P3
QRA0 = yes           4000 TO 5000    010 XX XXXY 0000   0000Y XXXX XXXXY   XXXX XXXXY
QRA1 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRA2 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRA3 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRA4 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRA5 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRA6 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRA7 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRA8 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRA9 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY

QRB0 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRB1 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRB2 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRB3 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRB4 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRB5 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRB6 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRB7 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRB8 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
QRB9 = no     XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX   XXXXY XXXX XXXXY   XXXX XXXXY
    'space bar': Toggle,  Arrow: next field,  Esc: Exit from this screen.
```

Figure 18. Screen for Trace Setup Example 1

Press ENTER and then press ESC. Now restart the emulation and trace by entering the following commands:

* RES CHI
* G
After a few minutes, the Trace Window at the bottom of the screen should look something like Figure 19.

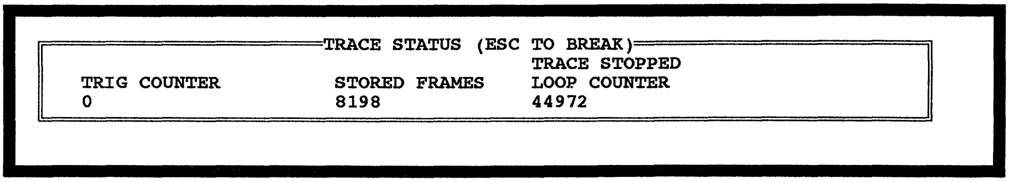


```
╔═══════════════════════════TRACE STATUS (ESC TO BREAK)══════════════════════════╗
║  GOT ACOND                  LOOP COUNT DONE                                   ║
║  TRIG COUNTER               STORED FRAMES      LOOP COUNTER                   ║
║                                                2654328                        ║
║  8192                       16383              6539562                        ║
╚═══════════════════════════════════════════════════════════════════════════════╝
```

Figure 19. Trace Window After Executing TE Command

Note that the value under *LOOP COUNTER* is a blur, because the emualtion and trace are both still running. No trigger is being used, so you will have to stop the trace manually, (with the ESC key or the TE command). Enter the following:
* TE

Doing this will cause the trace status to change to *TRACE STOPPED*, (see Figure 20).



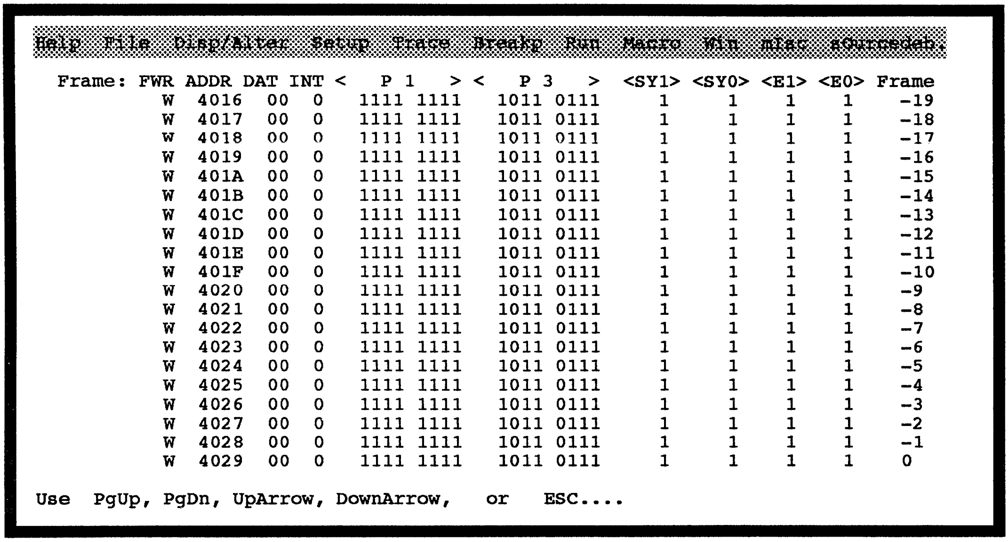
```
=======================TRACE STATUS (ESC TO BREAK)=========================
                                            TRACE STOPPED
 TRIG COUNTER              STORED FRAMES    LOOP COUNTER
 0                         8198             44972
```

Figure 20. Trace Window After Stopping Trace

Press ESC. Then use the TDF command (Trace Display Frame) to look at the contents of the trace buffer:

* TDF

Now your screen should look something like that shown in Figure 21.


Help File Disp/Alter Setup Trace Breakp Run Macro Win mIsc sOurcedeb.

```
Frame: FWR ADDR DAT INT <   P 1    > <   P 3    >  <SY1> <SY0> <E1> <E0> Frame
        W  4016  00  0   1111 1111   1011 0111     1     1     1    1    -19
        W  4017  00  0   1111 1111   1011 0111     1     1     1    1    -18
        W  4018  00  0   1111 1111   1011 0111     1     1     1    1    -17
        W  4019  00  0   1111 1111   1011 0111     1     1     1    1    -16
        W  401A  00  0   1111 1111   1011 0111     1     1     1    1    -15
        W  401B  00  0   1111 1111   1011 0111     1     1     1    1    -14
        W  401C  00  0   1111 1111   1011 0111     1     1     1    1    -13
        W  401D  00  0   1111 1111   1011 0111     1     1     1    1    -12
        W  401E  00  0   1111 1111   1011 0111     1     1     1    1    -11
        W  401F  00  0   1111 1111   1011 0111     1     1     1    1    -10
        W  4020  00  0   1111 1111   1011 0111     1     1     1    1    -9
        W  4021  00  0   1111 1111   1011 0111     1     1     1    1    -8
        W  4022  00  0   1111 1111   1011 0111     1     1     1    1    -7
        W  4023  00  0   1111 1111   1011 0111     1     1     1    1    -6
        W  4024  00  0   1111 1111   1011 0111     1     1     1    1    -5
        W  4025  00  0   1111 1111   1011 0111     1     1     1    1    -4
        W  4026  00  0   1111 1111   1011 0111     1     1     1    1    -3
        W  4027  00  0   1111 1111   1011 0111     1     1     1    1    -2
        W  4028  00  0   1111 1111   1011 0111     1     1     1    1    -1
        W  4029  00  0   1111 1111   1011 0111     1     1     1    1    0

Use  PgUp, PgDn, UpArrow, DownArrow,   or   ESC....
```

Figure 21. TDF Display for Trace Example 1

Setup Example 2 In this example you will change the trace setup so that the trace will trigger on a code fetch between address 100h and 300h. When the A condition becomes true, the TC trace counter will begin decrementing and tracing will stop when the TC reaches 0.

The steps you have to go through are much like the ones in Trace Example 1: use the TS command to bring up the trace setup screen, then move the cursor into the fields you want to change and type in the changes. Enter the TS command.

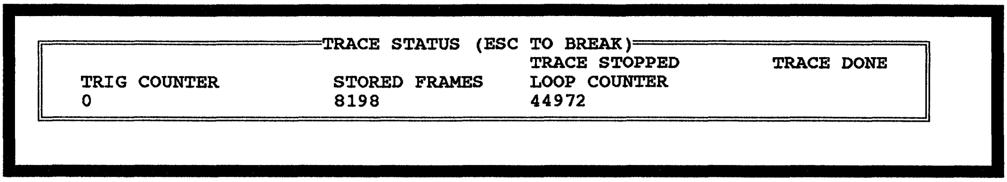
* TS

Move the cursor into the *TRACE* field and press SPACE until *ALL* appears. Move into the *TRIG* field and press SPACE until *A* appears. Move the cursor into the *ADDRESS* field for QRA0, then change the entry to read *100 TO 300.* Move the cursor to the *MISC* field for QRA0 and type *1XXX XXXY.* Move the cursor into the *DATA* field for QRA0, then change the entry to read *XXXX XXXXY.* Your screen should now look like Figure 23.

Press ENTER and then press ESC. Now restart the emulation and trace by entering the following commands:

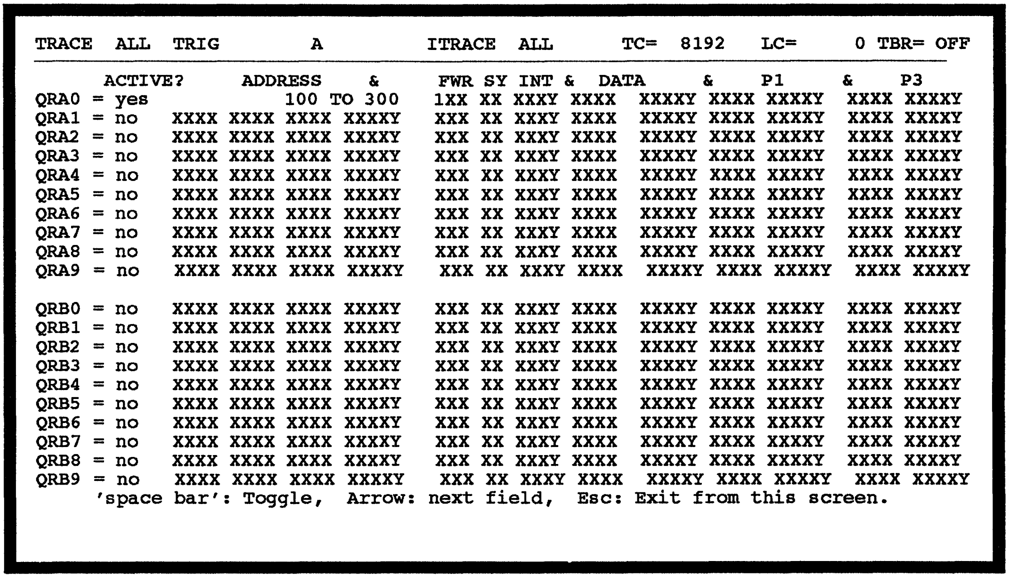
* RES CHI
* G

After a few minutes, the Trace Window at the bottom of the screen should look something like Figure 22.



```
=====================TRACE STATUS (ESC TO BREAK)=====================
                                        TRACE STOPPED           TRACE DONE
   TRIG COUNTER          STORED FRAMES  LOOP COUNTER
   0                     8198           44972
```

Figure 22. Trace Window After Stopping Trace



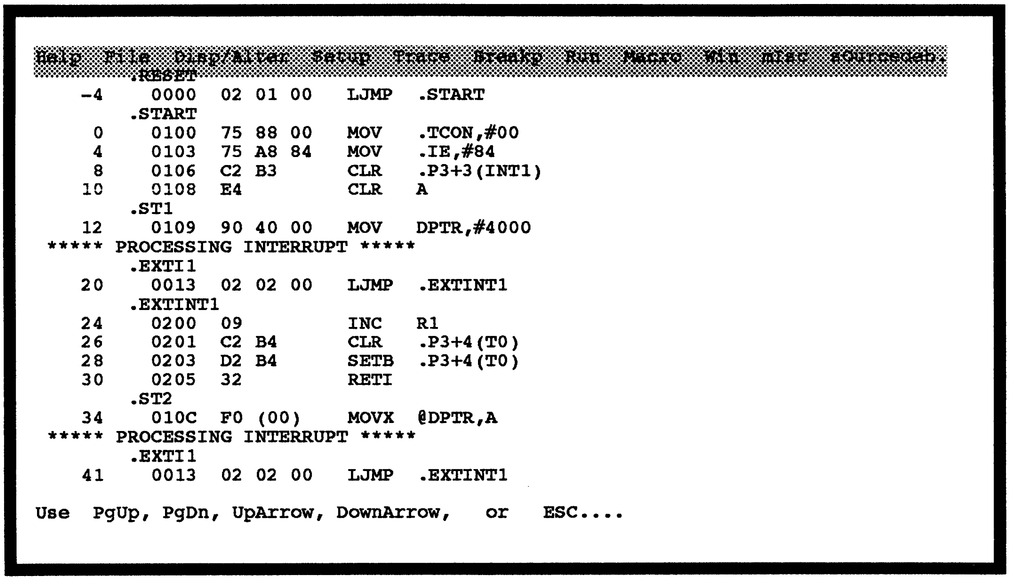
```
TRACE  ALL  TRIG         A          ITRACE  ALL      TC=  8192   LC=      0 TBR= OFF
------------------------------------------------------------------------------
      ACTIVE?      ADDRESS     &      FWR SY INT &  DATA      &    P1      &    P3
QRA0 = yes             100 TO 300     1XX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA1 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA2 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA3 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA4 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA5 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA6 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA7 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA8 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA9 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY

QRB0 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB1 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB2 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB3 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB4 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB5 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB6 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB7 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB8 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB9 = no    XXXX XXXX XXXX XXXXY     XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
    'space bar': Toggle,  Arrow: next field,  Esc: Exit from this screen.
```

Figure 23. Screen for Trace Setup Example 2

This time you'll use the TD command to look at the trace buffer. Enter the following:

\>TD

Now your screen should look something like that shown in Figure 24.


Help File Disp/Alter Setup Trace Breakp Run Macro Win misc sOurcedeb.

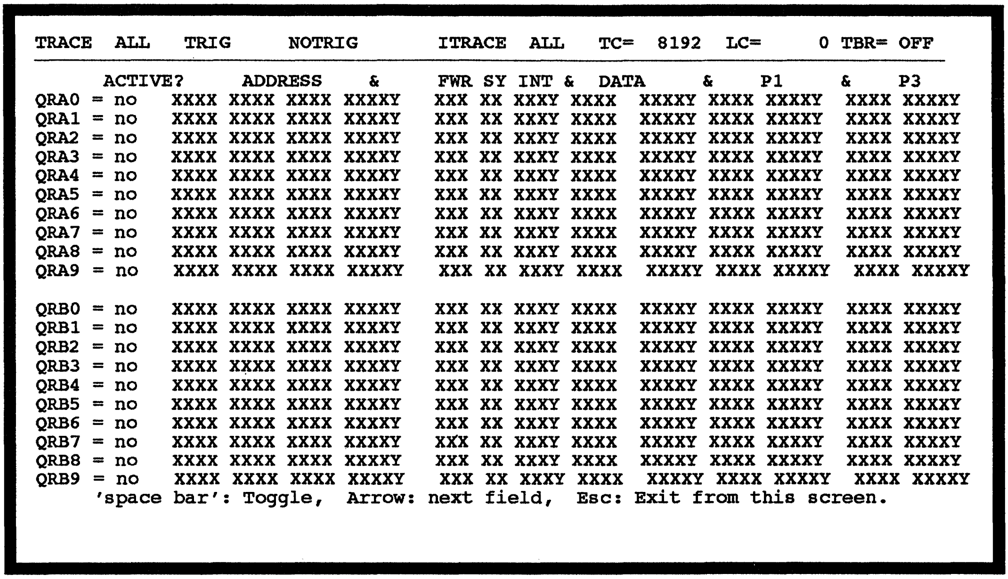
```
         .RESET
  -4      0000  02 01 00   LJMP  .START
         .START
   0      0100  75 88 00   MOV   .TCON,#00
   4      0103  75 A8 84   MOV   .IE,#84
   8      0106  C2 B3      CLR   .P3+3(INT1)
  10      0108  E4         CLR   A
         .ST1
  12      0109  90 40 00   MOV   DPTR,#4000
 ***** PROCESSING INTERRUPT *****
         .EXTI1
  20      0013  02 02 00   LJMP  .EXTINT1
         .EXTINT1
  24      0200  09         INC   R1
  26      0201  C2 B4      CLR   .P3+4(T0)
  28      0203  D2 B4      SETB  .P3+4(T0)
  30      0205  32         RETI
         .ST2
  34      010C  F0 (00)    MOVX  @DPTR,A
 ***** PROCESSING INTERRUPT *****
         .EXTI1
  41      0013  02 02 00   LJMP  .EXTINT1

Use  PgUp, PgDn, UpArrow, DownArrow,   or   ESC....
```

Figure 24. Screen for TD Command in Trace Setup Example 2

Note the Pre- and Post-trigger information. The numbers along the left side of the screen are the frame numbers.

Setup Example 3 In this example you will set up the trace to filter out everything except the interrupts that are being processed.

Press ESC to stop emulation. Then enter the TS command on the command line.
* TS
Doing this brings up the TS window so that you can change the entries in the fields. Move the cursor into the *TRIG* field and press SPACE until *NOTRIG* appears. Move the cursor into the *ITRACE* field and press SPACE until *INT* appears. Move the cursor into the first field after *QRA0 =* and press SPACE until *no* appears. Move the cursor into the *ADDRESS* field for QRA0, then change the entry to *XXXX XXXX XXXX XXXXY*. Move the cursor into the *MISC* field and change the entry to read *XXXX XXXXY*. Move the cursor into the *DATA* field for QRA0, then change the entry to read *XXXX XXXXY*. Your screen should now look like Figure 25.



```
TRACE  ALL   TRIG     NOTRIG        ITRACE  ALL   TC=  8192  LC=      0 TBR= OFF

      ACTIVE?     ADDRESS     &     FWR SY INT &  DATA     &     P1      &    P3
QRA0 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA1 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA2 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA3 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA4 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA5 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA6 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA7 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA8 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA9 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY

QRB0 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB1 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB2 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB3 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB4 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB5 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB6 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB7 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB8 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB9 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
     'space bar': Toggle,  Arrow: next field,  Esc: Exit from this screen.
```

Figure 25. Screen for Trace Setup Example 3

Press ENTER and then press ESC. Now restart the emulation and trace by entering the following commands:

* RES CHI

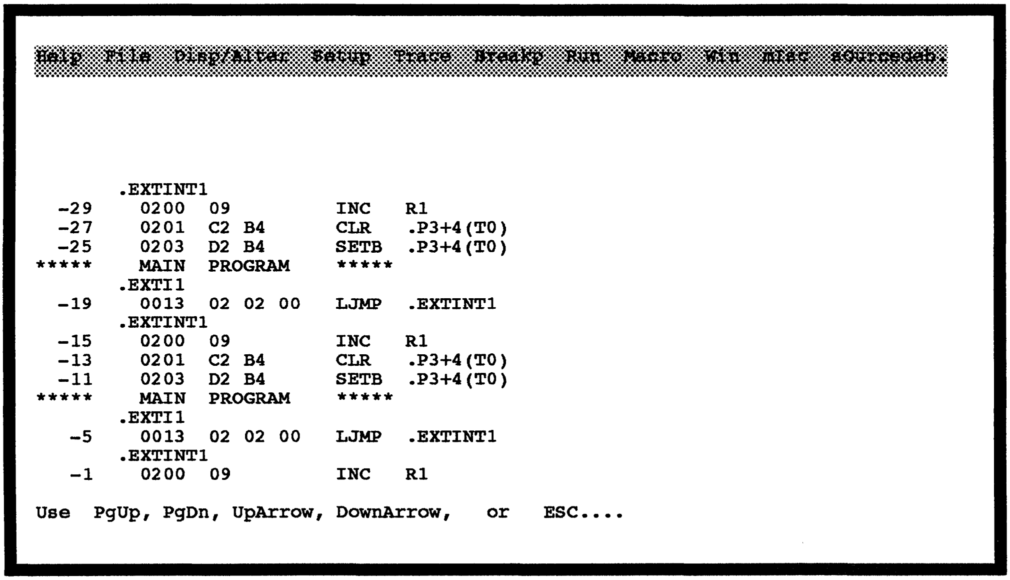
* G

To see the effects of this setup, stop the trace with the TE command and then use TD.

> TE

> TD

Now your screen should look something like that shown in Figure 26.


Help File Disp/Alter Setup Trace Breakp Run Macro Win misc sourcedeb.

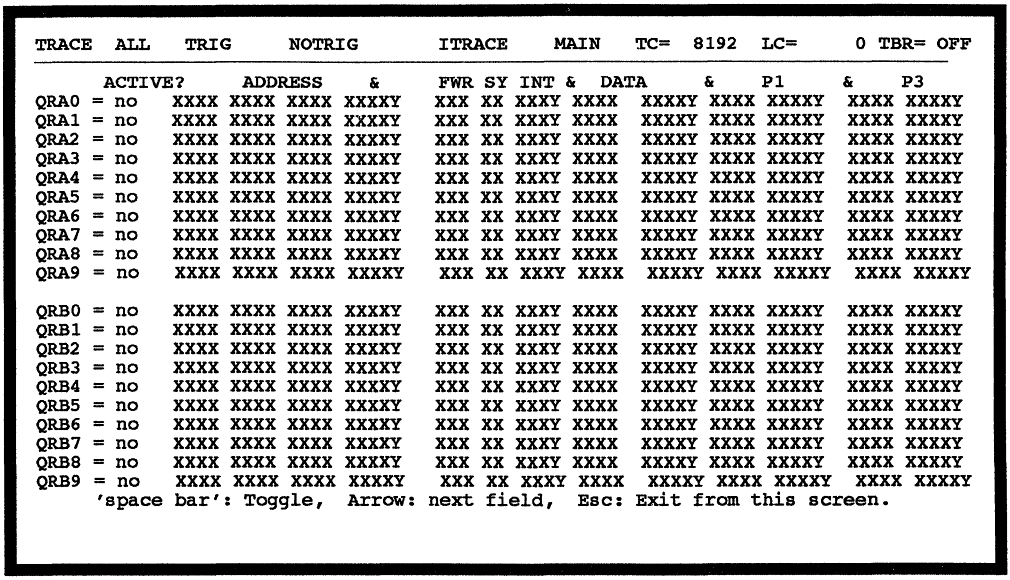
```
          .EXTINT1
  -29      0200  09          INC    R1
  -27      0201  C2 B4       CLR    .P3+4(T0)
  -25      0203  D2 B4       SETB   .P3+4(T0)
*****      MAIN  PROGRAM     *****
          .EXTI1
  -19      0013  02 02 00    LJMP   .EXTINT1
          .EXTINT1
  -15      0200  09          INC    R1
  -13      0201  C2 B4       CLR    .P3+4(T0)
  -11      0203  D2 B4       SETB   .P3+4(T0)
*****      MAIN  PROGRAM     *****
          .EXTI1
   -5      0013  02 02 00    LJMP   .EXTINT1
          .EXTINT1
   -1      0200  09          INC    R1

Use  PgUp, PgDn, UpArrow, DownArrow,   or   ESC....
```

Figure 26. Screen for TD Command in Trace Setup Example 3

Setup Example 4 To complement what you did in the previous example, next you will set up the trace to filter out <u>all</u> the interrupt processing. Press ESC to stop emulation. Then enter the TS command on the command line.

* TS

In the TS window you only need to change one entry. Move the cursor into the *ITRACE* field and press SPACE until *MAIN* appears, (see Figure 27).



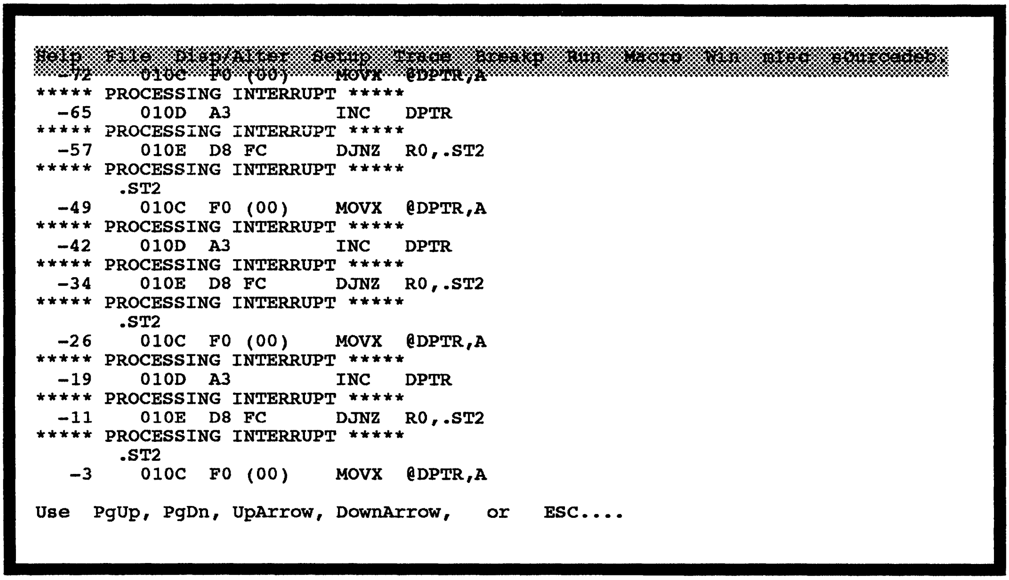
```
TRACE  ALL    TRIG      NOTRIG        ITRACE    MAIN   TC=  8192   LC=      0 TBR= OFF

      ACTIVE?     ADDRESS     &       FWR SY INT &  DATA      &     P1      &     P3
QRA0 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA1 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA2 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA3 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA4 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA5 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA6 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA7 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA8 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRA9 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY

QRB0 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB1 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB2 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB3 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB4 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB5 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB6 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB7 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB8 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
QRB9 = no    XXXX XXXX XXXX XXXXY   XXX XX XXXY XXXX  XXXXY XXXX XXXXY  XXXX XXXXY
     'space bar': Toggle,  Arrow: next field,  Esc: Exit from this screen.
```

Figure 27. Screen for Trace Setup Example 4

To see the effects of this setup, stop the trace with the TE command and then use TD.

```
> TE
> TD
```

Now your screen should look something like that shown in Figure 28.



Help File Disp/Alter Setup Trace Breakp Run Macro Win misc sOurcedeb.

```
  -72     010C  F0 (00)     MOVX  @DPTR,A
***** PROCESSING INTERRUPT *****
  -65     010D  A3          INC   DPTR
***** PROCESSING INTERRUPT *****
  -57     010E  D8 FC       DJNZ  R0,.ST2
***** PROCESSING INTERRUPT *****
       .ST2
  -49     010C  F0 (00)     MOVX  @DPTR,A
***** PROCESSING INTERRUPT *****
  -42     010D  A3          INC   DPTR
***** PROCESSING INTERRUPT *****
  -34     010E  D8 FC       DJNZ  R0,.ST2
***** PROCESSING INTERRUPT *****
       .ST2
  -26     010C  F0 (00)     MOVX  @DPTR,A
***** PROCESSING INTERRUPT *****
  -19     010D  A3          INC   DPTR
***** PROCESSING INTERRUPT *****
  -11     010E  D8 FC       DJNZ  R0,.ST2
***** PROCESSING INTERRUPT *****
       .ST2
   -3     010C  F0 (00)     MOVX  @DPTR,A

Use  PgUp, PgDn, UpArrow, DownArrow,   or   ESC....
```

Figure 28. Screen for TD Command in Trace Setup Example 4

Trace Breakpoint Reg. The trace board is provided with a Trace Breakpoint Register that makes it possible for the trace conditions to activate a breakpoint that stops the emulation as well as ends the trace. Any combination of the 48-bits wide trace frame can be used.

You have two ways to toggle the Trace Breakpoint Register ON and OFF. Entering the command *TBR ON* will toggle it on, and entering *TBR OFF* will toggle it off. (Entering *TBR* alone will cause the current TBR ON/OFF status to be displayed on the command line.)

The other way to set TBR to ON or OFF is from the trace setup. Note that *TBR=* is the last field on the top line in the TS Window, (Figure 27). You move the cursor into the *TBR=* field and press SPACE until the desired status, *ON* or *OFF*, appears.

Qualification Reg. For the examples in this Tutorial you have set up conditions for Qualification Register QRA0 only. However, all the other 19 registers, (QRA1 through QRA9 and QRB0 through QRB9), are also available as well. For each machine cycle executed by the 8051 (or other processor), the trace board compares the 48 bits in the frame with the pre-programmed A- and B- conditions. Because each condition actually consists of five separate fields (ADDRESS, MISC, DATA, P1 and P3), comparisons are made for each field separately and then the result for each field is vertically OR'ed (QRA0 - QRA9). The result for each field is then AND'ed with the other fields (ADDRESS * MISC * DATA * P1 *P3). This is done independently for both the A- and B-conditions. To be TRUE, each "FIELD" comparison has to match exactly with the current frame data from the active machine cycle.

Advanced Trace Commands

The goal of this module is to familiarize you with the advanced trace board and its commands. Approximate time for this portion of the practice is 10-15 minutes. This tutorial will attempt to show you equivalent setups for the 256K and 64K trace boards that will have the same results as the standard 16K and 4K trace boards.

Example command line to start emulator from DOS:

```
c:\emul51> emul51 -p8031 -e110 -t100 -m128 -f64
```

Among the disk files supplied by Nohau Corp. is a sample source file ***TEST.SRC*** and the object file ***TEST.A03*** including symbols. During installation of the emulator software, these files are put in the same directory as the ***EMUL51.EXE***. Starting on the Menu Line, move the cursor to the *file* item and the press **ENTER** to bring up the pull down menu for that item. Press **ENTER** again to choose ***LOAD***, (which is the command at the top of the pull down menu). Note that doing this moves the cursor down to the Command Line near the bottom of the screen and prints the name of the selected command (*LOAD*) there also.

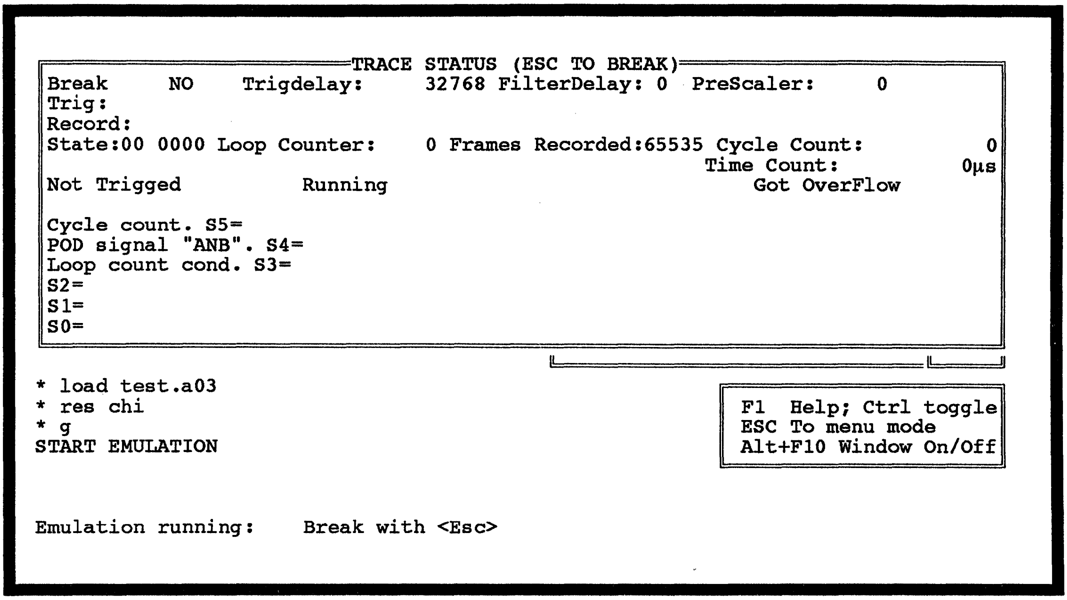
After *LOAD*, type *TEST.A03* and press **ENTER**. After the emulator carries out the command, the asterisk prompt will reappear on a new line.

For convenience, during the rest of this module you will enter commands directly at the command line: Type in what is shown after the prompt (*) and press **ENTER**. (You could also "fetch" the command from the top menu as in the "User Interface" module.)

Starting Emulation and Trace After loading the test file ***TEST.A03*** you first reset the processor chip (to ensure that you will start at location 0), and then you start the emulation with the **GO** command:

```
* res chi
* g
```

Doing this causes the TRACE Status Window to become visible at the top of the screen as shown in Figure 29. The trace starts automatically when the emulation is started.



```
═══════════════════════════TRACE STATUS (ESC TO BREAK)═══════════════════════════
Break      NO     Trigdelay:     32768 FilterDelay: 0  PreScaler:      0
Trig:
Record:
State:00 0000 Loop Counter:     0 Frames Recorded:65535 Cycle Count:           0
                                                        Time Count:          0µs
Not Trigged           Running                               Got OverFlow

Cycle count. S5=
POD signal "ANB". S4=
Loop count cond. S3=
S2=
S1=
S0=

* load test.a03
* res chi                                                F1  Help; Ctrl toggle
* g                                                      ESC To menu mode
START EMULATION                                          Alt+F10 Window On/Off


Emulation running:    Break with <Esc>
```

Figure 29. Screen After the GO Command

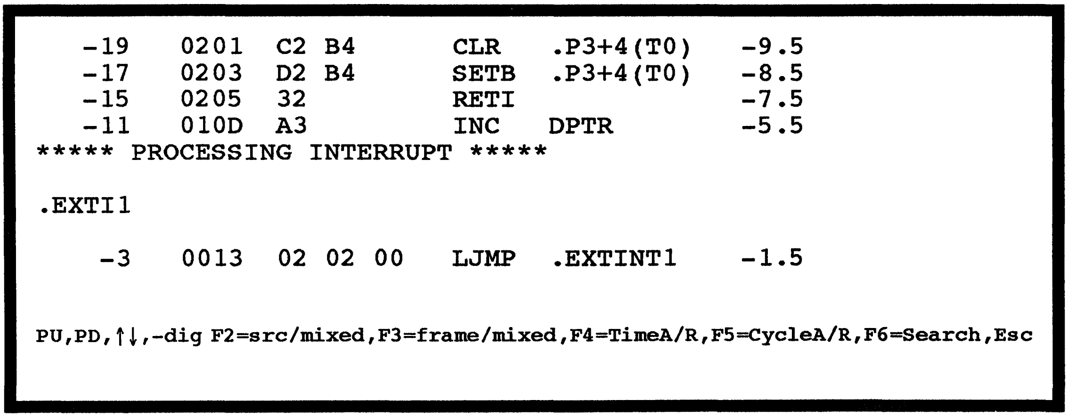
Trace Buffer To look at the contents of the trace buffer, you first have to stop the trace. This can be done in four different ways:

- Press the **ESC** key. (which stops the emulation)
- Use the **TE** command. (Trace End)
- When the trace buffer fills up after a trace trigger occurs.
- Set a BREAKPOINT

This module will demonstrate the first three methods listed above. Press **ESC** and note that doing this also brings back the command prompt (*). Then enter the **TD** command (Trace Display), which causes the buffer contents to be displayed in disassembled form.

```
* TD
```

Now your screen should look something like Figure 30.

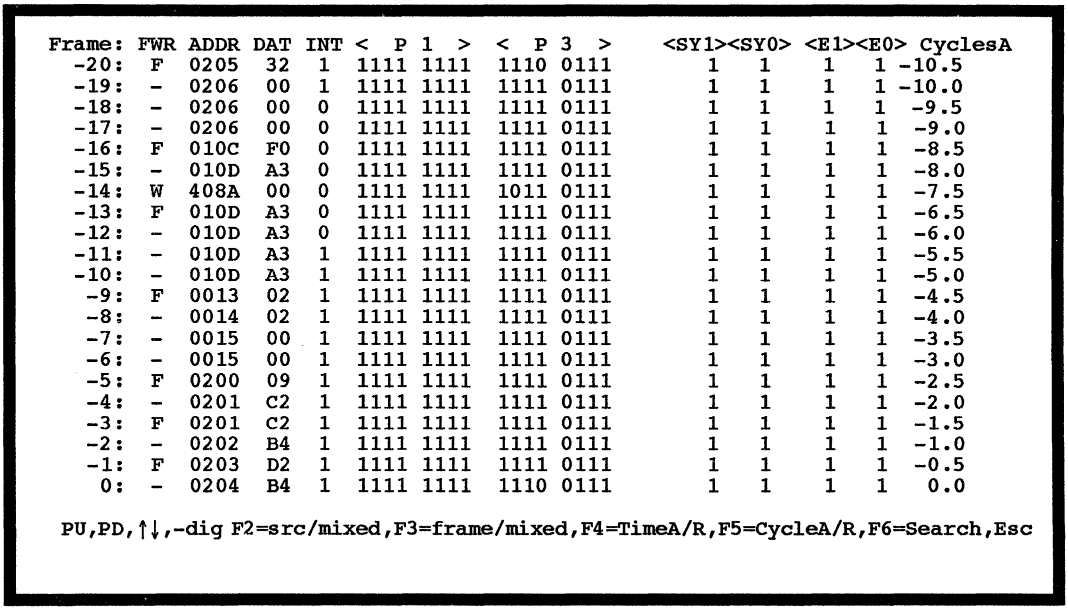


```
    -19    0201   C2 B4       CLR    .P3+4(T0)    -9.5
    -17    0203   D2 B4       SETB   .P3+4(T0)    -8.5
    -15    0205   32          RETI                -7.5
    -11    010D   A3          INC    DPTR         -5.5
***** PROCESSING INTERRUPT *****

.EXTI1

     -3    0013   02 02 00    LJMP   .EXTINT1     -1.5


PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 30. Screen after the TD command

Now you can display the trace buffer in the FRAMES mode by either pressing the ESC key and then typing in TDF at the command prompt (*), or by pressing the **F3** function key.

Now your screen should look similar to Figure 31.



```
Frame: FWR ADDR DAT INT <  P 1  >   <  P 3  >      <SY1><SY0> <E1><E0> CyclesA
  -20:  F  0205  32  1  1111 1111   1110 0111         1    1     1   1 -10.5
  -19:  -  0206  00  1  1111 1111   1111 0111         1    1     1   1 -10.0
  -18:  -  0206  00  0  1111 1111   1111 0111         1    1     1   1  -9.5
  -17:  -  0206  00  0  1111 1111   1111 0111         1    1     1   1  -9.0
  -16:  F  010C  F0  0  1111 1111   1111 0111         1    1     1   1  -8.5
  -15:  -  010D  A3  0  1111 1111   1111 0111         1    1     1   1  -8.0
  -14:  W  408A  00  0  1111 1111   1011 0111         1    1     1   1  -7.5
  -13:  F  010D  A3  0  1111 1111   1111 0111         1    1     1   1  -6.5
  -12:  -  010D  A3  0  1111 1111   1111 0111         1    1     1   1  -6.0
  -11:  -  010D  A3  1  1111 1111   1111 0111         1    1     1   1  -5.5
  -10:  -  010D  A3  1  1111 1111   1111 0111         1    1     1   1  -5.0
   -9:  F  0013  02  1  1111 1111   1111 0111         1    1     1   1  -4.5
   -8:  -  0014  02  1  1111 1111   1111 0111         1    1     1   1  -4.0
   -7:  -  0015  00  1  1111 1111   1111 0111         1    1     1   1  -3.5
   -6:  -  0015  00  1  1111 1111   1111 0111         1    1     1   1  -3.0
   -5:  F  0200  09  1  1111 1111   1111 0111         1    1     1   1  -2.5
   -4:  -  0201  C2  1  1111 1111   1111 0111         1    1     1   1  -2.0
   -3:  F  0201  C2  1  1111 1111   1111 0111         1    1     1   1  -1.5
   -2:  -  0202  B4  1  1111 1111   1111 0111         1    1     1   1  -1.0
   -1:  F  0203  D2  1  1111 1111   1111 0111         1    1     1   1  -0.5
    0:  -  0204  B4  1  1111 1111   1110 0111         1    1     1   1   0.0

  PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 31. Screen After the TDF or <F3> Command

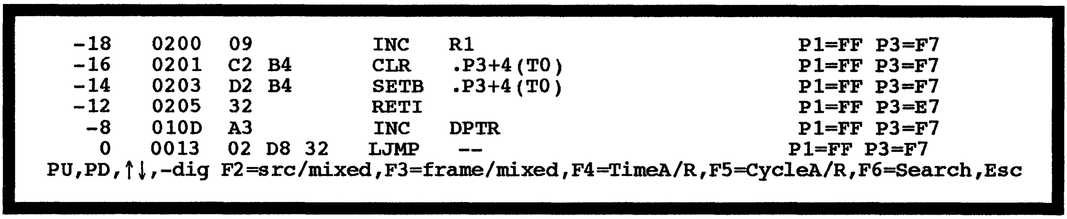
The trace buffer can hold many more frames than will fit on the screen at any one time. The line at the bottom of the screen in Figure 30 and Figure 31 identifies the keys you may use to scroll through the buffer or change the display modes. The numbers at the left side of the screen give the frame numbers: pre-trigger frames have negative numbers, the trigger frame is number zero, and the post-trigger frames have positive numbers.

In Figure 31, the column labeled ***FWR*** tells whether the frame was the first opcode fetch (F), a memory write (W), or a memory read (R). The memory writes or reads are for **XDATA** memory only. The column titles ***ADDR*** and ***DAT*** stand for Address and Data. ***INT*** stands for interrupt Level (which is 0 for the Main Program, 1 for a Level 1 interrupt, etc.). The entries under the ***P1*** and ***P3*** columns give the states of the ports P1 and P3, expressed in binary form. The entries under the columns ***SY1, SY0, E1, E0*** give the states of the external signals that are available on the POD boards (see pod board descriptions). For more information about the format of the frame data please refer to the section on "How the Advanced Trace Works" in the Orientation Chapter.

Press **ESC**. Now try the **TDP** command (Trace Display with Ports), which displays the trace buffer contents (like the TD command) as well as Ports P1 and P3 (in hexadecimal).

```
*TDP
```

Now your screen should look something like that shown in Figure 32.



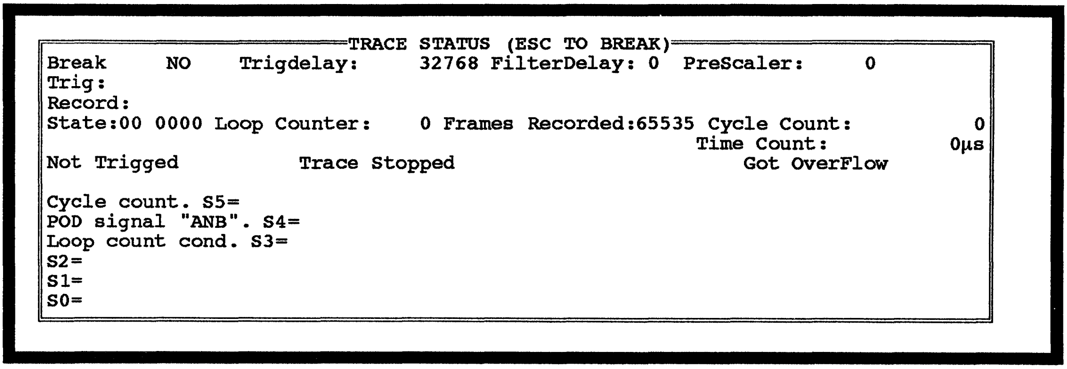
```
   -18    0200  09           INC    R1                            P1=FF P3=F7
   -16    0201  C2 B4        CLR    .P3+4(T0)                     P1=FF P3=F7
   -14    0203  D2 B4        SETB   .P3+4(T0)                     P1=FF P3=F7
   -12    0205  32           RETI                                 P1=FF P3=E7
    -8    010D  A3           INC    DPTR                          P1=FF P3=F7
     0    0013  02 D8 32    LJMP    --                           P1=FF P3=F7
 PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 32. Screen After the TDP Command

Stopping Trace Only After you have started the emulation you can halt the trace without stopping emulation by entering the **TE** command (Trace End); if you just start typing the command the prompt will appear.

```
* res chi
* go

> TE
```

Note that the prompt has now changed to a different character. The ***TRACE STOPPED*** message now appears inside the Trace Status Window (Figure 33), and the emulation LED on the POD board continues to be lit (which indicates that the emulation is still active). At this point you could restart the trace by entering the **TB** command (Trace Begin).



```
                          TRACE STATUS (ESC TO BREAK)
Break     NO    Trigdelay:     32768 FilterDelay: 0  PreScaler:      0
Trig:
Record:
State:00 0000 Loop Counter:    0 Frames Recorded:65535 Cycle Count:          0
                                                       Time Count:        0µs
Not Trigged          Trace Stopped                       Got OverFlow

Cycle count. S5=
POD signal "ANB". S4=
Loop count cond. S3=
S2=
S1=
S0=
```

Figure 33. Screen After the TE Command

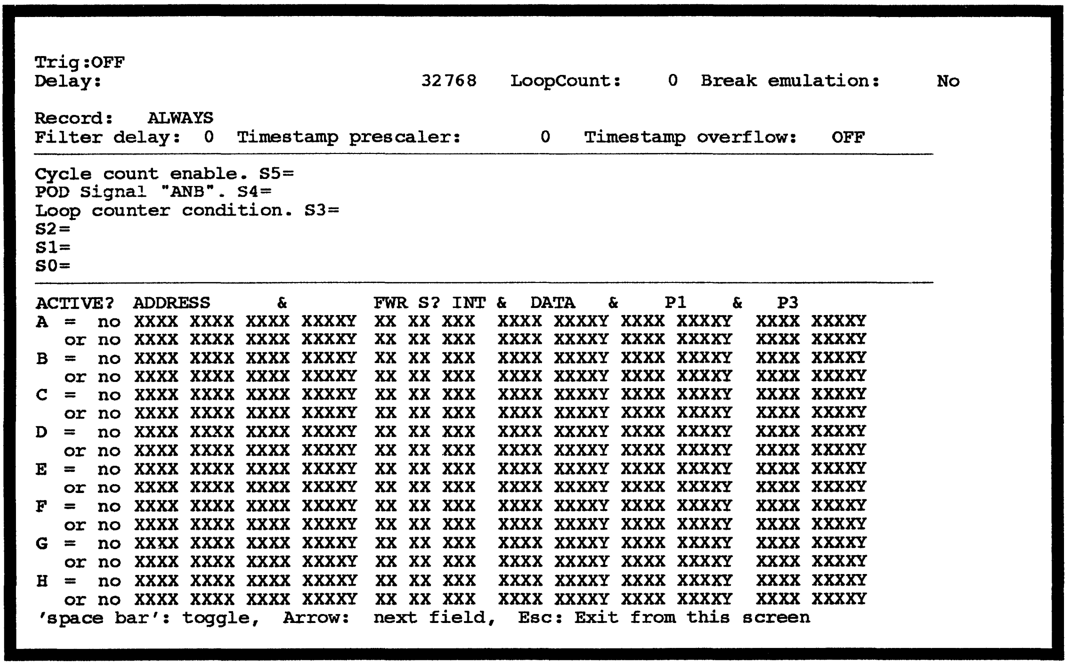
So far you have seen how the trace always starts automatically when emulation is started, and we have "taken snapshots" of what happened just before the ESC key was pressed or before you entered the TE command. With the **Advanced Trace** board, the trace buffer will always display the last 64k frames, (or 256k frames if you have the 256k trace board).

Next we'll discuss how to stop the trace in a more controlled way. You will be working with the Trace Setup Window, which is accessed with the **TS** command (Trace Setup). Press the ESC key and enter the following:

```
*TS
```

Doing this will bring up the TS window as shown in Figure 34.

(Since the trace setup screen is longer than 25 lines you will not see all of Figure 34 on a single screen.)



```
Trig:OFF
Delay:                              32768   LoopCount:    0  Break emulation:     No

Record:   ALWAYS
Filter delay:  0  Timestamp prescaler:       0   Timestamp overflow:   OFF
__________________________________________________________________________________
Cycle count enable. S5=
POD Signal "ANB". S4=
Loop counter condition. S3=
S2=
S1=
S0=
__________________________________________________________________________________
ACTIVE? ADDRESS       &         FWR S? INT &  DATA    &     P1     &    P3
A  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
B  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
C  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
D  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
E  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
F  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
G  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
H  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
'space bar': toggle,  Arrow:  next field,  Esc: Exit from this screen
```

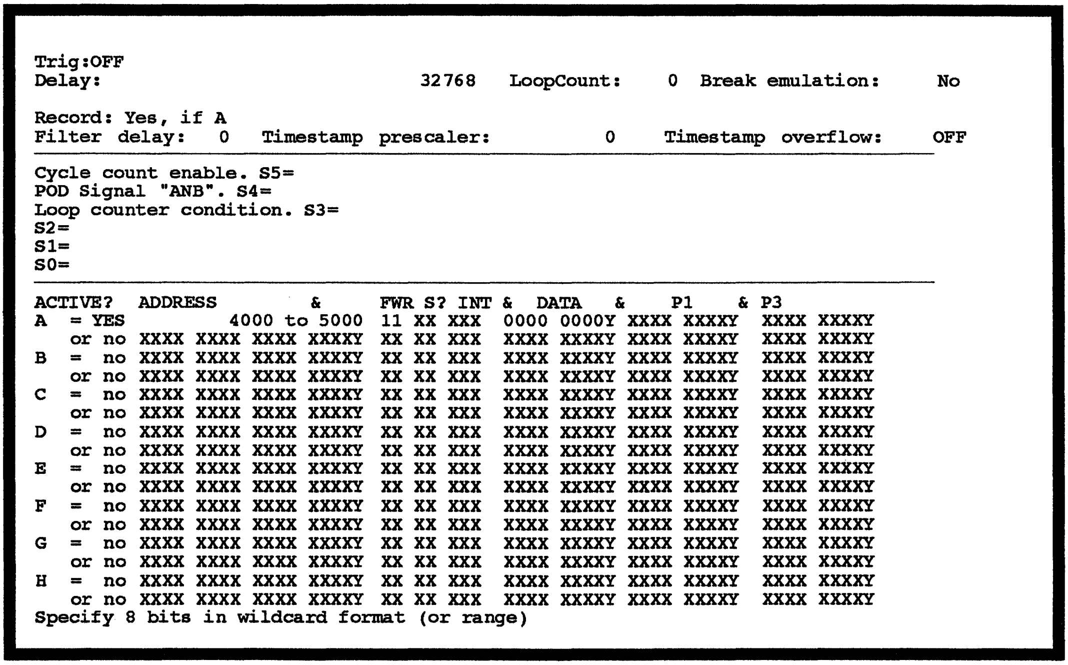
Figure 34. Screen After the TS Command

The following examples will demonstrate how changes to the trace setup will have an effect on the data stored in the trace buffer.

Setup Example 1

By performing the following steps you create a trace setup that will yield a trace capture of *ONLY* the writes to address range 4000h to 5000h with the data of 00h.

Starting with the screen shown in Figure 34, use the cursor arrow keys to move the cursor to the RECORD field and press the SPACE bar to select "Yes, if" and press *ENTER* and enter an 'A'. This will specify that the Qualifying register 'A' is to be used as the filter mechanism. Keep pressing *ENTER* until you come to the A = no in the ACTIVE column. Press the space bar to change this field to a 'yes' and press return to go to the address column and enter the address range **4000 to 5000**. Move your cursor into the 'FWR S? INT' field, press the insert key, and type '**11**' under the FWR column then press *ENTER*. Move your cursor into the DATA column, press <insert> (to turn off overtype mode), and type '**0000 0000Y**' then press enter. Your screen should look something like what is shown in Figure 35.



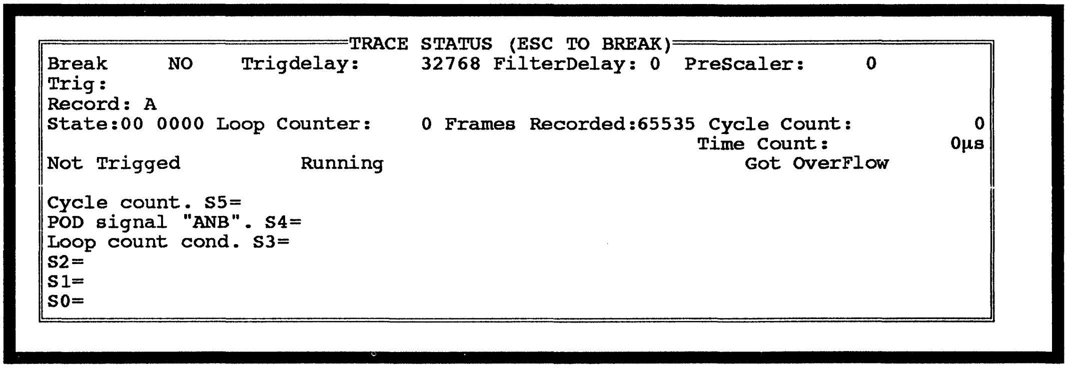
```
Trig:OFF
Delay:                                  32768   LoopCount:    0  Break emulation:     No

Record: Yes, if A
Filter delay:  0   Timestamp prescaler:          0     Timestamp overflow:     OFF
------------------------------------------------------------------------------------
Cycle count enable. S5=
POD Signal "ANB". S4=
Loop counter condition. S3=
S2=
S1=
S0=
------------------------------------------------------------------------------------
ACTIVE?  ADDRESS          &      FWR S? INT &  DATA   &      P1     & P3
A   = YES        4000 to 5000  11 XX XXX   0000 0000Y XXXX XXXXY   XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
B   = no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
C   = no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
D   = no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
E   = no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
F   = no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
G   = no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
H   = no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
Specify 8 bits in wildcard format (or range)
```

Figure 35. Trace Setup Example 1

Press ESC and restart the emulation and trace by entering the following commands:

```
* res chi
* g
```

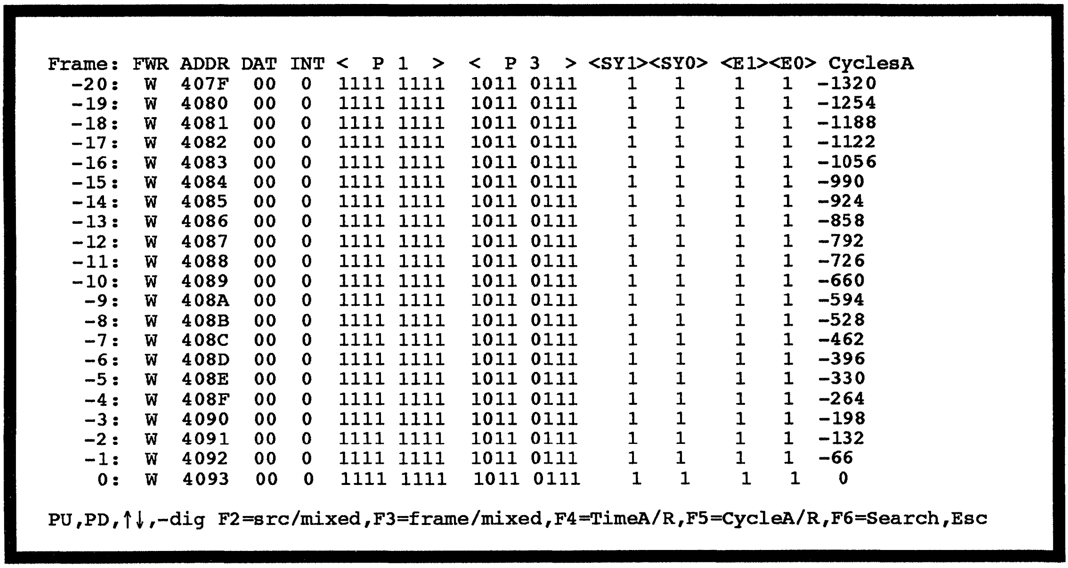
After a short while, the trace status window at the top of the screen should look something like Figure 36.



```
═══════════════════════TRACE STATUS (ESC TO BREAK)══════════════════════════
Break      NO     Trigdelay:     32768 FilterDelay: 0  PreScaler:      0
Trig:
Record: A
State:00 0000 Loop Counter:     0 Frames Recorded:65535 Cycle Count:         0
                                                    Time Count:          0µs
Not Trigged          Running                             Got OverFlow

Cycle count. S5=
POD signal "ANB". S4=
Loop count cond. S3=
S2=
S1=
S0=
```

Figure 36. Trace Window After Starting Emulation

Now type *TE* (trace end) and then *TDF* (Trace Display Frames) to display the trace buffer in the frames mode. Then the frames mode must be used because there is no actual code execution being captured in the trace buffer. Your screen should look something like Figure 37 after the TDF command.



```
Frame: FWR ADDR DAT INT <  P 1  >  <  P 3  > <SY1><SY0> <E1><E0> CyclesA
  -20:  W  407F  00  0  1111 1111  1011 0111    1    1     1   1  -1320
  -19:  W  4080  00  0  1111 1111  1011 0111    1    1     1   1  -1254
  -18:  W  4081  00  0  1111 1111  1011 0111    1    1     1   1  -1188
  -17:  W  4082  00  0  1111 1111  1011 0111    1    1     1   1  -1122
  -16:  W  4083  00  0  1111 1111  1011 0111    1    1     1   1  -1056
  -15:  W  4084  00  0  1111 1111  1011 0111    1    1     1   1  -990
  -14:  W  4085  00  0  1111 1111  1011 0111    1    1     1   1  -924
  -13:  W  4086  00  0  1111 1111  1011 0111    1    1     1   1  -858
  -12:  W  4087  00  0  1111 1111  1011 0111    1    1     1   1  -792
  -11:  W  4088  00  0  1111 1111  1011 0111    1    1     1   1  -726
  -10:  W  4089  00  0  1111 1111  1011 0111    1    1     1   1  -660
   -9:  W  408A  00  0  1111 1111  1011 0111    1    1     1   1  -594
   -8:  W  408B  00  0  1111 1111  1011 0111    1    1     1   1  -528
   -7:  W  408C  00  0  1111 1111  1011 0111    1    1     1   1  -462
   -6:  W  408D  00  0  1111 1111  1011 0111    1    1     1   1  -396
   -5:  W  408E  00  0  1111 1111  1011 0111    1    1     1   1  -330
   -4:  W  408F  00  0  1111 1111  1011 0111    1    1     1   1  -264
   -3:  W  4090  00  0  1111 1111  1011 0111    1    1     1   1  -198
   -2:  W  4091  00  0  1111 1111  1011 0111    1    1     1   1  -132
   -1:  W  4092  00  0  1111 1111  1011 0111    1    1     1   1  -66
    0:  W  4093  00  0  1111 1111  1011 0111    1    1     1   1   0

PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 37. TDF Display For Example 1

Afterwards press <ESC> <ESC> to get back to the (*) prompt.

Setup Example 2

In this example, you will change the trace setup so that the trace will trigger on a code fetch at the address 205. When the 'A' condition becomes true, the trace delay counter will be decremented until it reaches 0. Then the trace will stop automatically. If necessary, you could also use a range of addresses as a qualifier.

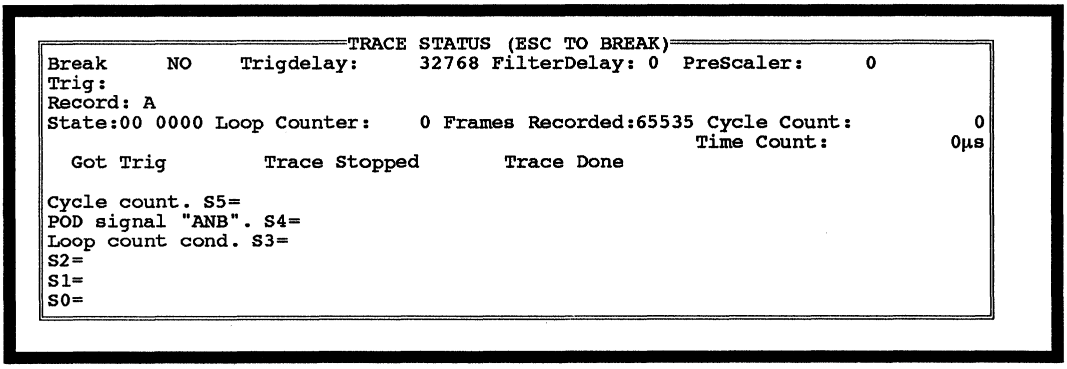
The steps taken to set up this new trace are much like those in Example 1, with the following changes:

■ TRIG =	Yes, if a
■ Record =	ALWAYS
■ A =	YES 205
■ FWR S? INT =	01 XX XXXY
■ DATA =	XXXX XXXXY

Restart the emulation with the following commands:

```
* res chi
* g
```

After a short time, the trace window should look something like Figure 38.



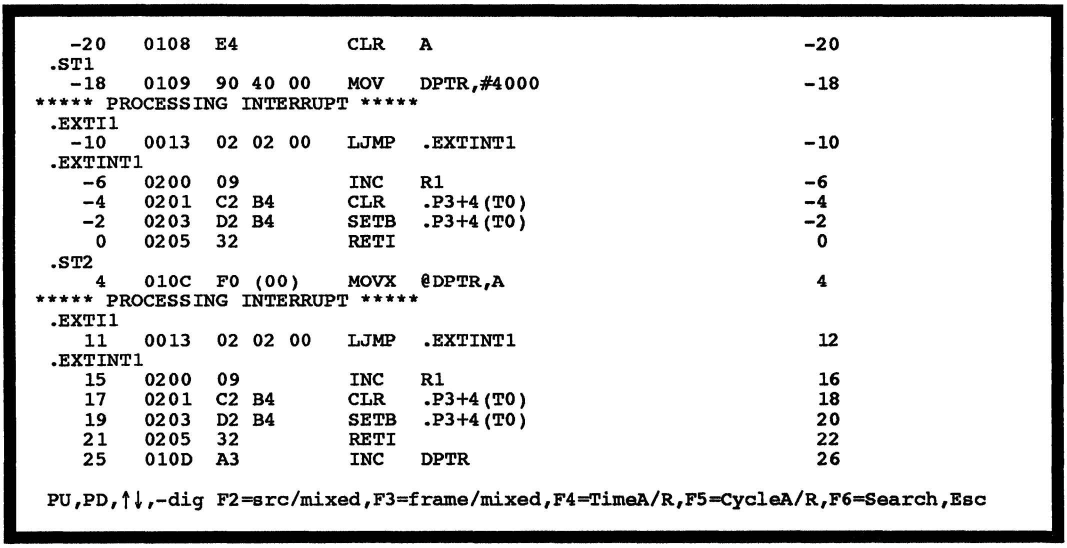
```
=================TRACE STATUS (ESC TO BREAK)==================
Break     NO    Trigdelay:     32768 FilterDelay: 0  PreScaler:      0
Trig:
Record: A
State:00 0000 Loop Counter:    0 Frames Recorded:65535 Cycle Count:          0
                                                   Time Count:          0µs
  Got Trig          Trace Stopped        Trace Done

Cycle count. S5=
POD signal "ANB". S4=
Loop count cond. S3=
S2=
S1=
S0=
```

Figure 38. Trace Window After Trace is Stopped

Now use the TD command to display the trace buffer:

> TD

Your screen should look something like what is shown in Figure 39.

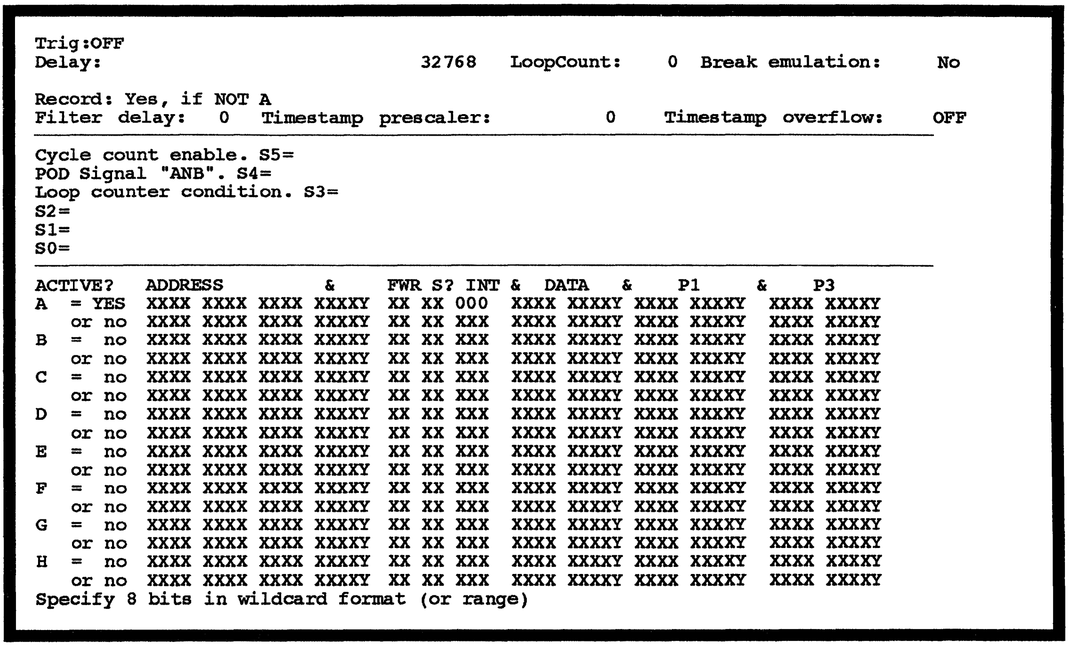


```
  -20   0108  E4          CLR   A                                    -20
 .ST1
  -18   0109  90 40 00    MOV   DPTR,#4000                           -18
***** PROCESSING INTERRUPT *****
 .EXTI1
  -10   0013  02 02 00    LJMP  .EXTINT1                             -10
 .EXTINT1
   -6   0200  09          INC   R1                                   -6
   -4   0201  C2 B4       CLR   .P3+4(T0)                            -4
   -2   0203  D2 B4       SETB  .P3+4(T0)                            -2
    0   0205  32          RETI                                       0
 .ST2
    4   010C  F0 (00)     MOVX  @DPTR,A                              4
***** PROCESSING INTERRUPT *****
 .EXTI1
   11   0013  02 02 00    LJMP  .EXTINT1                             12
 .EXTINT1
   15   0200  09          INC   R1                                   16
   17   0201  C2 B4       CLR   .P3+4(T0)                            18
   19   0203  D2 B4       SETB  .P3+4(T0)                            20
   21   0205  32          RETI                                       22
   25   010D  A3          INC   DPTR                                 26

 PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 39. Trace Display Example 2

Note the PRE and POST trigger frame numbers along the left side of the screen.

Setup Example 3 In this example you will set up the trace to capture only the code that is *NOT* part of the main program. In other words you will capture only the interrupts that are being processed. Please use the trace setup screen (TS) to match Figure 40.



```
Trig:OFF
Delay:                              32768   LoopCount:    0  Break emulation:     No

Record: Yes, if NOT A
Filter delay:  0  Timestamp prescaler:          0    Timestamp overflow:     OFF
------------------------------------------------------------------------------------
Cycle count enable. S5=
POD Signal "ANB". S4=
Loop counter condition. S3=
S2=
S1=
S0=
------------------------------------------------------------------------------------
ACTIVE?   ADDRESS            &     FWR S? INT &  DATA   &     P1     &     P3
A  = YES  XXXX XXXX XXXX XXXXY  XX XX 000  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
B  =  no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
C  =  no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
D  =  no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
E  =  no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
F  =  no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
G  =  no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
H  =  no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
Specify 8 bits in wildcard format (or range)
```

Figure 40. Trace Setup Example 3

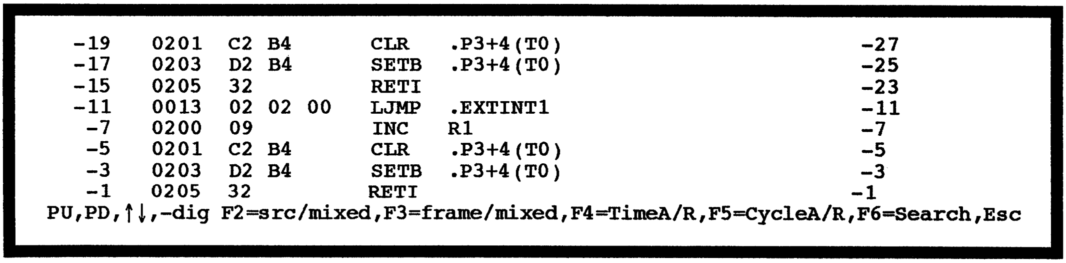
Press *ENTER* and then *ESC*. Now restart the emulation and trace with the following commands:

```
* res chi
* g
```

To see the effects of this setup, stop the trace (TE), and display the trace (TD).

```
> TE
> TD
```

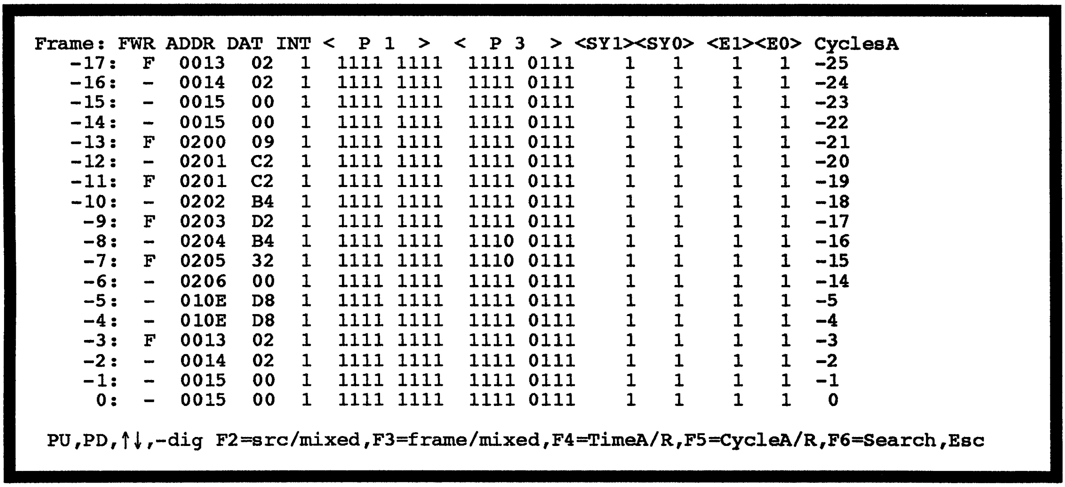
Now your screen should look something like Figure 41.



```
   -19    0201  C2 B4       CLR    .P3+4(T0)                       -27
   -17    0203  D2 B4       SETB   .P3+4(T0)                       -25
   -15    0205  32          RETI                                   -23
   -11    0013  02 02 00    LJMP   .EXTINT1                        -11
    -7    0200  09          INC    R1                              -7
    -5    0201  C2 B4       CLR    .P3+4(T0)                       -5
    -3    0203  D2 B4       SETB   .P3+4(T0)                       -3
    -1    0205  32          RETI                                   -1
PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 41. Trace Display Example 3

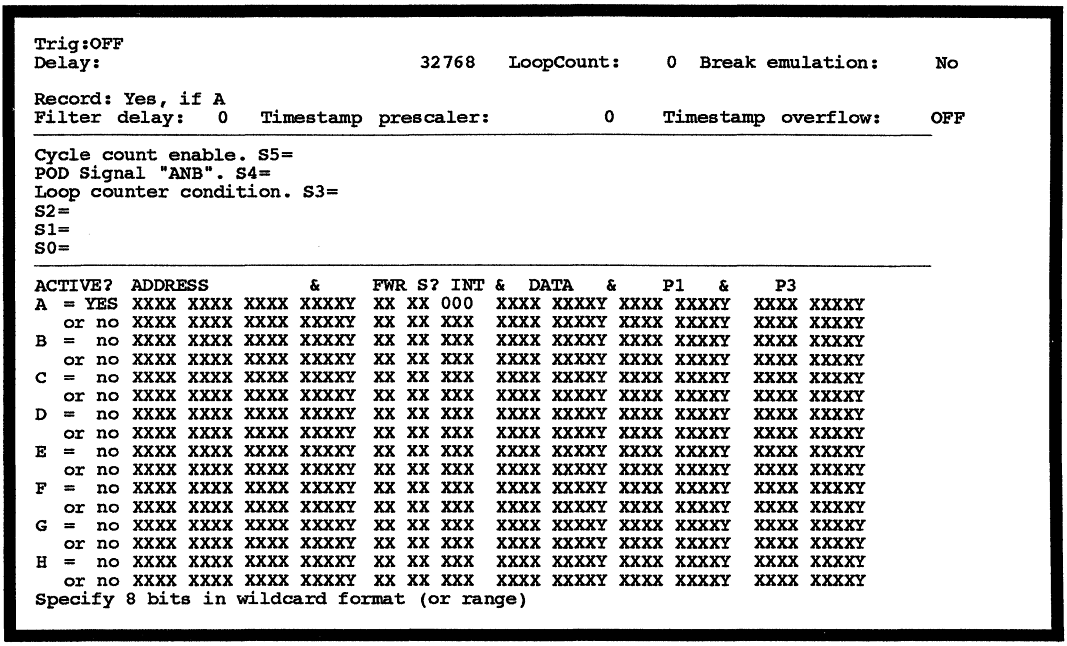
Now press **F3** and display the frames mode. Please notice that the INT column never shows a 0 (which confirms the fact that this is only the interrupt program). See Figure 42.



```
Frame: FWR ADDR DAT INT <  P 1   >  <  P 3   > <SY1><SY0> <E1><E0> CyclesA
   -17:  F  0013  02  1  1111 1111  1111 0111    1    1     1   1  -25
   -16:  -  0014  02  1  1111 1111  1111 0111    1    1     1   1  -24
   -15:  -  0015  00  1  1111 1111  1111 0111    1    1     1   1  -23
   -14:  -  0015  00  1  1111 1111  1111 0111    1    1     1   1  -22
   -13:  F  0200  09  1  1111 1111  1111 0111    1    1     1   1  -21
   -12:  -  0201  C2  1  1111 1111  1111 0111    1    1     1   1  -20
   -11:  F  0201  C2  1  1111 1111  1111 0111    1    1     1   1  -19
   -10:  -  0202  B4  1  1111 1111  1111 0111    1    1     1   1  -18
    -9:  F  0203  D2  1  1111 1111  1111 0111    1    1     1   1  -17
    -8:  -  0204  B4  1  1111 1111  1110 0111    1    1     1   1  -16
    -7:  F  0205  32  1  1111 1111  1110 0111    1    1     1   1  -15
    -6:  -  0206  00  1  1111 1111  1111 0111    1    1     1   1  -14
    -5:  -  010E  D8  1  1111 1111  1111 0111    1    1     1   1  -5
    -4:  -  010E  D8  1  1111 1111  1111 0111    1    1     1   1  -4
    -3:  F  0013  02  1  1111 1111  1111 0111    1    1     1   1  -3
    -2:  -  0014  02  1  1111 1111  1111 0111    1    1     1   1  -2
    -1:  -  0015  00  1  1111 1111  1111 0111    1    1     1   1  -1
     0:  -  0015  00  1  1111 1111  1111 0111    1    1     1   1   0

 PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 42. Trace Display Example 3 <F3>

Setup Example 4 To complement what you just did in the previous example, set up the trace to filter out <u>ALL</u> the interrupt processing. Press ESC to stop emulation. Then us the TS command to set up the new trace parameters as shown in Figure 43.



```
Trig:OFF
Delay:                              32768   LoopCount:    0  Break emulation:     No

Record: Yes, if A
Filter delay:  0   Timestamp prescaler:          0     Timestamp overflow:    OFF
-----------------------------------------------------------------------------------
Cycle count enable. S5=
POD Signal "ANB". S4=
Loop counter condition. S3=
S2=
S1=
S0=
-----------------------------------------------------------------------------------
ACTIVE? ADDRESS          &     FWR S? INT &  DATA   &    P1   &    P3
A  = YES XXXX XXXX XXXX XXXXY  XX XX 000  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
B  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
C  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
D  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
E  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
F  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
G  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
H  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no  XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
Specify 8 bits in wildcard format (or range)
```

Figure 43. Trace Setup Example 4

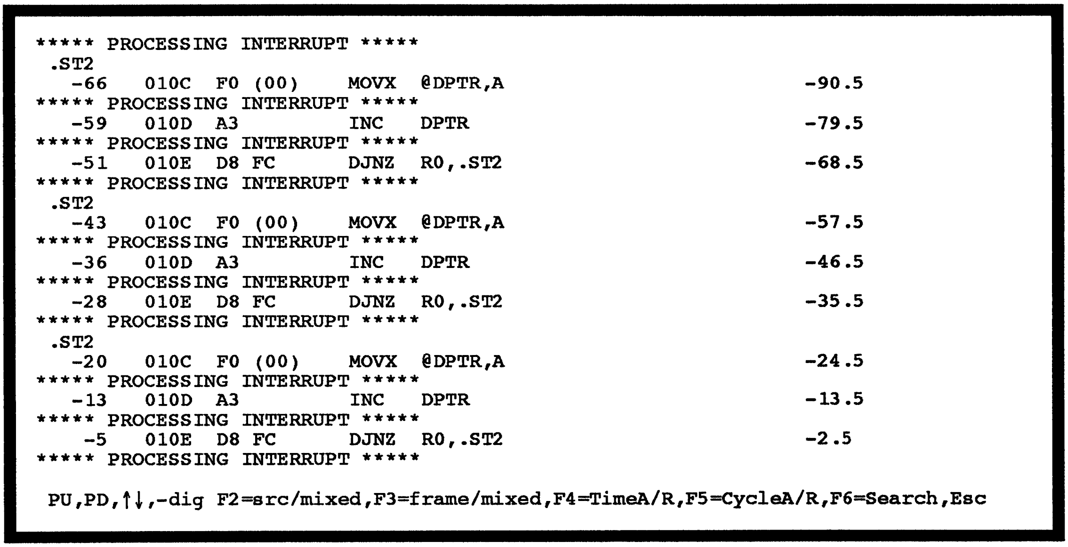
Now restart the emulation and the trace with the following commands:

```
* res chi
* g
```

To see the effects of this setup, stop the trace (TE), and display the trace buffer (TD):

```
> TE
> TD
```

After entering the trace display mode, use the PgUp then PgDn keys and now your screen should look something like Figure 44.



```
***** PROCESSING INTERRUPT *****
 .ST2
   -66    010C  F0 (00)     MOVX  @DPTR,A                                     -90.5
***** PROCESSING INTERRUPT *****
   -59    010D  A3          INC   DPTR                                        -79.5
***** PROCESSING INTERRUPT *****
   -51    010E  D8 FC       DJNZ  R0,.ST2                                     -68.5
***** PROCESSING INTERRUPT *****
 .ST2
   -43    010C  F0 (00)     MOVX  @DPTR,A                                     -57.5
***** PROCESSING INTERRUPT *****
   -36    010D  A3          INC   DPTR                                        -46.5
***** PROCESSING INTERRUPT *****
   -28    010E  D8 FC       DJNZ  R0,.ST2                                     -35.5
***** PROCESSING INTERRUPT *****
 .ST2
   -20    010C  F0 (00)     MOVX  @DPTR,A                                     -24.5
***** PROCESSING INTERRUPT *****
   -13    010D  A3          INC   DPTR                                        -13.5
***** PROCESSING INTERRUPT *****
    -5    010E  D8 FC       DJNZ  R0,.ST2                                     -2.5
***** PROCESSING INTERRUPT *****

 PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 44. Trace Display Example 4

Trace Breakpoint Register

The trace board is provided with a feature to break emulation with the BREAK EMULATION register. This register can be set with more possibilities than the standard 16k or 4k trace boards. Using one of the following examples, you can set up your trace to break emulation.

- On trigger = When the trace is triggered
- When done = trace full (Delay decrements to 0). *Requires Trigger*
- On S2 = When the condition on S2 comes true

ADVANCED TRACE FEATURES

This segment's goal is to expand upon the features of the *Advanced Trace Board.* It covers the following fields of the trace setup (TS):

- Trig
- Break emulation
- Record
- Filter delay
- Timestamp prescaler
- Timestamp overflow
- Special registers S0 - S5

For this section of the tutorial we have included a program on the distribution disk that was installed at the time the EMUL51-PC software was installed. This program is named ***ADV_TRA***. This portion of the tutorial will take approximently 15-20 minutes.

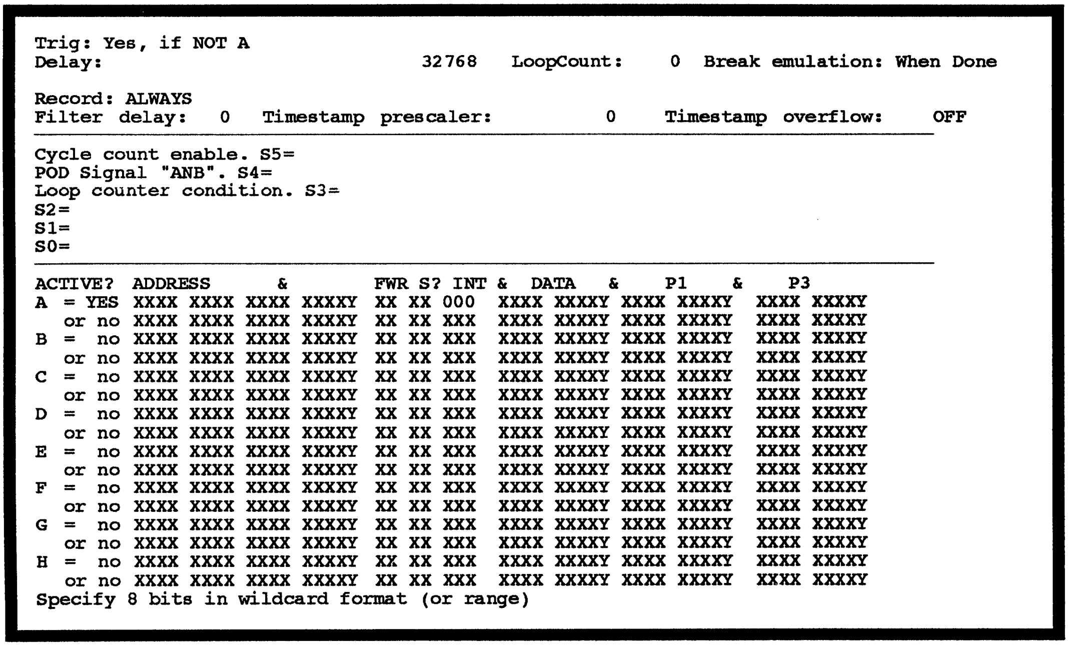
Refer to Appendix A for the disassembly of the code for ADV_TRA.

Advanced Setup Example 1

Please follow the commands below to clear out code memory, remove <u>ALL</u> old symbols from the previous program from memory, and load the new program.

```
* fill 0 to ffff 0 cb
* rem sym
* load adv_tra
* tsd = off       ; since we are not concerned about the time stamp at
                    this time.
* cw              ; enable code window
```

The following trace setup example shows you how the trace can be triggered by the first interrupt that occurs after reset. Also, this setup will break emulation when the trace buffer is full (the *DELAY* register has decremented to 0). See Figure 45 for trace setup.



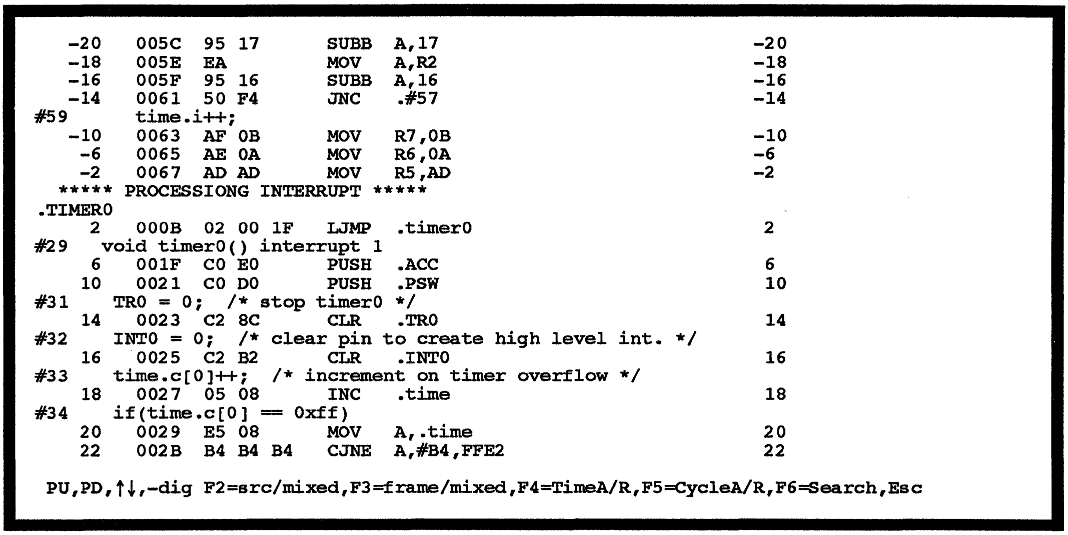
```
Trig: Yes, if NOT A
Delay:                              32768   LoopCount:     0  Break emulation: When Done

Record: ALWAYS
Filter delay:   0   Timestamp prescaler:          0    Timestamp overflow:     OFF
________________________________________________________________________________
Cycle count enable. S5=
POD Signal "ANB". S4=
Loop counter condition. S3=
S2=
S1=
S0=
________________________________________________________________________________
ACTIVE? ADDRESS      &         FWR S? INT &  DATA   &     P1    &     P3
A  = YES XXXX XXXX XXXX XXXXY  XX XX 000  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
B  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
C  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
D  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
E  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
F  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
G  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
H  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
Specify 8 bits in wildcard format (or range)
```

Figure 45. Advanced Trace Setup Example 1

Now start the emulation and trace with the following commands:

```
* res chi
* g
```

After the emulation breaks, which should happen quickly, display the trace buffer with the TD command. Your screen should look something like Figure 46.



```
   -20    005C  95 17      SUBB  A,17                                  -20
   -18    005E  EA         MOV   A,R2                                  -18
   -16    005F  95 16      SUBB  A,16                                  -16
   -14    0061  50 F4      JNC   .#57                                  -14
#59       time.i++;
   -10    0063  AF 0B      MOV   R7,0B                                 -10
    -6    0065  AE 0A      MOV   R6,0A                                 -6
    -2    0067  AD AD      MOV   R5,AD                                 -2
  ***** PROCESSIONG INTERRUPT *****
.TIMER0
     2    000B  02 00 1F   LJMP  .timer0                               2
#29     void timer0() interrupt 1
     6    001F  C0 E0      PUSH  .ACC                                  6
    10    0021  C0 D0      PUSH  .PSW                                  10
#31       TR0 = 0;  /* stop timer0 */
    14    0023  C2 8C      CLR   .TR0                                  14
#32       INT0 = 0;  /* clear pin to create high level int. */
    16    0025  C2 B2      CLR   .INT0                                 16
#33       time.c[0]++;  /* increment on timer overflow */
    18    0027  05 08      INC   .time                                 18
#34       if(time.c[0] == 0xff)
    20    0029  E5 08      MOV   A,.time                               20
    22    002B  B4 B4 B4   CJNE  A,#B4,FFE2                            22

 PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

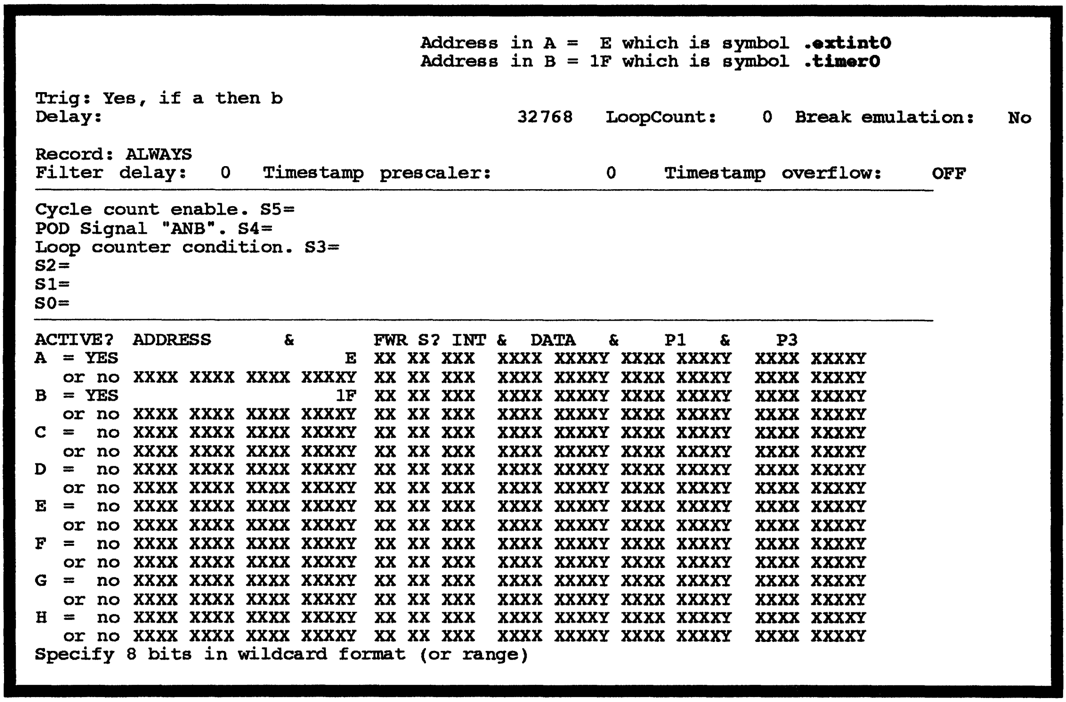
Figure 46. Trace Display for Advanced Trace Example 1

As you can see, the frames on the left side show a pre-trigger frame at -3 and then a post-trigger frame at 1 just after line #29. If you use the <F3> key to switch to frames mode you will be able to see the actual frame 0. Observe that the actual trigger in the trace buffer occured when the INT column of the trace went from 0 to 1.

Use the <F6> key to active the search window, to be covered later, and then press <Ctrl+PgDn>. Note that hte last frame is the instruction **LCALL 103E.** If you go back to the main display you will see that in the code window; the address 103E is the next address that will be executed (highlighted bar is on that address in the code window).

Comments Having a trace setup with the break enabled for "When Done" a user could use the trace listing commands (TDL or TDFL) to save the trace data to a disk file and then continue to execute from where the program was stopped.

Advanced Setup Example 2 In this example we will show how to set a trigger on an event that follows another event. We have written this program so that a lower priority interrupt is halted by a higher priority interrupt. Refer to Figure 47 for the Trace Setup.



```
                                  Address in A =  E which is symbol .extint0
                                  Address in B = 1F which is symbol .timer0

Trig: Yes, if a then b
Delay:                                    32768   LoopCount:    0  Break emulation:   No

Record: ALWAYS
Filter delay:  0  Timestamp prescaler:         0    Timestamp overflow:    OFF
----------------------------------------------------------------------------------
Cycle count enable. S5=
POD Signal "ANB". S4=
Loop counter condition. S3=
S2=
S1=
S0=
----------------------------------------------------------------------------------
ACTIVE? ADDRESS      &        FWR S? INT &  DATA   &     P1   &      P3
A  = YES                    E  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
B  = YES                   1F  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
C  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
D  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
E  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
F  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
G  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
H  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
  or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
Specify 8 bits in wildcard format (or range)
```

Figure 47. Advanced Trace Setup Example 2

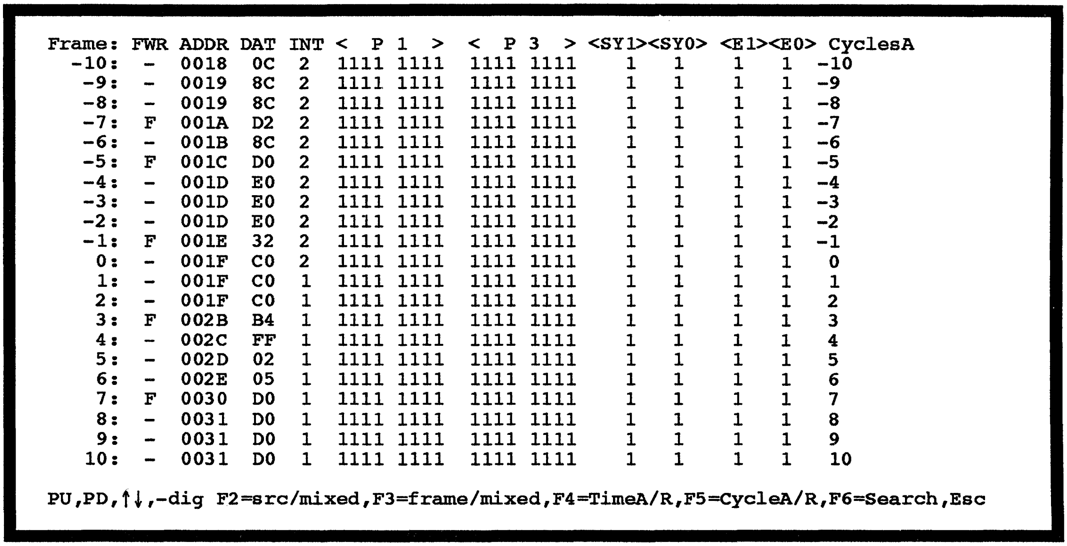
Now we will restart the emulation and tracing with the following commands:

```
* res chi
* g
```

So that you can see the INT column of the trace display we will use the TDF command to display the trace buffer.

```
> TDF -10        ; start display at frame -10
```

Your screen should look something like what is shown in Figure 48.



```
Frame: FWR ADDR DAT INT <  P 1  >   <  P 3  > <SY1><SY0> <E1><E0> CyclesA
  -10:  -  0018  0C  2  1111 1111  1111 1111   1    1     1   1  -10
   -9:  -  0019  8C  2  1111 1111  1111 1111   1    1     1   1  -9
   -8:  -  0019  8C  2  1111 1111  1111 1111   1    1     1   1  -8
   -7:  F  001A  D2  2  1111 1111  1111 1111   1    1     1   1  -7
   -6:  -  001B  8C  2  1111 1111  1111 1111   1    1     1   1  -6
   -5:  F  001C  D0  2  1111 1111  1111 1111   1    1     1   1  -5
   -4:  -  001D  E0  2  1111 1111  1111 1111   1    1     1   1  -4
   -3:  -  001D  E0  2  1111 1111  1111 1111   1    1     1   1  -3
   -2:  -  001D  E0  2  1111 1111  1111 1111   1    1     1   1  -2
   -1:  F  001E  32  2  1111 1111  1111 1111   1    1     1   1  -1
    0:  -  001F  C0  2  1111 1111  1111 1111   1    1     1   1   0
    1:  -  001F  C0  1  1111 1111  1111 1111   1    1     1   1   1
    2:  -  001F  C0  1  1111 1111  1111 1111   1    1     1   1   2
    3:  F  002B  B4  1  1111 1111  1111 1111   1    1     1   1   3
    4:  -  002C  FF  1  1111 1111  1111 1111   1    1     1   1   4
    5:  -  002D  02  1  1111 1111  1111 1111   1    1     1   1   5
    6:  -  002E  05  1  1111 1111  1111 1111   1    1     1   1   6
    7:  F  0030  D0  1  1111 1111  1111 1111   1    1     1   1   7
    8:  -  0031  D0  1  1111 1111  1111 1111   1    1     1   1   8
    9:  -  0031  D0  1  1111 1111  1111 1111   1    1     1   1   9
   10:  -  0031  D0  1  1111 1111  1111 1111   1    1     1   1   10

PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 48. Trace Display Frames 'TDF' For Example 2

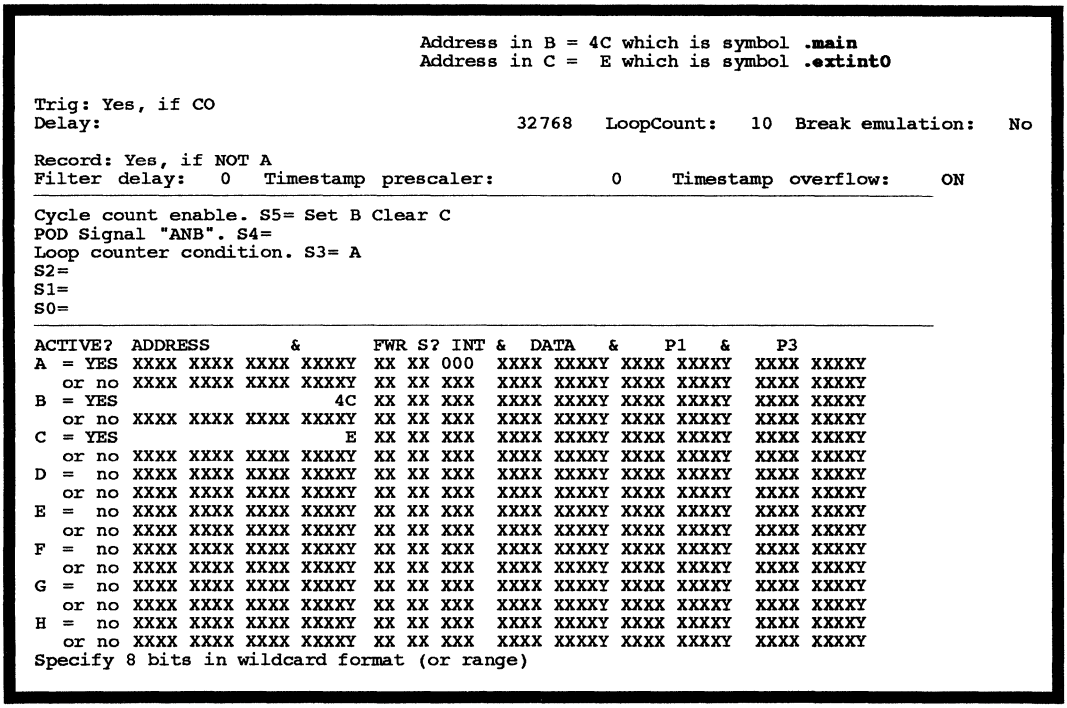
In this example, the condition in the 'A' qualifier does not come true until after the 'B' qualifier. Therefore the trigger occurs on the second time that the 'B' condition comes true. As explained earlier, the second portion after the "THEN" condition needs to come true sometime after the first condtion for the "THEN" to be true.

When using the "THEN" in an equation for the trigger or filter, the special state machine bits S0-S2 cannot be utilized because they are internally used to generate states to implement "THEN".

Advanced Setup Example 3 In this example we will use several of the functions in combination to show you how they will affect the trace captures. The trace setup for this example is pictured in Figure 49. We will use the following features:

- Trigger on LoopCount = CO
- S5 (timecount condition)
- S3 (loopcount condition)
- Record overflow of timestamp counter
- A Filtering condition

Now type ESC to exit the trace display, then TS to setup a new trace condition.



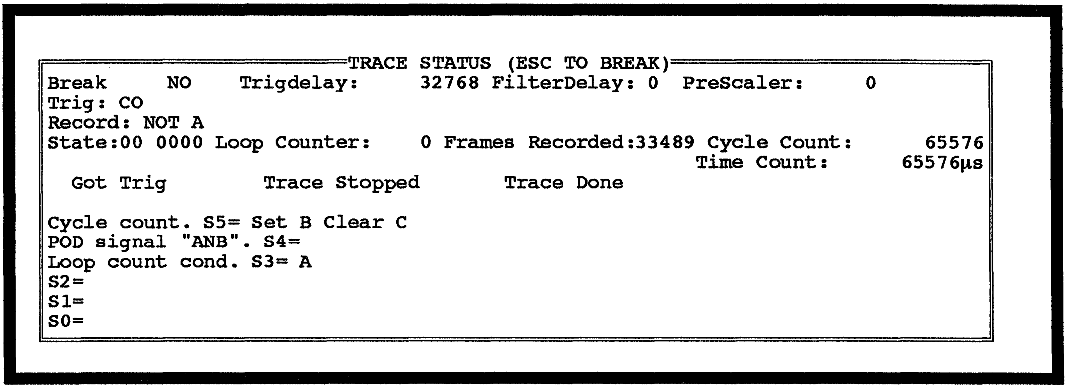
```
                                        Address in B = 4C which is symbol .main
                                        Address in C =  E which is symbol .extint0

Trig: Yes, if CO
Delay:                                            32768  LoopCount:  10  Break emulation:  No

Record: Yes, if NOT A
Filter delay:  0  Timestamp prescaler:           0    Timestamp overflow:     ON
------------------------------------------------------------------------------------
Cycle count enable. S5= Set B Clear C
POD Signal "ANB". S4=
Loop counter condition. S3= A
S2=
S1=
S0=
------------------------------------------------------------------------------------
ACTIVE? ADDRESS          &       FWR S? INT &  DATA     &     P1    &     P3
A = YES XXXX XXXX XXXX XXXXY  XX XX 000  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
B = YES                   4C  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
C = YES                    E  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
D =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
E =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
F =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
G =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
H =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
Specify 8 bits in wildcard format (or range)
```

Figure 49. Advanced Trace Setup Example 3

Restart the emulation and trace with the following commands, but first we must break emulation:

```
<esc> /* exit trace setup */
<esc> /* break emulation */

* res chi
* go
* tsd = on        /* no we need accurate timestamp information */
```

Let's look at the trace status window after the trace has stopped and see what is displayed. Refer to Figure 50.



```
=============================TRACE STATUS (ESC TO BREAK)==============================
Break     NO     Trigdelay:      32768 FilterDelay: 0  PreScaler:      0
Trig: CO
Record: NOT A
State:00 0000 Loop Counter:     0 Frames Recorded:33489 Cycle Count:       65576
                                                         Time Count:     65576µs
  Got Trig          Trace Stopped         Trace Done

Cycle count. S5= Set B Clear C
POD signal "ANB". S4=
Loop count cond. S3= A
S2=
S1=
S0=
```

Figure 50. Trace Window After Trace is Stopped

Here is a description of what the trace setup is set up to capture.

- Trigger = 'A' LOOP , where LoopCount = 10. Trigger after the 10th loop of (main program-> interrupt-> main program).
- Record = record only the interrupt routines and the timestamp overflow.
- Display Time Count from the start of the 'C' program [.main] to the second interrupt level. This can be accomplished as it will cause the external interrupt, in this case the 'high priority' interrupt, to halt the 'lower priority' interrupt (timer 0) service routine.

We can see that the Time Count field in the trace status window shows the number 65576 microseconds at 12MHz. The emulator has a command 'XTal' which allows the trace to show the properly scaled time measurement. The XTal command will allow up to 6 decimals places after the decimal point.

The syntax for the *XTAL* command is as follows:

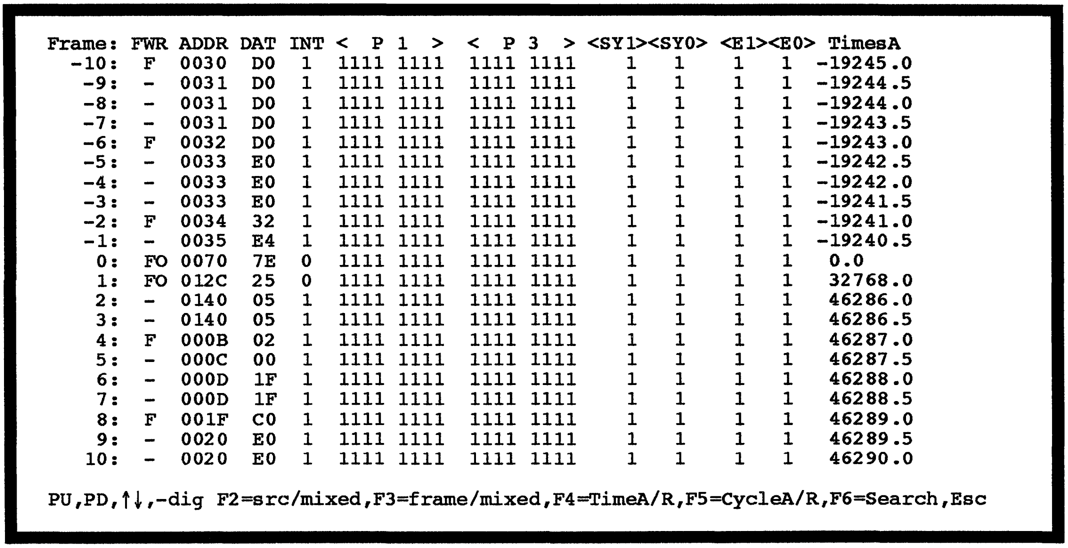
```
* XTal = <some number in MHz>eg. XT = 10.738635
```

You can also set this number from the DOS command line or batch file with a -x option using the same number scheme as the XTal command.

Now display the trace buffer with the TDF command, using the following commands:

```
<esc>          /* break emulation */
* TDF -10      /* start display at frame number -10 */
```

Your screen should look something like Figure 51.



```
Frame: FWR ADDR DAT INT <  P 1  >  <  P 3  > <SY1><SY0> <E1><E0> TimesA
  -10:  F  0030  D0  1  1111 1111  1111 1111   1    1    1   1  -19245.0
   -9:  -  0031  D0  1  1111 1111  1111 1111   1    1    1   1  -19244.5
   -8:  -  0031  D0  1  1111 1111  1111 1111   1    1    1   1  -19244.0
   -7:  -  0031  D0  1  1111 1111  1111 1111   1    1    1   1  -19243.5
   -6:  F  0032  D0  1  1111 1111  1111 1111   1    1    1   1  -19243.0
   -5:  -  0033  E0  1  1111 1111  1111 1111   1    1    1   1  -19242.5
   -4:  -  0033  E0  1  1111 1111  1111 1111   1    1    1   1  -19242.0
   -3:  -  0033  E0  1  1111 1111  1111 1111   1    1    1   1  -19241.5
   -2:  F  0034  32  1  1111 1111  1111 1111   1    1    1   1  -19241.0
   -1:  -  0035  E4  1  1111 1111  1111 1111   1    1    1   1  -19240.5
    0:  FO 0070  7E  0  1111 1111  1111 1111   1    1    1   1   0.0
    1:  FO 012C  25  0  1111 1111  1111 1111   1    1    1   1   32768.0
    2:  -  0140  05  1  1111 1111  1111 1111   1    1    1   1   46286.0
    3:  -  0140  05  1  1111 1111  1111 1111   1    1    1   1   46286.5
    4:  F  000B  02  1  1111 1111  1111 1111   1    1    1   1   46287.0
    5:  -  000C  00  1  1111 1111  1111 1111   1    1    1   1   46287.5
    6:  -  000D  1F  1  1111 1111  1111 1111   1    1    1   1   46288.0
    7:  -  000D  1F  1  1111 1111  1111 1111   1    1    1   1   46288.5
    8:  F  001F  C0  1  1111 1111  1111 1111   1    1    1   1   46289.0
    9:  -  0020  E0  1  1111 1111  1111 1111   1    1    1   1   46289.5
   10:  -  0020  E0  1  1111 1111  1111 1111   1    1    1   1   46290.0

PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 51. Trace Display Frames 'TDF' For Example 3

Notice the INT column and the FWR column. At frame 0 you can see that the

TimeStamp counter overflowed, with the '0' displayed. If the counter had overflowed on an F, W, or R frame, you would have seen an FO, WO, or RO in the FWR column. And since we are displaying the TimeA, (time absolute), you can see the roll-over of the counter to 0.

Comments

RECORD In this setup we have requested the trace to capture only the frames when the condition in **A** is not true. Since the **A** qualifier is set to INT = 000, all frames where INT ≠ 0 were recorded. An exception to this condition is the overflow of the timestamp which forced extra frames to be placed in the trace buffer.

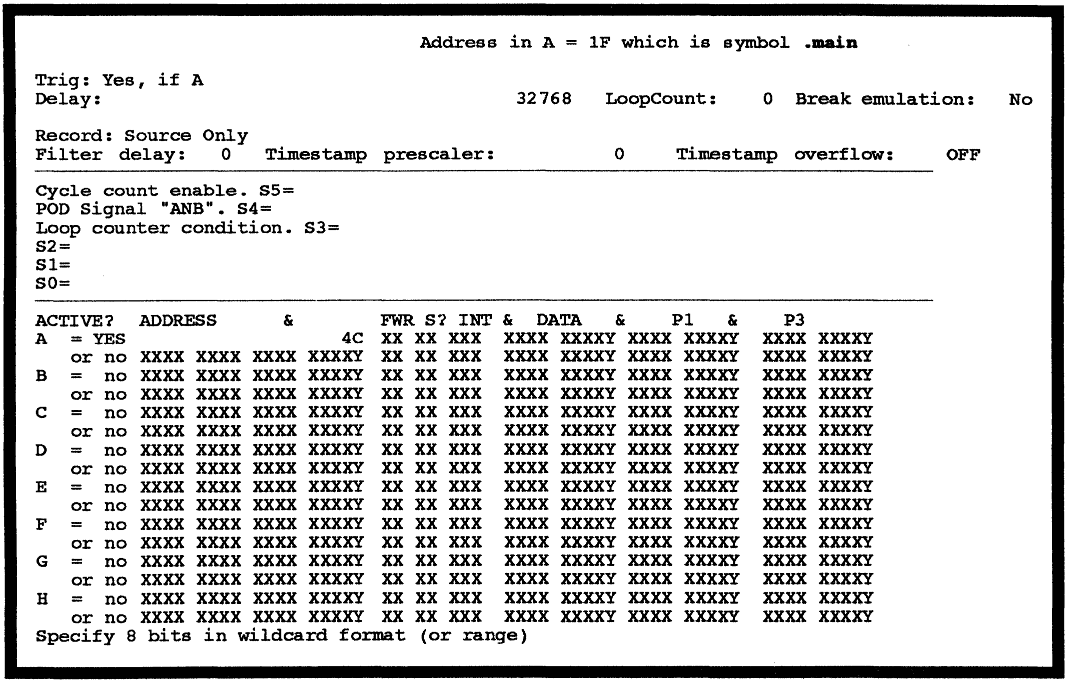
LoopCount The loop count field is set to 10 and the **Loop Counter** condtion (S3) is set to **A**. This means that the loop counter will decrement when the **A** condition goes from TRUE (high) to FALSE (low). Since the **A** condition is set to INT = 000, the loop counter will decrement every time the program returns from interrupt to the main program. When this has happened 10 times the CO bit is set, which then triggers the trace. The trace will continue to record until the number in the DELAY field decrements to zero (0). In this case it will continue to trace another 32,768 frames.

Cycle Count The cycle count (S5) will count the number of cycles between the events that are set up. In this case, when the event in **B** (start of the main program) comes true the counter will start counting. When the event in **C** (the start of the external interrupt) comes true, the counter will stop.

Time Count The time count field will then be updated to show the execution time in microseconds (us). This number is based on the XTal command or the -x parameter that was set at the time the emulator was started.

Since we have shown you how to get in and out of the trace functions and emulation, we will not tell you every key press that you need to make in the following examples.

Advanced Setup Example 4 In this example let's say that you want to capture only high level lines in the trace display. Refer to Figure 52 for setting up the trace (TS).



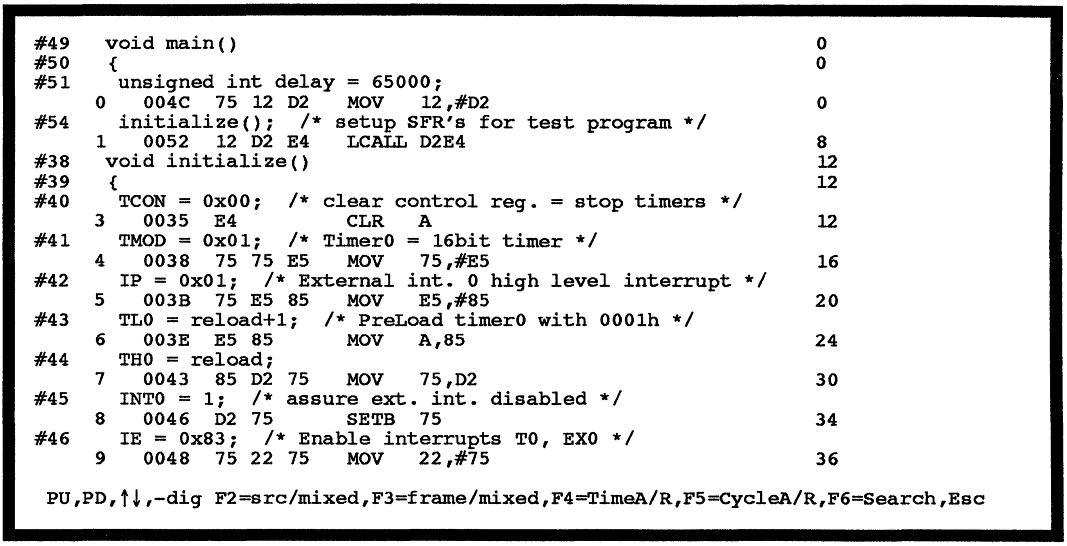
```
                                    Address in A = 1F which is symbol .main

Trig: Yes, if A
Delay:                                    32768   LoopCount:    0  Break emulation:   No

Record: Source Only
Filter delay:   0   Timestamp prescaler:          0     Timestamp overflow:    OFF
Cycle count enable. S5=
POD Signal "ANB". S4=
Loop counter condition. S3=
S2=
S1=
S0=
ACTIVE?  ADDRESS       &         FWR S? INT &  DATA    &     P1    &     P3
A   = YES                    4C  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
    or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
B   =  no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
    or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
C   =  no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
    or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
D   =  no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
    or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
E   =  no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
    or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
F   =  no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
    or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
G   =  no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
    or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
H   =  no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
    or no XXXX XXXX XXXX XXXXY  XX XX XXX   XXXX XXXXY XXXX XXXXY   XXXX XXXXY
Specify 8 bits in wildcard format (or range)
```

Figure 52. Advanced Trace Setup Example 4

After the trace has stopped, display the trace buffer with the TD command:

```
> TD
```

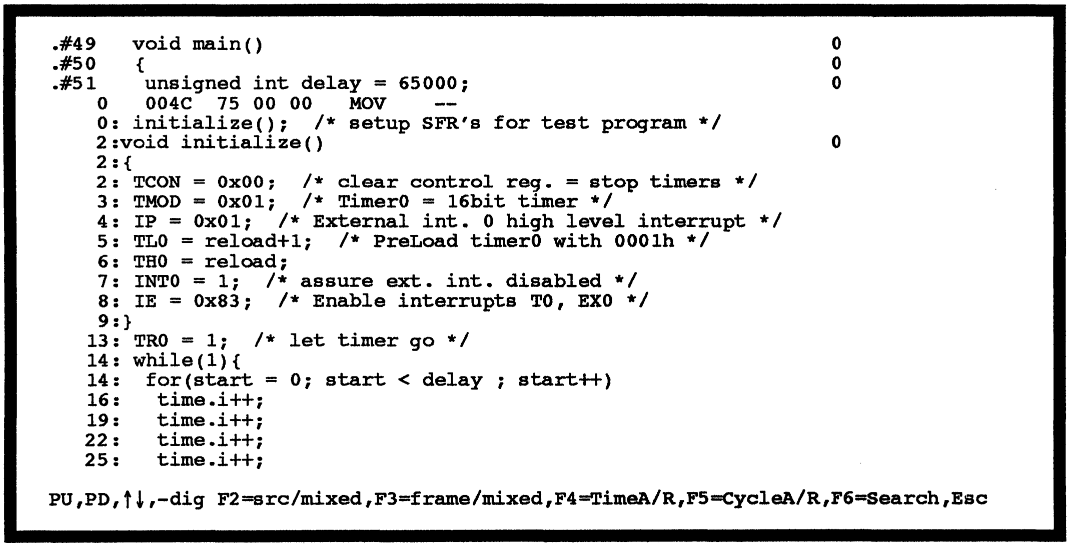
Now your screen should look something like Figure 53. You will notice that the assemble level instructions are included in the trace display as well as the source lines.



```
#49   void main()                                                        0
#50   {                                                                  0
#51    unsigned int delay = 65000;
      0   004C  75 12 D2   MOV   12,#D2                                  0
#54    initialize(); /* setup SFR's for test program */
      1   0052  12 D2 E4   LCALL D2E4                                    8
#38   void initialize()                                                  12
#39   {                                                                  12
#40    TCON = 0x00;  /* clear control reg. = stop timers */
      3   0035  E4         CLR   A                                       12
#41    TMOD = 0x01;  /* Timer0 = 16bit timer */
      4   0038  75 75 E5   MOV   75,#E5                                  16
#42    IP = 0x01;  /* External int. 0 high level interrupt */
      5   003B  75 E5 85   MOV   E5,#85                                  20
#43    TL0 = reload+1;  /* PreLoad timer0 with 0001h */
      6   003E  E5 85      MOV   A,85                                    24
#44    TH0 = reload;
      7   0043  85 D2 75   MOV   75,D2                                   30
#45    INT0 = 1;  /* assure ext. int. disabled */
      8   0046  D2 75      SETB  75                                      34
#46    IE = 0x83;  /* Enable interrupts T0, EX0 */
      9   0048  75 22 75   MOV   22,#75                                  36

 PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 53. Trace Display Frames 'TD' For Example 4

If you press the <F2> key, you can eliminate the assembly level instructions in the trace display. Your screen should look something like what is shown in Figure 54.



```
.#49   void main()                                                        0
.#50   {                                                                  0
.#51    unsigned int delay = 65000;                                       0
   0    004C  75 00 00   MOV    --
   0: initialize();  /* setup SFR's for test program */
   2:void initialize()                                                    0
   2:{
   2: TCON = 0x00;  /* clear control reg. = stop timers */
   3: TMOD = 0x01;  /* Timer0 = 16bit timer */
   4: IP = 0x01;  /* External int. 0 high level interrupt */
   5: TL0 = reload+1;  /* PreLoad timer0 with 0001h */
   6: TH0 = reload;
   7: INT0 = 1;  /* assure ext. int. disabled */
   8: IE = 0x83;  /* Enable interrupts T0, EX0 */
   9:}
  13: TR0 = 1;  /* let timer go */
  14: while(1){
  14:  for(start = 0; start < delay ; start++)
  16:   time.i++;
  19:   time.i++;
  22:   time.i++;
  25:   time.i++;

PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 54. Trace Display Frames <F2> For Example 4

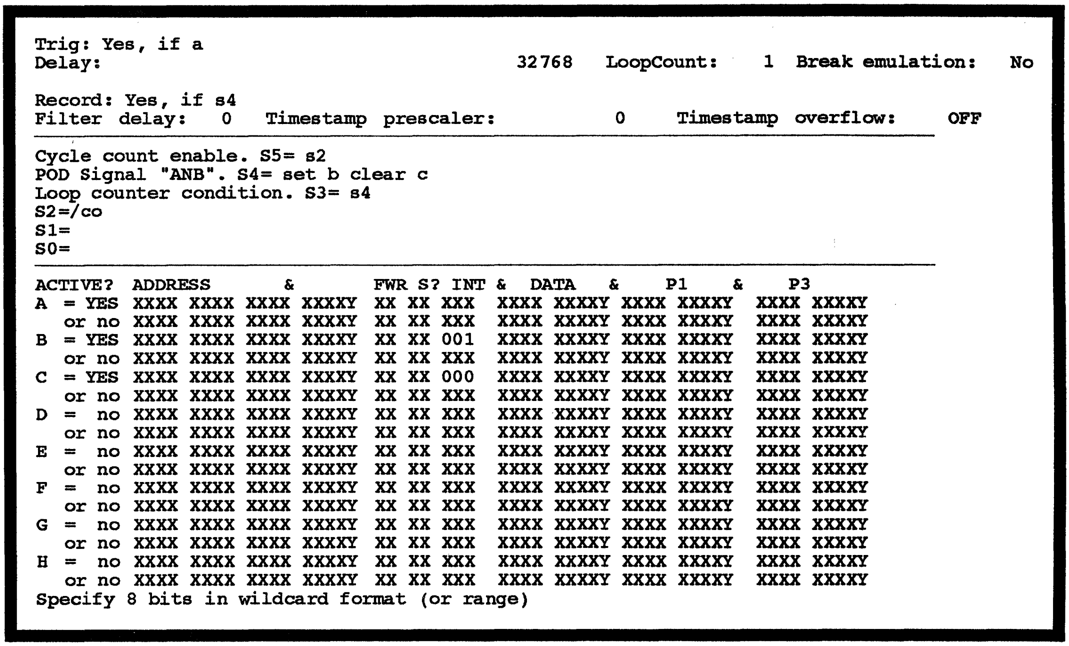
Comments With the RECORD set to capture "SOURCE ONLY", a user can capture many lines of the source code for evaluation. As an experiment, you can alter the 'C' variable *delay* in this program using the following commands:

```
* cb 4E = 0
* cb 51 = 3                    /* set delay = 3 */
```

Now display the trace, and you can see what kind of effect this change had on the execution of the code. Use the PgUp and PgDn keys to scan through some of the pages of the trace disassembly:

```
* res chi
* g
```

Advanced Setup Example 5 In this example we will capture a section of code that is between two of our qualifiers, known as a windowed capture. We will also use the cycle count field to see how long this piece of code takes to execute. Using the Trace Setup command (TS), set up your trace to look like Figure 55.



```
Trig: Yes, if a
Delay:                                    32768   LoopCount:    1  Break emulation:   No

Record: Yes, if s4
Filter delay:  0   Timestamp prescaler:          0     Timestamp overflow:     OFF

Cycle count enable. S5= s2
POD Signal "ANB". S4= set b clear c
Loop counter condition. S3= s4
S2=/co
S1=
S0=

ACTIVE? ADDRESS       &        FWR S? INT & DATA    &     P1     &      P3
A  = YES XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
B  = YES XXXX XXXX XXXX XXXXY  XX XX 001  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
C  = YES XXXX XXXX XXXX XXXXY  XX XX 000  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
D  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
E  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
F  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
G  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
H  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
Specify 8 bits in wildcard format (or range)
```

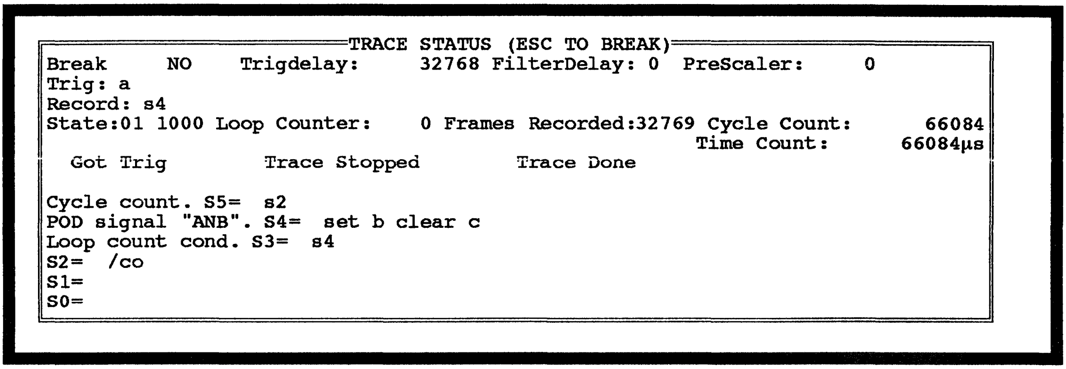
Figure 55. Advanced Trace Setup Example 5

6

Now restart the emulation and trace with the following commands:

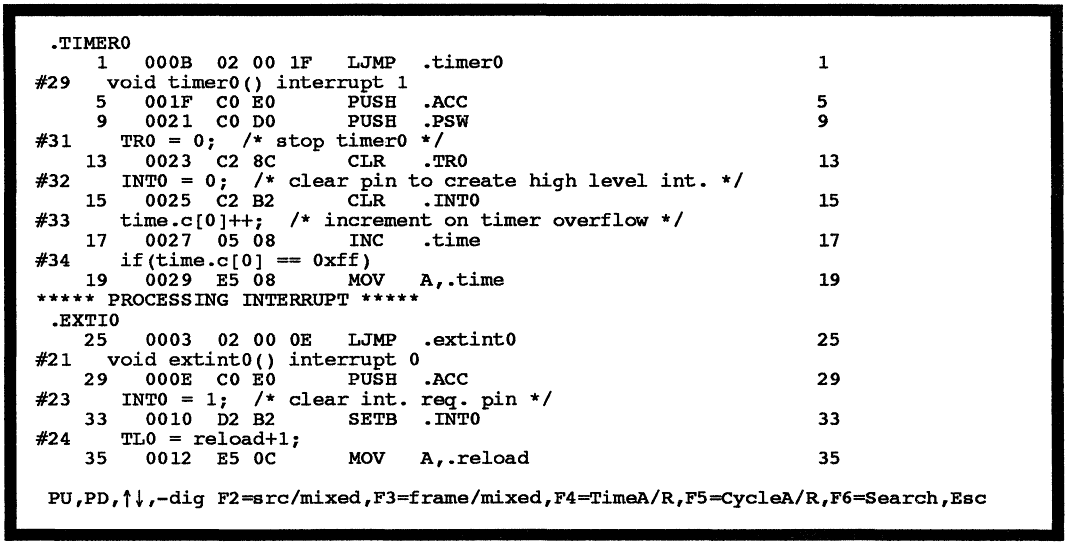
```
<esc> /* exit TS */
<esc> /* break emulation */

* res chi
* g
```

Tthe trace status reports "Trace Stopped", which takes about a minute. If you examine the TRACE STATUS WINDOW (Figure 56) you can see how long this section of code took to execute (66084μs). Use the TD command to display the trace buffer. Your display should look something like Figure 57.



```
=======================TRACE STATUS (ESC TO BREAK)=========================
Break      NO    Trigdelay:      32768 FilterDelay: 0  PreScaler:      0
Trig: a
Record: s4
State:01 1000 Loop Counter:    0 Frames Recorded:32769 Cycle Count:     66084
                                                  Time Count:     66084µs
  Got Trig          Trace Stopped          Trace Done

Cycle count. S5=  s2
POD signal "ANB". S4=  set b clear c
Loop count cond. S3=  s4
S2=  /co
S1=
S0=
```

Figure 56. Trace Display From Example 5



```
 .TIMER0
     1    000B  02 00 1F   LJMP  .timer0                          1
#29   void timer0() interrupt 1
     5    001F  C0 E0      PUSH  .ACC                             5
     9    0021  C0 D0      PUSH  .PSW                             9
#31     TR0 = 0;  /* stop timer0 */
    13    0023  C2 8C      CLR   .TR0                            13
#32     INT0 = 0;  /* clear pin to create high level int. */
    15    0025  C2 B2      CLR   .INT0                           15
#33     time.c[0]++;  /* increment on timer overflow */
    17    0027  05 08      INC   .time                           17
#34     if(time.c[0] == 0xff)
    19    0029  E5 08      MOV   A,.time                         19
***** PROCESSING INTERRUPT *****
 .EXTI0
    25    0003  02 00 0E   LJMP  .extint0                        25
#21   void extint0() interrupt 0
    29    000E  C0 E0      PUSH  .ACC                            29
#23     INT0 = 1;  /* clear int. req. pin */
    33    0010  D2 B2      SETB  .INT0                           33
#24     TL0 = reload+1;
    35    0012  E5 0C      MOV   A,.reload                       35

 PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 57. Trace Display From Example 5

Comments In the above example we also used the state machine bits to allow us to have a ONE-SHOT capture of the "set B clear C" condition. By doing this we can get an idea of how long this code segment took to execute. Using the state machine bits to qualify a function can be vary helpful when you have a section of code that is either in a loop or called may times.

LoopCount: Set for a single pass of S3 condtion.

Record: Record only the code between the **B** (true) & **C** (true).

Cycle Count: Count the number of cycles between **B** and **C**.

LoopCounter:(S3) Set loop counter for (Set-Clear) condtion for single-shot window display.

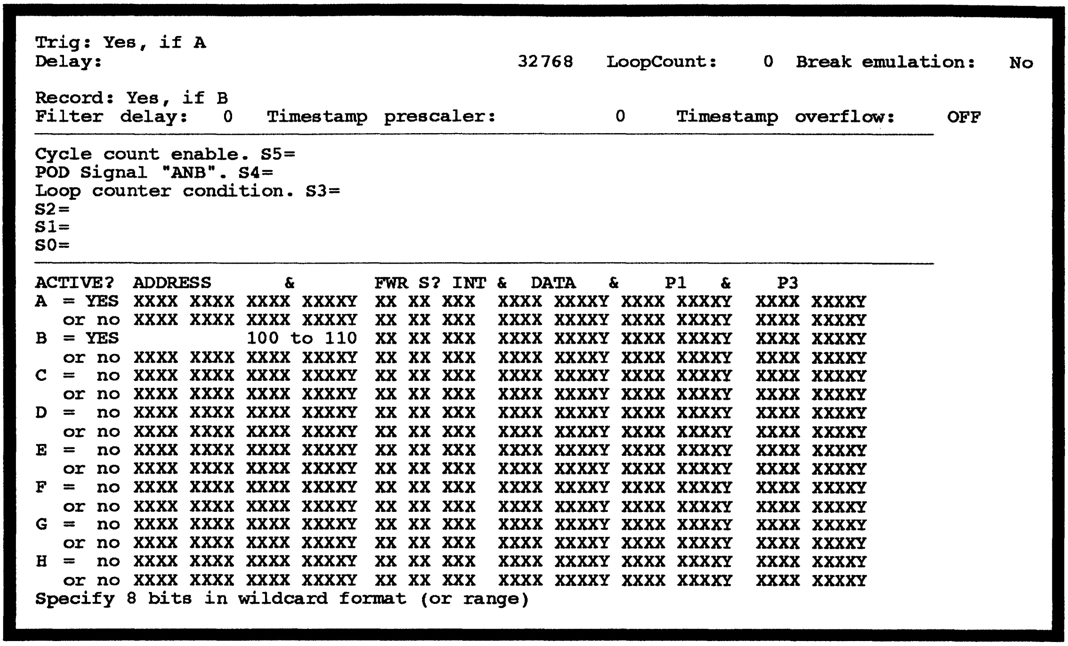
S2: Set for (not CO) = record cycles while loopcount ≠ 0.

In the above configuration the trace will record all the frames between the qualifier **B** and the qualifier **C**. The cycle count will be enabled only while the loop count is greater than 0; given the loop counter condition of being given in the S4 state bit.

Advanced Setup Example 6 In this next example we will show how the *Filter delay* field can affect the trace display. We will now use the TEST.A03 program that we used in the early examples of this tutorial. Please follow the next steps to clear out the code memory, remove the old symbols, and load the new code:

```
* fill 0 to ffff 0 cb
* rem symbols
* load test.a03
* res chi
```

Please refer to Figure 58 for the trace setup values.



```
Trig: Yes, if A
Delay:                                     32768   LoopCount:     0  Break emulation:    No

Record: Yes, if B
Filter delay:   0   Timestamp prescaler:          0     Timestamp overflow:     OFF
-------------------------------------------------------------------------------------
Cycle count enable. S5=
POD Signal "ANB". S4=
Loop counter condition. S3=
S2=
S1=
S0=
-------------------------------------------------------------------------------------
ACTIVE? ADDRESS        &         FWR S? INT &  DATA    &     P1    &     P3
A = YES XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
B = YES           100 to 110  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
C =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
D =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
E =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
F =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
G =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
H =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
Specify 8 bits in wildcard format (or range)
```

Figure 58. Advanced Trace Setup Example 6

Now restart the emulation and trace with the following commands:

```
* res chi
* g
```

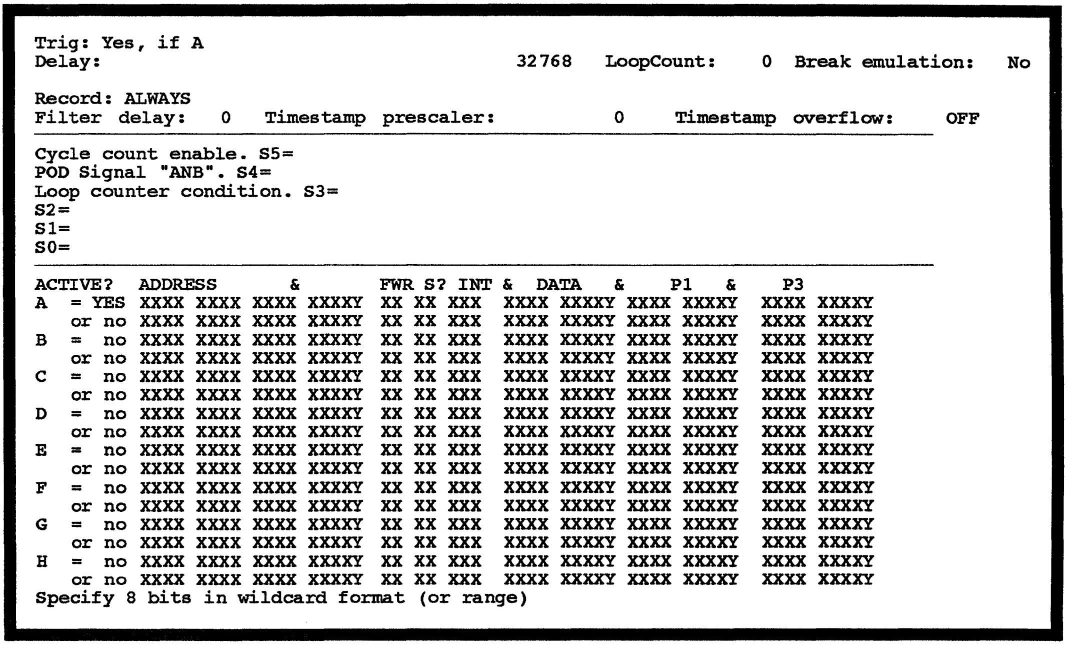
Stop the trace with the TE command after a few seconds. Use the TD command to display the trace buffer. As you can see, the data at the addresses of the MOVX is missing. Now display the trace buffer with the TDF command and then press the <F3> key to see what the disassembly looks like.

Next, alter the trace setup by changing **ONLY** the Filter delay field of the trace setup. Enter the number **1** in this field.

Now restart the trace as you did before. Break the emulation and examine the trace buffer with the TD command. Now you can see the data at the MOVX'x in the disassembly of the trace buffer. This is a handy tool to use if you want to capture the data from MOVX in the trace buffer without including the address or address range of the possible MOVX reads and writes in the quailifiers.

SEARCH FUNCTION

The EMUL51-PC software contains a search function in the trace display modes. This helps you get around in the trace buffer quickly. You can set up your trace with the example in Figure 58.



```
Trig: Yes, if A
Delay:                               32768  LoopCount:    0  Break emulation:    No

Record: ALWAYS
Filter delay:   0  Timestamp prescaler:        0    Timestamp overflow:    OFF
--------------------------------------------------------------------------------
Cycle count enable. S5=
POD Signal "ANB". S4=
Loop counter condition. S3=
S2=
S1=
S0=
--------------------------------------------------------------------------------
ACTIVE?  ADDRESS      &        FWR S? INT &  DATA   &     P1   &     P3
A  = YES XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
B  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
C  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
D  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
E  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
F  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
G  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
H  =  no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
   or no XXXX XXXX XXXX XXXXY  XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY
Specify 8 bits in wildcard format (or range)
```

Figure 58. Trace Setup Example

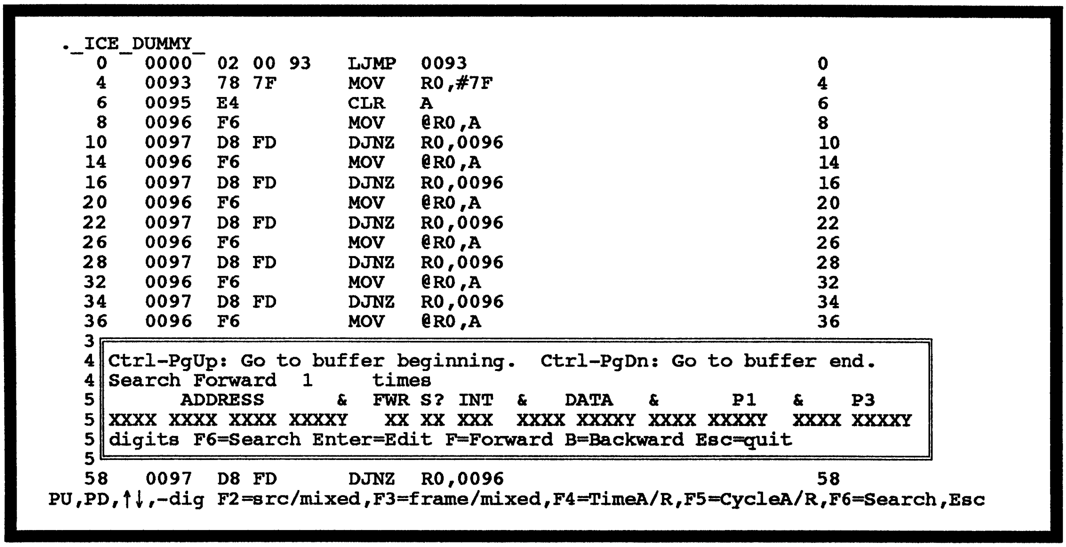
Now restart the emulation and trace with the following commands:

```
<esc> /* exit TS */
<esc> /* break emulation */

* res chi
* g
```

When you see the message displayed in the trace status window that the trace has *STOPPED.* Press the <ESC> key and then display the trace buffer with the **TD** command:

```
<esc> /* Break emulation */
* TD
```

Now press the function key 6 **<F6>** and the search window will pop up on the screen. See Figure 59.



```
 ._ICE_DUMMY_
   0   0000  02 00 93   LJMP  0093                              0
   4   0093  78 7F      MOV   R0,#7F                            4
   6   0095  E4         CLR   A                                 6
   8   0096  F6         MOV   @R0,A                             8
  10   0097  D8 FD      DJNZ  R0,0096                          10
  14   0096  F6         MOV   @R0,A                            14
  16   0097  D8 FD      DJNZ  R0,0096                          16
  20   0096  F6         MOV   @R0,A                            20
  22   0097  D8 FD      DJNZ  R0,0096                          22
  26   0096  F6         MOV   @R0,A                            26
  28   0097  D8 FD      DJNZ  R0,0096                          28
  32   0096  F6         MOV   @R0,A                            32
  34   0097  D8 FD      DJNZ  R0,0096                          34
  36   0096  F6         MOV   @R0,A                            36
  3╔═════════════════════════════════════════════════════════════════════╗
  4║Ctrl-PgUp: Go to buffer beginning.  Ctrl-PgDn: Go to buffer end.     ║
  4║Search Forward  1     times                                          ║
  5║      ADDRESS       &  FWR S? INT  &    DATA    &       P1    &    P3 ║
  5║XXXX XXXX XXXX XXXXY   XX XX XXX  XXXX XXXXY XXXX XXXXY  XXXX XXXXY  ║
  5║digits F6=Search Enter=Edit F=Forward B=Backward Esc=quit           ║
  5╚═════════════════════════════════════════════════════════════════════╝
  58   0097  D8 FD      DJNZ  R0,0096                          58
PU,PD,↑↓,-dig F2=src/mixed,F3=frame/mixed,F4=TimeA/R,F5=CycleA/R,F6=Search,Esc
```

Figure 59. Trace Search Window

On the second line of the trace search window you will see whether a forward or backward search is to be done and the number of times it will match the condition set up in the 'ADDRESS & FWR S? INT & DATA & P1 & P3' fields. The default search direction is FORWARD and number of times is one (1).

By pressing either the **F** or the **B** key you can change the direction of the search. By entering a new number you will change the times field number. An integer from 0 to 65535 can be accepted. This will allow you to set a number to check for the Nth occurrence of whatever pattern that you have entered.

By using the Ctrl-PgUp or Ctrl-PgDn you can get to the first or last page in the trace buffer quickly.

By pressing the **ENTER** key you can enter the search condition fields,which can be edited much like the trace setup window fields. By using the ESC key when in the pattern field, you will exit that setup mode.

Use the F6 key to initiate the search function.

Conclusion

As we hope you have discovered from working through this tutorial, the trace is a very flexible tool with many possible applications. To develop your understanding of this important EMUL51-PC feature, try experimenting with other trace setups and observe their effects on the trace capture.

Depending on the type of POD board you have connected to the EMUL51-PC, the P1, and P3 columns in the TS window can be made to show other ports (besides P1 and P3). To do this, move the appropriate jumpers on the POD board. Likewise, the P1 and P3 columns can be made to show external signals; simply remove the jumpers and connect external wires to the jumper pin(s).

7 SIMULATOR OPTION

Overview 7-3
Invocation 7-3
User Interface 7-4
Simulator Engine 7-4
External Environment Simulator
(EVENT - MACRO) 7-4
External Environment Simulator
(SERIAL CHANNEL) 7-6

7

SIMULATOR OPTION

Overview The optional SIMUL51-PC has been derived from the EMUL51-PC's software. The EMUL51-PC is an in-circuit emulator, so its software communicates with the emulator's hardware. In the simulator version the hardware has been replaced with software that simulates the 8051. All hardware features like ports, timers, interrupts etc. are also simulated. External hardware is simulated through a scheme using the EVENT command combined with macros as described below.

SIMUL51-PC consists of the following three major components:

1. User interface (Same as EMUL51-PC).
2. Simulator engine (Instruction and SFR simulator).
3. External environment simulator. EVENT - MACRO and SERIAL CHANNEL simulation.

Invocation SIMUL51 is normally started with two parameters which specify how much CODE memory and XDATA memory the program needs to allocate. The -m parameter specifies amount of CODE memory expressed in number of Kilobytes. The -x parameter specifies amount of XDATA memory expressed in number of Kilobytes.

Examples:

```
SIMUL51 -m10 -x5                ;10K CODE and 5K XDATA
SIMUL51 -m64 -x64               ;64K CODE and 64K XDATA
SIMUL51 -m1                     ;1K CODE and no XDATA
SIMUL51                         ;32K CODE (default) and no XDATA (default)
```

Don't allocate more memory than you need! It will "eat up" memory that could be used for symbol tables and other things. The HEAP command can be used to allocate more memory in EMS or EXT memory or on DISK. (See the HEAP command for details.) For your convenience six batch files for starting the SIMUL51-PC have been included:

ss.bat	;SIMUL51 -m4 -x4	4K CODE 4K XDATA
ms.bat	;SIMUL51 -m32 -x32	32K CODE 32K XDATA
ls.bat	;SIMUL51 -m64 -x64	64K CODE 64K XDATA
sso.bat	;SIMUL51O -m4 -x4	4K CODE 4K XDATA
mso.bat	;SIMUL51O -m32 -x32	32K CODE 32K XDATA
lso.bat	;SIMUL51O -m64 -x64	64K CODE 64K XDATA

SIMUL51O is an "overlay version" of SIMUL51. The benefit of SIMUL51O over SIMUL51 is that it uses 90K less of your DOS memory. The disadvantage is that it might be somewhat slower.

User Interface Please refer to chapters 1, 3, and 6 in this manual.

Simulator Engine The simulator performs a number of tasks:

- It allocates memory to simulate the different memory areas in the 8051.

- It simulates 8051 instructions. The speed at which this is done is about 650 times slower than executing code at 12 MHz if you run the simulator on a '286 20MHz computer.

- It simulates the 8051 hardware like SFR's and interrupts.

- The trace is simulated only as straight tracing. Features like triggering and filtering which are available in the emulator are not implemented in the simulator.

External Environment Simulator (EVENT - MACRO) For a formal description of the EVENT command please type EVE at the command line and then F1.

An EVENT is a READ or WRITE to any memory location in the 8051. When a READ or WRITE occurs as specified with the EVENT command, a break occurs <u>or</u> a macro is executed. (See MACROS 1-15)

A macro is basically one or a number of commands executed from a preprogrammed macro rather than from the command line. When the macro is executed the program continues to execute, after the macro has completed. For more details on macros see MACROS page 1-15. We will start with examples to clarify how the EVENT - MACRO concept is used. After that follows a formal description of the EVENT command and the MACDELAY command.

Examples:

```
EVE * 10C PC            ;breaks when the program attemps to execute an instruction
                        a address 10C.

EVENT W 34 db           ;breaks when the program writes to internal data memory 34
                        hex
EVENT R 80 rb           ;breaks when the program reads address 80.  (P0)
EVENT W 1289 XB 12      ;breaks when the value 12 hex is written to address 1289

EVENT W 1289 XB 12 56   ;breaks when a value within the range 12 to 56 hex inclusive,
                        is written to address 1289 hex


EVENT W 45 DB :MAC      ;executes macro :MAC when the program writes to data
                        address 45 hex,then the program continues.

EVENT W 1234 TO 1256 XB 30 40 :FEED     ;executes macro :FEED when a value
                                        between 30 and 40 hex inclusive is
                                        written to an address between 1234 and
                                        1256 hex inclusive
```

It is important to note that macros must be defined before they are used in an EVENT.

Examples of how a macro can work with two EVENTs:

```
DEF :MAC
RBI .P3+2 = RBI .P1+5
EM
```

Together with the following macros this example forces bit 2 of Port 3 (P3) to follow bit 5 of Port 1 (P1).

```
EVENT W .P1 RBY :MAC
EVENT W .P1+5 RBI :MAC
```

To summarize: When an instruction writes to P1, whether byte or bit, P3.2 will be made to follow P1.5.

You may want to implement a delay from when the EVENT is detected until the macro is executed. This can be done with the MACDELAY command. If you want to delay changing P3.2 for 50ms from when P1.5 was written to, you would define a new macro which uses the MACDELAY command:

```
DEF :MACD
MACDELAY 50 MSEC :MAC
EM
```

If you change :MAC in the EVENTs above to MACD, 50ms delay will be added before :MAC is executed. Observe that it is 50ms counted in cycles where it is assumed that 12MHz is used.

External Environment Simulator (SERIAL CHANNEL)

For a formal description and examples of the EVENT and MACDELAY commands, please refer to the COMMANDS Chapter.

For a formal description of the SER command please type SER at the command line and then F1.

Simulating the serial channel of the 8051 presents a challenge both for us and for you. For us to make it as easy for you as possible, and for you to set it up so that it resembles the real world. We start with the formal definition of the SERIAL command which is used to specify how serial data is received and transmitted. For a formal description and examples of the SERIAL Command, please refer to the COMMANDS Chapter.

APPENDIX A

Absolute disassembly from the EMUL51-PC emulator for the 'C' program ADV_TRA.

```
 ._ICE_DUMMY_
0000  02 00 93   LJMP  0093
 .EXTI0
0003  02 00 0E   LJMP  .extint0
0006  00         NOP
0007  00         NOP
0008  00         NOP
0009  00         NOP
000A  00         NOP
 .TIMER0
000B  02 00 1F   LJMP  .timer0
#21   void extint0() interrupt 0
000E  C0 E0      PUSH  .ACC
#23     INT0 = 1;                 /* clear int. req. pin */
0010  D2 B2      SETB  .INT0
#24     TL0 = reload+1;
0012  E5 0C      MOV   A,.reload
0014  04         INC   A
0015  F5 8A      MOV   .TL0,A
#25     TH0 = reload;
0017  85 0C 8C   MOV   .TH0,.reload
#26     TR0 = 1;                  /* restart timer0 */
001A  D2 8C      SETB  .TR0
#27   }
001C  D0 E0      POP   .ACC
001E  32         RETI
#29   void timer0() interrupt 1
001F  C0 E0      PUSH  .ACC
0021  C0 D0      PUSH  .PSW
#31     TR0 = 0;                  /* stop timer0 */
0023  C2 8C      CLR   .TR0
#32     INT0 = 0;                 /* clear pin to create high level int. */
0025  C2 B2      CLR   .INT0
#33     time.c[0]++;              /* increment on timer overflow */
0027  05 08      INC   .time
#34     if(time.c[0] == 0xff)
0029  E5 08      MOV   A,.time
002B  B4 FF 02   CJNE  A,#FF,.#36
#35             time.c[1]++;
```

APPENDIX A

```
002E  05 09     INC   09
#36  }
0030  D0 D0     POP   .PSW
0032  D0 E0     POP   .ACC
0034  32        RETI
#38  void initialize()
#39  {
#40    TCON = 0x00;          /* clear control reg. = stop timers */
0035  E4        CLR   A
0036  F5 88     MOV   .TCON,A
#41    TMOD = 0x01;                  /* Timer0 = 16bit timer */
0038  75 89 01  MOV   .TMOD,#01
#42    IP = 0x01;            /* External int. 0 high level interrupt */
003B  75 B8 01  MOV   .IP,#01
#43    TL0 = reload+1;               /* PreLoad timer0 with 0001h */
003E  E5 0C     MOV   A,.reload
0040  04        INC   A
0041  F5 8A     MOV   .TL0,A
#44    TH0 = reload;
0043  85 0C 8C  MOV   .TH0,.reload
#45    INT0 = 1;             /* assure ext. int. disabled */
0046  D2 B2     SETB  .INT0
#46    IE = 0x83;            /* Enable interrupts T0, EX0 */
0048  75 A8 83  MOV   .IE,#83
#47  }
004B  22        RET
#49  void main()
#50  {
#51    unsigned int delay = 65000;
004C  75 16 FD  MOV   16,#FD
004F  75 17 E8  MOV   17,#E8
#54    initialize();         /* setup SFR's for test program */
0052  12 00 35  LCALL .initialize
#55    TR0 = 1;              /* let timer go */
0055  D2 8C     SETB  .TR0
#57    while(1){
#58            for(start = 0; start < delay ; start++)
0057  E4        CLR   A
0058  FB        MOV   R3,A
0059  FA        MOV   R2,A
005A  C3        CLR   C
005B  EB        MOV   A,R3
```

APPENDIX A

```
005C  95 17     SUBB  A,17
005E  EA        MOV   A,R2
005F  95 16     SUBB  A,16
0061  50 F4     JNC   .#57
#59                      time.i++;
0063  AF 0B     MOV   R7,0B
0065  AE 0A     MOV   R6,0A
0067  AD 09     MOV   R5,09      [00] <- [00]
0069  AC 08     MOV   R4,.time
006B  12 01 3E  LCALL .?C_LPUSH
006E  7F 01     MOV   R7,#01
0070  7E 00     MOV   R6,#00
0072  7D 00     MOV   R5,#00
0074  7C 00     MOV   R4,#00
0076  12 01 2B  LCALL .?C_LADD
0079  8F 0B     MOV   0B,R7
007B  8E 0A     MOV   0A,R6
007D  8D 09     MOV   09,R5
007F  8C 08     MOV   .time,R4
0081  0B        INC   R3
0082  EB        MOV   A,R3
0083  70 01     JNZ   0086
0085  0A        INC   R2
0086  80 D2     SJMP  005A
#60     }
#61   }
0088  22        RET
0089  04        INC   A
008A  08        INC   R0
008B  00        NOP
008C  00        NOP
008D  00        NOP
008E  00        NOP
008F  01 0C     AJMP  000C
0091  00        NOP
0092  00        NOP
0093  78 7F     MOV   R0,#7F
0095  E4        CLR   A
0096  F6        MOV   @R0,A
0097  D8 FD     DJNZ  R0,0096
0099  75 10 00  MOV   .?C_IBP,#00
009C  75 0E 00  MOV   .?C_XBP,#00
```

APPENDIX A

```
009F  75 0F 00   MOV   0F,#00
00A2  75 11 00   MOV   .?C_PBP,#00
00A5  75 81 17   MOV   .SP,#17
00A8  02 00 E6   LJMP  00E6
00AB  02 00 4C   LJMP  .main
00AE  E4         CLR   A
00AF  93         MOVC  A,@A+DPTR
00B0  A3         INC   DPTR
00B1  F8         MOV   R0,A
00B2  E4         CLR   A
00B3  93         MOVC  A,@A+DPTR
00B4  A3         INC   DPTR
00B5  40 03      JC    00BA
00B7  F6         MOV   @R0,A
00B8  80 01      SJMP  00BB
00BA  F2         MOVX  @R0,A
00BB  08         INC   R0
00BC  DF F4      DJNZ  R7,00B2
00BE  80 29      SJMP  00E9
00C0  E4         CLR   A
00C1  93         MOVC  A,@A+DPTR
00C2  A3         INC   DPTR
00C3  F8         MOV   R0,A
00C4  54 07      ANL   A,#07
00C6  24 0C      ADD   A,#0C
00C8  C8         XCH   A,R0
00C9  C3         CLR   C
00CA  33         RLC   A
00CB  C4         SWAP  A
00CC  54 0F      ANL   A,#0F
00CE  44 20      ORL   A,#20
00D0  C8         XCH   A,R0
00D1  83         MOVC  A,@A+PC
00D2  40 04      JC    00D8
00D4  F4         CPL   A
00D5  56         ANL   A,@R0
00D6  80 01      SJMP  00D9
00D8  46         ORL   A,@R0
00D9  F6         MOV   @R0,A
00DA  DF E4      DJNZ  R7,00C0
00DC  80 0B      SJMP  00E9
00DE  01 02      AJMP  0002
```

APPENDIX A

```
00E0  04        INC   A
00E1  08        INC   R0
00E2  10 20 40  JBC   20,0125
00E5  80 90     SJMP  0077
00E7  00        NOP
00E8  89 E4     MOV   E4,R1
00EA  7E 01     MOV   R6,#01
00EC  93        MOVC  A,@A+DPTR
00ED  60 BC     JZ    00AB
00EF  A3        INC   DPTR
00F0  FF        MOV   R7,A
00F1  54 3F     ANL   A,#3F
00F3  30 E5 09  JNB   .ACC+5,00FF
00F6  54 1F     ANL   A,#.TIMER0
00F8  FE        MOV   R6,A
00F9  E4        CLR   A
00FA  93        MOVC  A,@A+DPTR
00FB  A3        INC   DPTR
00FC  60 01     JZ    00FF
00FE  0E        INC   R6
00FF  CF        XCH   A,R7
0100  54 C0     ANL   A,#C0
0102  25 E0     ADD   A,.ACC
0104  60 A8     JZ    00AE
0106  40 B8     JC    00C0
0108  E4        CLR   A
0109  93        MOVC  A,@A+DPTR
010A  A3        INC   DPTR
010B  FA        MOV   R2,A
010C  E4        CLR   A
010D  93        MOVC  A,@A+DPTR
010E  A3        INC   DPTR
010F  F8        MOV   R0,A
0110  E4        CLR   A
0111  93        MOVC  A,@A+DPTR
0112  A3        INC   DPTR
0113  C8        XCH   A,R0
0114  C5 82     XCH   A,.DPL
0116  C8        XCH   A,R0
0117  CA        XCH   A,R2
0118  C5 83     XCH   A,.DPH
011A  CA        XCH   A,R2
```

APPENDIX A

```
011B  F0        MOVX  @DPTR,A
011C  A3        INC   DPTR
011D  C8        XCH   A,R0
011E  C5 82     XCH   A,.DPL
0120  C8        XCH   A,R0
0121  CA        XCH   A,R2
0122  C5 83     XCH   A,.DPH
0124  CA        XCH   A,R2
0125  DF E9     DJNZ  R7,0110
0127  DE E7     DJNZ  R6,0110
0129  80 BE     SJMP  00E9
.?C_LADD
012B  EF        MOV   A,R7
012C  25 15     ADD   A,15
012E  FF        MOV   R7,A
012F  EE        MOV   A,R6
0130  35 14     ADDC  A,14
0132  FE        MOV   R6,A
0133  ED        MOV   A,R5
0134  35 13     ADDC  A,13
0136  FD        MOV   R5,A
0137  EC        MOV   A,R4
0138  35 12     ADDC  A,12
013A  FC        MOV   R4,A
013B  02 01 65  LJMP  .?C_LSTKDEC
.?C_LPUSH
013E  E5 0D     MOV   A,.?C_DSTKLEVEL
0140  05 0D     INC   .?C_DSTKLEVEL
0142  60 10     JZ    0154
0144  D0 F0     POP   .B
0146  D0 E0     POP   .ACC
0148  C0 12     PUSH  12
014A  C0 13     PUSH  13
014C  C0 14     PUSH  14
014E  C0 15     PUSH  15
0150  C0 E0     PUSH  .ACC
0152  C0 F0     PUSH  .B
0154  8F 15     MOV   15,R7
0156  8E 14     MOV   14,R6
0158  8D 13     MOV   13,R5
015A  8C 12     MOV   12,R4
015C  22        RET
```

APPENDIX A

```
.?C_LPULL
015D  AF 15    MOV   R7,15
015F  AE 14    MOV   R6,14
0161  AD 13    MOV   R5,13
0163  AC 12    MOV   R4,12
.?C_LSTKDEC
0165  E5 0D    MOV   A,.?C_DSTKLEVEL
0167  14       DEC   A
0168  60 10    JZ    017A
016A  D0 F0    POP   .B
016C  D0 E0    POP   .ACC
016E  D0 15    POP   15
0170  D0 14    POP   14
0172  D0 13    POP   13
0174  D0 12    POP   12
0176  C0 E0    PUSH  .ACC
0178  C0 F0    PUSH  .B
017A  15 0D    DEC   .?C_DSTKLEVEL
017C  22       RET
```

INDEX

? 3-77
25 3-77
2500 AD 3-43
8031.STR 2-26
8032 2-26, 4-4, 4-13, 4-18
8051 4-3, 4-21, 4-31
80535 3-40, 4-4, 4-151
80C452 4-105
AASM 3-5
ABR 3-5
ACALL 3-61
ACC 3-7, 3-52
ACOND 3-73, 6-27, 6-31
Adapter(s) 4-6, 4-10
ADJ, AdjWin 1-7, 3-7
Advanced Trace, ATR 1-28
AOMF 3-42
Archimedes See IAR (Archimedes)
ASCII 3-19, 3-30, 3-32
ASM 3-5, 3-7
Assemble 3-7
AV 1-17, 3-8
Avocet 3-43
BANK 2-39, 3-8
BANKADDRESS 2-40, 3-9
BANKBYTE 2-39, 3-9
BANKMODULE 2-40, 2-41, 3-9
BANKSEGMENT 2-40, 3-10
BANKSY1 3-10
Bargraph 3-49
Batch files 2-19, 2-20, 2-26
Baud 2-29, 2-31
BB 1-47, 3-10
BCOND 3-73
BIC 3-11, 3-13, 3-19, 3-34
BIN(Y) See SUF
Binary 3-29, 3-44, 3-65, 6-29
Box-M, -S, -P 1-4, 1-5, 2-27, 2-28

BR . . . 3-14, 3-36, 3-55
Breakpoints . . . 1-46, 2-43, 3-5, 3-11, 3-14, 3-15, 3-16, 3-19, 3-33, 3-38, 3-55, 3-70
BRM . . . 3-14
BRS . . . 3-15, 3-36
BRX . . . See Breakpoints
BSO/TASKING . . . 1-18, 3-41, 3-43
BUFFERSIZE, BUF . . . 3-15
Bus(es) . . . 4-4, 4-5, 4-6
BYB . . . 1-47, 3-11, 3-12, 3-16, 3,19, 3-33
C-Compiler . . . 3-20, 3-32
C-variable . . . 1-17
CAL, CALC . . . 3-17
CALL . . . 3-8, 3-61
CASE . . . 3-23, 3-27
CB . . . 3-18
CBYTE . . . 3-18
CFLOAT . . . 3-32
Checksum . . . 3-17
CHIP . . . 3-55
CINT . . . 3-32
CLB . . . 3-19
Clock . . . Ch.4 (PODs)
CODE . . . 3-23, 3-27, 3-33
Command . . . 1-6, 1-9, 3-39
COU, COUNT . . . 3-20
Counters . . . 1-46
CPL . . . 3-10
CpyChr . . . 1-7
CpyCmd . . . 1-7
Crystal . . . 3-58, Ch.4 (PODs)
CSTRING . . . 3-32
Ctrl+W . . . 1-20
CUINT . . . 3-32
CW . . . 1-14, 3-20
Cycle Count Enable . . . 1-34
Cycles . . . 1-47, 3-58
DAS, DASM . . . 3-21
DATA . . . 3-11, 3-12, 3-16, 3-18, 3-31, 3-33 3-46, 3-65, 3-66, 3-71
DBYTE . . . 3-18, 3-22

Decimal(T) . . . 3-65
DEF, DEFINE . . . 1-49, 3-22
Delay . . . 1-32
DFLOAT . . . 3-32
DINT . . . 3-32
DIP40-ISO . . . 4-11, 5-9
DIP40-PLCC44 . . . 4-12
DIR . . . 3-23
DIS, DISABLE . . . 1-53, 1-56, 3-23
Disassembler, disassembly . . . 3-21, 3-71, 4-24, 6-16
DJNZ . . . 3-12, 3-16
DMA . . . 2-20, 3-71, 4-5, 4-69, 4-73, 4-105
DOM, DOMAIN . . . 3-26
DOS . . . 3-69
DP . . . 3-26
DPH, DPL . . . 3-26
DPTR . . . 3-26
DSTRING . . . 3-32
DUINT . . . 3-32
E-Bus . . . 4-5
E0, E1 . . . Ch.4 (PODs), 6-29
ELSE . . . 3-39
EM/ . . . Ch.4 (PODs)
ENA, ENABLE . . . 1-56, 3-27
END . . . 3-20, 3-22, 3-29, 3-39, 3-54, 3-73
ENDIF . . . 3-39
ENDREPEAT . . . 3-54
EPROM . . . See PROM
EPSEN/ . . . 4-75
EQ . . . 3-39, 3-54
EVA, Evaluate . . . 3-29
EVENT . . . 3-30, 7-4
EVENT-MACRO . . . 7-4
EW . . . 4-173
EXI, EXIT . . . 3-31
Expansion . . . 3-23, 3-24, 3-27
EXT-DIP40 . . . 4-12
EXTERNAL ENVIRONMENT . . . 7-4, 7-6
FIL, FILL . . . 3-31
Filter Delay . . . 1-33
FOREVER . . . 3-34, 3-36
FORM, FORMAT . . . 3-32

FRAME(S) 3-70, 6-24, 6-28, 6-34

Franklin/Keil 1-17, 2-36, 3-21, 3-41, 3-43

GB 3-33

GI 3-11, 3-13, 3-33

GLOBAL 3-23, 3-27

GO 1-46, 3-14, 3-19, 3-34, 3-41, 3-58

GoCurs 1-8

GOT 6-27

GR 3-14, 3-36

GS 1-16, 3-15, 3-36

HEAP 3-37

Help 1-7, 1-9, 3-23, 3-27, 3-38, 6-5, 6-8

HEX, hexadecimal 3-42, 3-56, 3-65, 3-76

HEXCODE 3-56

Histogram See Bargraph

HlpOnf 1-8

I/O 4-4, 4-17

IAR (Archimedes) 1-17, 2-34, 3-28, 3-41, 3-43

IB 3-38

IE, IEN0, IEN1 3-40

IF 3-38

IIP0, IIP1 3-40

INC 4-39

INCLUDE 2-20, 3-39

Installation Ch. 2

INT 3-40, 3-73

Integers 3-32

INTERMETRICS/WHITESMITHS 1-18, 3-41

Interrupt(s) 3-40, 3-61

IP 3-40, 4-22, 4-23

IP0 3-40

IP1 3-40

ITRACE, ITRAC 1-24, 6-28, 6-36

J See SN

JBC 3-10

JMP 3-8

Jumpers Ch.2, Ch.4

L See LINE

LABEL 3-23, 3-27

LC, LOOP COUNTER 1-24, 1-32, 1-36, 3-40, 6-27, 6-29

LENGTH 3-14, 3-19, 3-21

LIN, LINE, Linestep 3-41, 3-63, 5-11

Line OVER 3-44
LIS, List 3-41, 3-63, 3-71, 3-72
LOA, LOAD 3-42
LOB, Load Binary 3-44
Local variables 1-18
Logical Operators 1-49
LOOP COUNTER . . . . See LC
LOP 3-44, 3-56
LOVER 3-44
LOW 3-44, 3-57
LPT1 3-41
LSYMB 3-23, 3-27
MAC, Macro(s) 1-49, 3-22, 3-23, 3-24, 3-27
3-39, 3-45, 3-49, 3-53, 7-4
MACDELAY 3-45, 7-4, 7-6
MAIN 3-71
MAP 3-46
MAPC 3-46
MAPX 3-46
MEM 3-46
Menus 1-6
MOD 3-47
MOD_BEG 3-67
MOD_END 3-67
Module(s) 3-22, 3-24, 3-27, 3-42
3-53, 3-57, 3-67
MOV 3-12, 3-16
MUL 3-12
N . . . . See SN
;name . . . . See MACRO
NOCODE 3-42, 3-56
NOHIT 3-47
NOS, NOSNOW 3-47
NOSYMBOLS 3-42, 3-56
NOTRIG 1-23, 3-73, 6-36
NSYMB 3-23, 3-27
NUMB 3-23, 3-27
NxtCmd 1-8
NxtWin 1-7
Object, OBJ 3-28, 3-42, 3-63
Octal 3-29, 3-65
OP 3-14

ORG 3-5, 3-8
ORL 3-16
OSC .... Ch.4 (PODs)
OST 3-36, 3-47, 3-71
P0, Port 0 4-3
P1 3-7
P2 4-3
P3 3-7
Path 2-26, 3-63
PBYTE .... 3-15, 3-18, See CB
PC 1-3, 3-26, 3-35, 3-48, 3-61, 3-64
PGA 4-4, 4-12, 4-73, 4-95, 4-105, 4-151
4-161, 4-180
PGA68-PLCC68 4-12
PL/M-51 .... 1-15, 3-21, 3-41, 3-63, 4-21
PLCC 4-4, 4-12, 4-73, 4-211
PLUS 3-23, 3-27
Ports .... 3-71, 4-4, 4-6, Ch. 4 (PODs)
Post-trigger 6-25, 6-35
Power-down, power-up 4-5, 4-7, 4-9, 4-10, 4-31,
4-180, 4-211
PPA 3-44, 3-48, 3-56
Pre-trigger 1-24, 6-25
Processor 2-29
Program Fields 1-32
PROM .... 1-9, 3-56, Ch.4 (PODs)
PrvCmd 1-7
PS/2 1-4
PSEN .... 1-46, Ch.4 (PODs), 5-9
PSW 3-13, 3-51
PUSH: 3-12
PUT 3-49
QR, QRA, qualification 1-22, 3-49, 3-70,
6-28, 6-40
QUIET 3-51
-r0, -r1 1-18
R0, R1 .... 3-12, 3-16, 3-52, 3-55
RBIT 3-18, 3-51
RBS 3-51, 3-55
RBYTE 3-11, 3-12, 3-15, 3-18, 3-24, 3-28, 3-52
RCS 3-52
ReaD/ 4-4

Read-Modify-Write 3-13
Real-time 3-35, 3-36, 4-3
Record 1-33
REG 3-23, 3-27, 3-52
Registers 3-7, 3-11, 3-13, 3-19, 3-36
3-52, 3-55, 3-64, 4-21, 4-22
RELMS 3-42, 3-43
REM, remove 3-6, 3-42, 3-53
REP, repeat 1-8, 3-54
RES, Reset, RST 3-6, 3-40, 3-55, 3-58
Ch.4 (PODs), 5-9, 5-11
RETI 3-40
RS 3-53
SAP 3-56
SAV 3-56
SAW 3-44, 3-57
SBUF 3-13, 3-63, 4-6
SCON 3-13
SCOPE 2-41, 3-57
SDLC 4-69, 4-73
SEB 3-57
SEC 3-58
SERIAL 3-58
SERIAL CHANNEL 7-6
SFR 3-11
SHELL 3-60
SIG 3-27
SIMUL51-PC 7-3
Simulator 7-3
SL 3-60
SN 3-60
SNAP 3-62
SO 3-63
SOURCE 3-21, 3-23, 3-27, 5-11
Source-level 1-14
SP, stack pointer 3-52, 3-64
Speed? 1-7
SrcAsm 1-7
SST 3-63
STA 3-64
Stack pointer . . . . . See SP
Stack-variables 1-21

State Machine 1-37

STE, Step 3-36, 3-41, 3-47, 3-64

STOP 3-35, 3-62

STR files (*.STR) 2-26

Subdirectory 2-27

Subroutine 3-61

SUF, Suffix 3-65

SWD 3-65

SWR 3-65

SWX 3-66

SX 1-37

SY0, SY0A 3-33, 3-67

SYM files for microcontrollers (*.SYM) 2-26

SYM, symbol(s) 1-19, 2-20, 2-26, 3-18, 3-22, 3-23, 3-25, 3-41, 3-43, 3-53, 3-67

SYMBOLIC 3-23, 3-28, 3-43

SYMLOAD 3-67

SYNTAX 3-23, 3-27

SYS, system 3-46, 3-69

-t 2-20

TABS 3-69

TB 1-11, 1-25, 3-69, 6-27

TBR 1-11, 3-70

TC 1-11, 1-24, 3-70, 6-29

TCON 3-13

TD 1-11, 1-25, 3-71, 6-23, 6-33, 6-34, 6-35, 6-37, 6-38

TDD 3-71

TDF 1-11, 1-25, 3-71, 6-24, 6-32, 6-33

TDFL 1-11, 1-25, 3-71

TDFS 1-11, 1-25, 3-45, 3-47, 3-71

TDL 1-11, 1-25, 3-72

TDP 1-11, 1-25, 3-71, 6-26

TDPL 1-11, 1-25, 3-72

TDS 1-11, 1-25, 3-71

TDSL 3-72

TE 1-11, 1-25, 3-71, 3-73, 6-23, 6-27, 6-31, 6-37

TEQ 4-22

TH0 3-13

TH1 3-13

TH2 3-13

THEN 3-39

Timeout 5-8
Timers 1-47
Timestamp 1-33, 1-34
TL0, TL1 3-26
Tone-dialing 2-20
TR . . . . . 1-11 , 3-73
TRACE 1-3, 1-11, 1-22, 1-23, 1-26, 6-22, 6-28
TRIG 1-23, 1-32, 6-23, 6-27, 6-28, 6-34, 6-36
Troubleshooting 5-3
TS 1-11, 1-22, 1-30, 3-74, 6-28, 6-33, 6-36, 6-37, 6-40
TSG, TSGET 3-74
TSP, TSPUT 3-74
UNDEF 3-23
Unnamed 3-22
Untyped 3-43
Variable(s) 1-17, 1-18
VER 1-13, 3-75
VGA 3-77
WAI, WAIT 3-75
WAT 1-20
WATCH 1-18, 1-23, 3-27, 3-75
Watchdog . . . . . Ch.4 (PODs)
WHILE 3-20, 3-54
Window(s) 1-6, 1-14, 3-23, 3-27, 6-22, 6-27, 6-31, 6-34, 6-36, 6-37, 6-40
WOnOff 1-7
WRI 3-76
XBYTE 3-76, 5-10
XDATA 3-23, 3-27, 3-46, 3-66
XFLOAT 3-32
xINT 3-32
XSTRING 3-32
XTAL 3-76
XUINT 3-32

We will be pleased to answer any questions you may have, or discuss your comments. If you prefer, you may write to us at the following address:

NOHAU CORPORATION
51 E. Campbell Ave.
Campbell, CA 95008

Phone	**(408) 866-1820**
FAX	**(408) 378-7869**

NOHAU Sales Offices, Reps, And Distributors

INTERNATIONAL

AUSTRALIA

Electro Optics Pty. Ltd.
Level 1, 1 Nelson Street
Kenthurst
New South Wales 2156
Australia
Local tel: (02) 654 1873
Int. Tel: 61 - 2 654 1873
Fax: 61 - 2 654 1539

AUSTRIA

Burisch Elektronik-Bauteile GmbH
Leopoldauer Strasse 29
A-1210 Wien
Austria
Local tel: 0277 20-0
Int. Tel: 43 1 277 20-0
Fax: 43 1 277 20 / 277
Telex: 111082 BURIS A

BELGIUM

See Netherlands

BENELUX

(Belgium Netherlands Luxembourg)
See Netherlands

BRAZIL

Anacom Software
Praça dos Expedicionáros,
101-Sala 01
São Bernardo do Campo
São Paulo CEP 09750
Brazil
Local Tel: (011) - 458-8755
Int. Tel: 55 - 11 - 458-8755
Fax: 55 - 11 - 458-3614

CANADA

Techmatron Instruments Inc.
130 Principale
Laval, Prov. of Québec H7W 3S6
Canada
Tel: 1 - 514 689-5889
Fax: 1 - 514 689-0868

CROATIA

See Slovenia

CZECHOSLOVAKIA

EDI Corporation
P. O. Box 1
25003 Brandys n/L
Czechoslovakia
Local Tel: 0202 - 2683
Int. Tel: 42 - 202 - 2683
Fax: 42 - 202 - 2208

DENMARK

Nohau Danmark A/S
Vibeholms Alle 11-15
DK 2605 Brøndby
Denmark
Tel: 45 43 44 60 10
Fax: 45 43 44 60 20

FINLAND

Yleiselektroniikka Oy
Luomannotko 6
SF-02200 Espoo
Finland
Local Tel: 90 - 4526 21
Int. Tel: 358 0 4526 21
Fax: 358 0 4526 2202
Telex: 123212 yleoy sf

NOHAU Sales Offices, Reps, And Distributors

INTERNATIONAL

FRANCE

Emulations s.a.r.l.
A13 - Burospace
Chemin de Gizy
91572 Bièvres Cedex
France
Local Tél: (1) 69 41 28 01
Int. Tél: 33 1 69 41 28 01
Fax: 33 - 1 60 19 29 50

GERMANY

iSYSTEM GmbH
Einsteinstraße 5
D-8060 Dachau
Germany
Local Tel: 08131/25083
Int. Tel: 49 - 8131-25083
Fax: 49 - 8131 14024

GREAT BRITAIN

Nohau UK Ltd.
The Station Mill, Alresford
Hampshire S024 9JG
England
Local Tel: 0962-733140
Int. Tel: 44 - 962-73 31 40
Fax: 44 - 962-73 54 08

GREECE

UPTEK S. A.
P. O. Box 25014
Athens 10026
Greece
Local Tel: (01) 862 9901
Int. Tel: 30 - 1 - 862 9901
Fax: 30 - 1 - 862 9902
Telex: 224487 UPTK GR

HUNGARY

DATAWARE Electronic
Engineering Ltd.
Angol u. 22
H-1149 Budapest
Hungary
Local Tel: (1) 163 7461
Int. Tel: 36-1-163 7461
Fax: 36-1-163 5867

INDIA

Microcomputer Solutions
Pvt. Ltd.
22/6 Premnagar (behind
Pushpa Mangal Karyalaya)
Bibwe Wadi
Pune 411 039
India
Local Tel: (0212) 422164
Int. Tel: 91 - 212 - 422164
Fax: 91 - 212 - 436798

NOHAU Sales Offices, Reps, And Distributors

INTERNATIONAL

INDONESIA

See Singapore

ISRAEL

ITEC Ltd.
P. O. Box 10002
Tel Aviv 61100
Israel
Local Tel: 03 - 491202
Int. Tel: 972 - 3 491202
Fax: 972 - 3 497661
Telex: 341307 ITEC IL

ITALY

BRM Italiana Srl
Via Medail 4
10144 Torino
Italy
Local Tel: (011) 771.00.10
Int. Tel: +39 11 7710010
Fax: +39 11 7710198

KOREA

Zeus Computer Co. Ltd.
#17-1 Chuan Dong Nam-Ku
Inchon
Korea
Local Tel: (032) 860-5470
Int. Tel: 82 - 32 860-5470
Fax: 82 - 2 784-7844

LUXEMBOURG

See Netherlands

MALAYSIA

See Singapore

NETHERLANDS

TRITEC Benelux B. V.
Postbus 212
3340 AE Hendrik Ido Ambacht
Netherlands
Local Tel: (01858) 16133
Int. Tel: +31-1858-16133
Fax: +31-1858-20030

NEW ZEALAND

G.T.S. - Nilsen Instruments
5 Porters Avenue; Eden Terrace
P. O. Box 9613; Newmarket
Auckland
New Zealand
Local Tel: 09-3092464
Int. Tel: 64-9-3092464
Fax: 64-9-3092968

NORWAY

Nortelco AS
Ryensvingen 3
Postboks 116, Manglerud
N-0612 Oslo
Norway
Local Tel: 02 - 67 40 20
Int. Tel: 47 - 2 - 67 40 20
Fax: 47 - 2 - 67 40 30

NOHAU Sales Offices, Reps, And Distributors

INTERNATIONAL

PORTUGAL

Fatrónica Lda
Rua Manuel dos Santos No. 17A
Olalas
1900 Lisboa
Portugal
Local Tel: 01 - 80 95 18
Int. Tel: 351 - 1-847 37 11
Fax: 351 - 1-847 37 12
Telex: 65904 FATRON P

ROMANIA

Datatim S.A.
Str. Gh. Lazar Nr. 9
1900 Timisoara, Romania
Local Tel: 961-30078
Int. Tel: +40-961-30078
Fax: +40-961-33418
Telex: 71380

SINGAPORE

Systemteq (S'pore) Pte Ltd
Block 3014 Ubi Road 1
#02-302 Singapore 1440
Tel: +65 749-0870
Fax: +65 749-0332

SLOVENIA

AHIL D.O.O.
Krizna 1a
SLO-61110 Ljubljana
Slovenia
Local Tel: 061 - 445-526
Int. Tel: 38 - 61 - 445-526
Fax: 38 - 61 - 445-526

SOUTH AFRICA

Eagle Electronics
31 - 35 Hout St.
P. O. Box 4376
Cape Town 8000
South Africa
Local Tel: (021) 23-4943
Int. Tel: 27 - 21 23-4943
Fax: 27 - 21 24-4637

SPAIN

Semiconductores, S. A.
Rda. General Mitre, 240
08006 Barcelona
Spain
Local Tel: (93) 217 23 40
Int. Tel: 34 - 3 217 23 40
Fax: 34 - 3 217 65 98
Telex: 97787 SMCD E

SWEDEN

Nohau Electronik AB
Fosievägen 6
214 31 Malmö
Sweden
Local Tel: 040 - 92 24 25
Int. Tel: 46 - 40 92 24 25
Fax: 46 - 40 96 81 61
Telex: 32152 Cymko S

SWITZERLAND

Thau Computer AG
Löwenstrasse 21
8953 Dietikon
Switzerland
Local Tel: 01 - 740 41 05
Int. Tel: 41 - 1 740 41 05
Fax: 41 - 1 740 15 67

NOHAU Sales Offices, Reps, And Distributors

INTERNATIONAL

TAIWAN

Hi-Lo System Research Co., Ltd.
4F, No. 2, Sec. 5
Ming Shen E. Rd.
Taipei, Taiwan, R. O. C.
Local Tel: 02 7640215
Int. Tel: 886-2-7640215
Fax: 886-2-7566403
Telex: 24071 Hilores

THAILAND

Complex Technology Co., Ltd.
102/10 Krungkaseam Rd.
Thewes, Bangkok 10200
Thailand
Local Tel: (02) 281-9596
Int. Tel: 66-2-281-9596
Fax: 66-2-280-3649

YUGOSLAVIA

See Slovenia

NOHAU Sales Offices, Reps, And Distributors

UNITED STATES

Use this Zip Code chart to find sales office:

00001 - 01199	Contact Nohau Corp.
01201 - 02799	MA: AD Electronics
02801 - 02999	RI: AD Electronics
03001 - 03899	NH: AD Electronics
03901 - 04999	ME: AD Electronics
05001 - 05999	VT: AD Electronics
06001 - 06999	CT: AD Electronics
07001 - 07999	NJ: J. Kunze Co.
08001 - 08399	NJ: Delta Technical Sales
08401 - 08499	Contact Nohau Corp.
08501 - 08799	NJ: Delta Technical Sales
08801 - 08999	NJ: J. Kunze Co.
09001 - 09999	Contact Nohau Corp.
10001 - 11999	NY: J. Kunze Co.
12001 - 12399	Contact Nohau Corp.
12401 - 12799	NY: J. Kunze Co.
12801 - 16799	Contact Nohau Corp.
16801 - 19199	PA: Delta Technical Sales
19301 - 19699	PA: Delta Technical Sales
19701 - 19999	DE: Delta Technical Sales
20001 - 20399	DC: Embedded Technology
20401 - 20699	Contact Nohau Corp.
20701 - 20999	MD: Embedded Technology
21001 - 21699	Contact Nohau Corp.
21701 - 21999	MD: Embedded Technology
22001 - 22599	VA: Embedded Technology
22601 - 31999	Contact Nohau Corp.
32001 - 34999	FL: Trilogic Sunforce
35001 - 47999	Contact Nohau Corp.
48001 - 48599	MI: Dearborn Group
48601 - 48799	Contact Nohau Corp.
48801 - 48999	MI: Dearborn Group
49001 - 49199	Contact Nohau Corp.
49201 - 49299	MI: Dearborn Group
49301 - 84999	Contact Nohau Corp.
85001 - 86599	AZ: MicroWare Technology
87001 - 89999	Contact Nohau Corp.
90001 - 93599	CA: MicroWare Technology
93601 - 97999	Contact Nohau Corp.
98001 - 98599	WA: Exectronics
98601 - 99999	Contact Nohau Corp.

NOHAU Sales Offices, Reps, And Distributors

UNITED STATES

(Continued)

Use the Zip Code Chart to find sales office.

AD Electronics

10 Pepperidge Trail
Old Saybrook, CT 06475
Tel: 1 - 203 388-2204
Fax: 1 - 203 388-9607

The Dearborn Group, Inc.

29655 Eastfield Dr.
Farmington Hills, MI 48334
Tel: 1 - 313 932-1850
Fax: 1 - 313 932-2169

Delta Technical Sales

122 N. York Rd.
Hatboro, PA 19040
Tel: 1 - 215 957-0600
Fax: 1 - 215 957-0920

Embedded Technology Corp.

1307 Benicia Lane
Herndon, VA 22070
(Mail Address: P. O. Box 1065
Herndon, VA 22070)
Tel: 1 - 703 742-0040
Fax: 1 - 703 318-9390

Exectronics

26 243rd Place S. E.
Bothell, WA 98021
Tel: 1 - 206 947-2115
Fax: 1 - 206 481-7845

J. Kunze Co.

29 Jefferson St.
P. O. Box 312
Glen Cove, NY 11542
Tel: 1 - 516 671-2376
Fax: 1 - 516 674-0221

MicroWare Technology

9761 Caminito Suelto
San Diego, CA 92131
Tel: 1 - 619 693-4280
Fax: 1 - 619 693-1438

Nohau Corporation

51 E. Campbell Ave.
Campbell, CA 95008
Tel: 1 - 408 866-1820
Fax: 1 - 408 378-7869

Trilogic Sunforce, Inc.

1103 Hibiscus Blvd., Suite 310
West Melbourne, FL 32904
Tel: 1 - 407 722-1324
Fax: 1 - 407 722-1281

RENTALS:

Technilease

43 Fulton Street
Newark, NJ 07102
Tel: 1 - 800 451-RENT
(1-800 451-7368)
1 - 201 621-7368
Fax: 1 - 201 643-4409

USER ORIENTATION 1

INSTALLATION 2

COMMANDS 3

PODS 4

TROUBLESHOOTING 5

TUTORIAL 6

SIMULATOR 7

INDEX I

1 USER ORIENTATION

2 INSTALLATION

3 COMMANDS

4 PODS

5 TROUBLESHOOTING

6 TUTORIAL

7 SIMULATOR

I INDEX

nohau
CORPORATION
51 E. Campbell Ave.
Campbell, CA 95008