ORGANIZATION OF THIS MANUAL

This User's Manual is organized into several sections. The number of sections included is dependent on the software ordered with the emulator. A breakdown of the contents of each section is as follows:

Introduction:

This outline.

Emulator Manual:

This section consists of the Emulator User's Manual. This section should be consulted when using the emulator with a dumb terminal and as a reference for the features and operation of the emulator.

HMI-110/HMI-100 Manual:

This section consists of the User's Manual for the optional HMI-110 (Source Level Debugger) or HMI-100 (Symbolic Debugger) communication software. The operation of these communication packages is described in this section.

SourceGate Manual:

This section consists of the User's Manual for the optional SourceGate (Window Driven Source Level Debugger) communication software. The operation of the SourceGate program is described in this section.

Appendix:

This section consists of general information such as application notes, new features, and other information of interest to the user.

If there are any questions concerning this manual or the operation of the emulator, please contact HMI's Technical Support at (205) 881-6005.

HMI 200 SERIES
68000 USER'S MANUAL

Revision 2.1

To be added to HMI's registered user list, return the User Registration Page included at the front of this manual.

The contents of this document are believed to be accurate. However, Huntsville Microsystems, Inc. assumes no responsibility for the use of this document and issues no warranties, implied or otherwise on the contents herein. Furthermore, Huntsville Microsystems, Inc. reserves the right to revise this publication and to make changes to the contents, without obligation to notify any persons of such revisions or changes.

Table of Contents

LIST OF DRAWINGS AND TABLES

I. PREFACE

This manual describes the features and operation of the HMI- 200-68000 In-Circuit Emulator. We assume in writing this manual that you have a computer with a text editor and know how to use them. You will also need a cross assembler or compiler to convert your programs into 68000 code. Some communication program is needed to link your program to the emulator. A basic understanding of the 68000 processor is necessary in order to utilize the emulator to its fullest potential.

II. TUTORIAL

A. INTRODUCTION

The HMI-200-68000 emulator is the heart of a high performance development system. It combines the control of an in-circuit emulator with the tracing power of a logic analyzer. When it is linked to a computer by the proper communications software, it provides a complete debugging environment for hardware and software on 68000 microprocessor based systems.

As the title suggests, this chapter is intended to be tutorial in nature. Only the essential features of the emulator are described to help you get started. As you need more detail, you should refer to the following chapters which serve as the reference portion of this manual.

The emulator is designed to run in stand-alone operation by being interfaced with a dumb terminal or host computer via an RS-232 cable. When interfacing to a host computer, you will need either the symbolic debugger, HMI-100, or source level debugger, HMI-110.

The HMI-200-68000 features real-time emulation up to 12.5 MHz and 256K bytes of emulator memory. Options for 1M byte and battery back-up emulator memory are available. Two RS- 232 ports support baud rates up to 38.4K baud. Four events for break and trigger operations can be defined in terms of addresses, data, status or external signals. Break and trigger points can be defined in terms of predefined, or user defined, sequences of events. An interval timer and two 4K X 72 bit trace buffers provide operation history in the manner of a logic analyzer.

The HMI-200-68000 commands do not make any distinction between the emulator memory and the target system memory. The target system memory is always used, unless specifically mapped as emulator memory in the memory map.

B. SYSTEM CONFIGURATION

1. Hardware Configuration

The two RS-232 ports on the HMI-200-68000 are labeled "Main" and "Aux". The main port can be connected to a terminal or to a personal computer, and the auxiliary port can be connected to a printer or to a computer. Thus there are several possible configurations for your microprocessor development station. Three configurations are shown below.

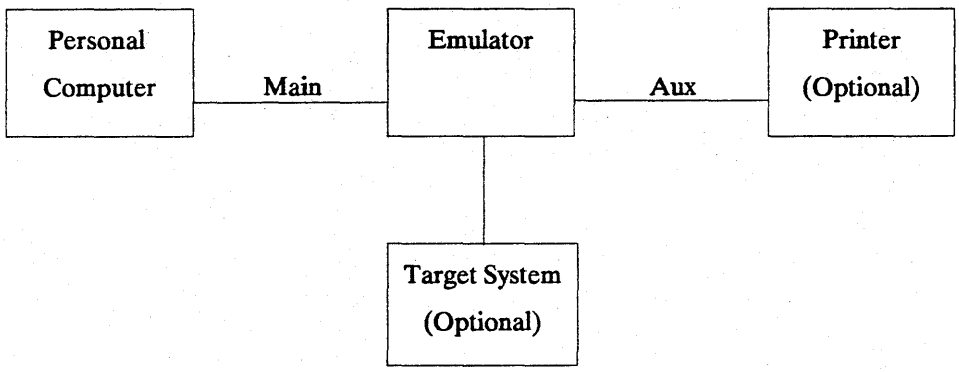


Fig. II - 1 Control from a personal computer.

In the configuration shown in Fig. II-1, the personal computer would be used to write and compile/assemble the code for the target system. Some communications software, either the symbolic debugger or source level debugger from HMI, will be needed to down- load the code into the target system and to send commands to the emulator. In many 68000 based designs the code will be written in a high level language such as C. If you will write your programs in C, then you will want the HMI source level debugger, because it will serve as a symbolic debugger for assembly language programs, and will also serve as a source level debugger for programs written in C.

The target system is noted to be optional - this is not a mistake. Why would you buy an emulator if you did not have a target system to debug? The answer is that in the design of microprocessor systems today, the

software and hardware phases of the design must progress simultaneously. For this reason the HMI-200-68000 has been designed to operate either with or without a target system. Software development can be initiated long before the first hardware model is available. When the first hardware model is available, it can be connected to the emulator and control can be gradually passed from the emulator to the target system - various blocks of the memory space can be assigned to the target hardware or to the emulator.

The auxiliary port can be used to dump a trace buffer to the printer.

Another configuration which could be used with a multi-user computer is shown in Fig. II-2.

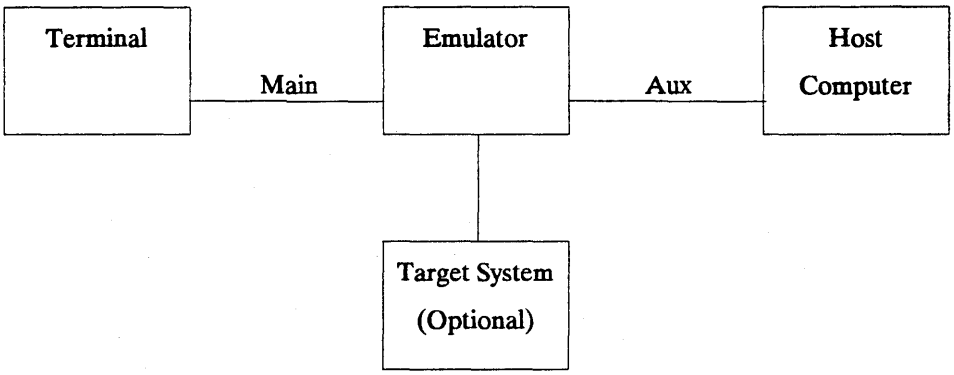


Fig. II - 2 A terminal served by a host computer

In this configuration the computer would be used to prepare the target software. Once compiled, the executable code can be downloaded from the host computer to the emulator/target system memory through the Aux. port on the emulator. The terminal would be used to communicate to the emulator and host computer. To communicate with the host, simply enter "pass-thru" mode on the emulator (see the passthru command description in chapter VI). In this mode, the emulator becomes transparent so that you will be communicating with the computer just as though the emulator were not present. When you leave the passthru mode, you will bring the emulator back on-line and therefore release the computer. You can see that this configuration would not tie up the computer nearly as much as the configuration in Fig. II-1.

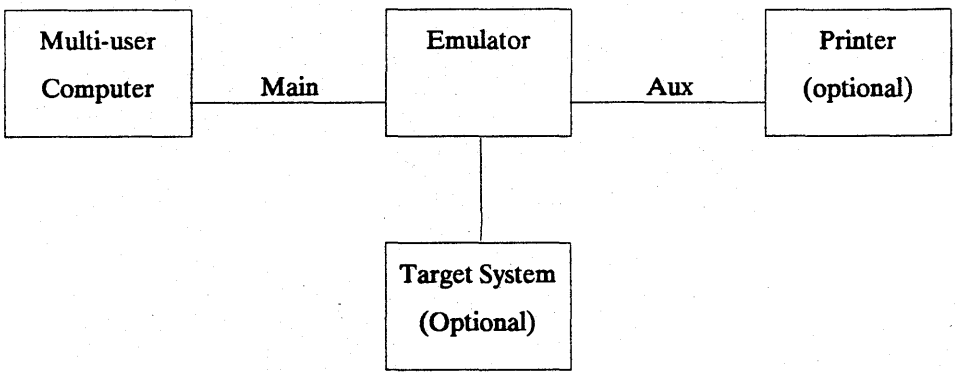


Fig. II - 3 Control from a multi-user computer

One more example configuration is shownin Fig II-3. In this case the emulator is controlled from a multi-user computer.

Here, as in the first case above, communication software is needed on the computer to control the emulator. Depending on the type of computer, it may be necessary to purchase the source code for the communication software to install it on the computer. This configuration would support both the software engineer and the hardware engineer simultaneously on the computer, although only one could be using the emulator at any time.

2. Software Configuration

When your system components are connected and you apply power to the system, you enter the communications program and go to the configuration menu of the emulator. The parameters you define here are saved in non-volatile memory so that you do not have to define your system each time it is turned on. In the left block and bottom half of this menu, you define the memory map of your system, allocating memory to be Supervisor Program/Data or User Program/Data. Any memory addresses which are not mapped to the emulator are mapped to the target system. The right block of this menu allows you to select the control parameters such as the source of the clock and the data acknowledge signal. The interrupt control and bus arbitration control signals can be enabled or disabled from this menu.

C. BREAKPOINTS

In order to debug a microprocessor system it is necessary to be able to set breakpoints. Setting a breakpoint allows you to execute your program until some condition is met. The HMI-200-68000 has two methods of setting breakpoints. The simplest method is to enter up to four breakpoint addresses or symbols on the command line with the GO (G) command. These breakpoints stop the emulator as soon as the address of any one is accessed - that is, they have an assumed pass count of one. A breakpoint set on the command line is interpreted as an address field on an op-code execution. These command line breakpoints are temporary in the sense that they are cleared when one causes the emulator to stop. Temporary breakpoints, if desired, must be entered each time a G command is executed. Temporary breakpoints take precedence over, and are independent of, breakpoints set from the menus. This means that if a G command with a breakpoint is executed, the events and sequences (described below) which define breakpoints from the menus are not used to stop emulation and also are not changed. Trigger conditions and the timer function are also not active when temporary breakpoints are defined on the command line.

The other method of setting breakpoints on the HMI-200-68000 is more versatile. From the menus, events are defined, and then breakpoints are defined in terms of sequences of these events. From the event menu, up to four events can be defined. An event is some combination of an address, data, status signal, external signal and pass count. For example you might define event A to be the 27th time that the processor writes the hex data C3xx to some address in the range 002Exx while the external signal on lead 2 is at the logic 1 level. Of course, most event definitions are simpler than this, but the capability is there.

When up to four events have been defined, you proceed to the Sequence Menu. Here you define a break emulation condition, trace trigger, timer operation and emulator output in terms of sequences of the previously defined events. Break emulation can be set ON to stop emulation on a sequence of events. This sequence can be one of the 14 predefined sequences, or you can define a sequence in terms of the events A, B, C and D connected with the functions AND, OR, THEN, WITHOUT and parentheses. For example A OR B OR C OR D would set four independent breakpoints. It is important to remember that this is a SE-

QUENCE menu in order to understand the construction of the sequences. For example the AND function, as A AND B, means that event A with its passcount is true (has occurred) and B with its pass count is true. A and B could become true in either order, but probably not at the same time. The function THEN is similar to AND except that an order is specified. A THEN B means that when A becomes true, the emulator will start looking for B. A complete description of the rules for constructing sequences is given in chapter V with the discussion of the Sequence Menu.

After the events and break emulation condition is defined, you can start your program from the Command Menu by the GO command. It is possible that you would want to execute part of your program before the emulator started looking for the break condition. In this case you could exit from the menus to the command line and issue a GO command with a temporary breakpoint. This would ignore and not alter the event or sequence definitions. At this point you might want to start watching for the break emulation sequence, so you could type a GO command without a breakpoint on the command line or you could return to the Command Menu to issue the GO command.

It is also possible to set pass counts on the events. If you use the sequence A OR B to break emulation, you may want the 7th time event A occurs or the 34th time B occurs to cause the break. Simply set the pass count for A to 7 and the pass count for B to 34. The Pass Count box in the Sequence Menu shows two numbers for each event, for example B = F9D1/0021H. This means that event B has occurred hex F9D1 times since the last GO or CONTINUE command and that the pass count for B is set to hex 21. In this case B was not used to stop emulation because it occurred more times than its pass count.

The pass counter is dynamically updated. This could be useful if you wanted to watch the number of times an event occurred. Just return to the Sequence Menu with the system running and watch the count of that event.

D. THE INTERVAL TIMER

The sequence timer counts microseconds between the start sequence and the stop sequence up to a maximum time of 71 minutes. To use the interval timer it is necessary to enable it on the Sequence Menu by tog-

gling the OFF/ON field to ON. The timer is then started and stopped by sequences of events defined on that menu. The definition of the events and sequences for timer operation is the same as for the break emulation condition described above. The time displayed in the interval timer box of the sequence menu is dynamically updated, so that you can watch this menu to see the timer operate as your system runs.

One important thing to remember about the timer is that the stop sequence takes precedence over the start sequence. In normal operation the timer starts when the start sequence occurs and counts buss cycles until the stop sequence occurs. If, however, the start and stop sequences occur simultaneously, or if the stop sequence is first, then the timer will never start. One popular use of the timer is to determine the time it takes for some program loop to execute. To start and stop the timer on the same instruction to time the loop, define event A to be the opcode execution of this instruction with a pass count of 1, and define event B to be the opcode execution of this instruction with a pass count of 2. Then set the timer to start on A, and to stop on B.

E. THE TRACE BUFFERS

The HMI-200-68000 contains two 4k (4096) by 72 bit trace buffers. These serve as break trace and trigger trace buffers to give the emulator the history memory which is the feature which makes a logic analyzer so useful. To make full use of the emulator you need to understand the control of these two buffers. The break emulation, trace qualifier and trace trigger conditions are set on the Sequence Menu, then the emulator is started from the command menu. As the emulator executes instructions, the first buffer starts recording any cycles which satisfy the trace qualifier. This buffer works like a 4k window, so that after it fills it will contain the most recent 4k qualified cycles. There are then three different modes of operation of the buffers.

(1) If the trace trigger condition is met and the break emulation condition is not satisfied until after the trace delay number of cycles, the number of cycles specified by the trace delay are counted from the trace trigger and the buffer is held as the trigger trace buffer. The following qualified cycles are recorded in the other buffer which becomes the break trace buffer. Either buffer can be displayed, printed or stored in a file as your computer configuration permits.

(2) If a break emulation condition is met without satisfying the trace trigger condition, emulation is stopped and the active buffer is called the break trace buffer. In this case there is no trigger trace buffer.

(3) If the trigger trace condition is met, but the break emulation condition is met before the trace delay number of cycles, emulation is stopped and the active buffer is called the break trace buffer. As far as the buffers are concerned, this case is the same as case 2 above, and there is no trigger trace buffer.

The trace delay is a four digit hex number (0000 to ffff). This is the number of cycles after the trace trigger that the trigger trace buffer is stopped. With this feature the trigger trace buffer can start at any cycle from 4095 qualified cycles before the trace trigger to 61,439 qualified cycles after the trace trigger. For example, if you wanted to record 100 cycles before the trigger and 3995 cycles after the trigger, you would set the trace delay to hex 0f9b (3995 converted to hex).

The data captured in the buffers can be examined from the Trace Menu. From this menu you can select the buffer to be displayed, and the mode of display - as bus state or as disassembled 68000 instructions. You can scroll through the data, or search for some pattern.

Example: You have a newly modified program that wanders into the weeds. The only thing you know about the problem is that the program executed properly through a section of code at address 1000. You can configure the event sequence for the trace trigger to be the execution of the code at address 1000. You will also set some delay count, say 800H (which would be half of the trace). After executing the code again, you can view the trigger trace from the Trace Menu. What is displayed will be 800H bus cycles of trace after the code at address 1000 was executed as well as 800H bus cycles of trace before. This feature not only gives you a trace history, but a forward or after event trace as well.

There are many powerful uses of this feature. Trigger points have been available in logic analyzers and used primarily for hardware development, but the use of such a feature is easily extended into the software world as well. The broad selection of criteria with which you may define an event should make this feature in an emulator superior to its counterpart in a logic analyzer. For more

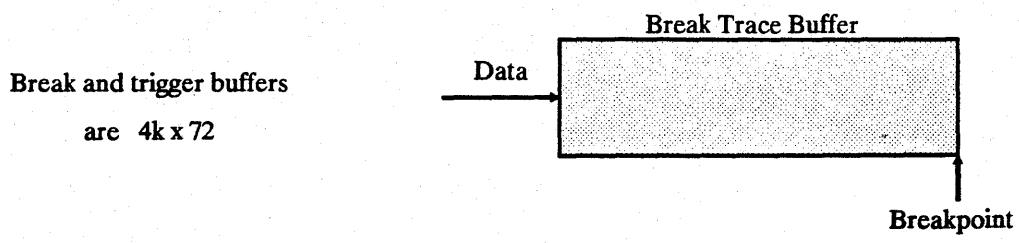
information on how to setup the trigger point parameter, see the section titled Sequence Menu.

The sketches in Fig. II-4 should help you understand the relation between the buffers. The (a) part of the figure shows the case where a breakpoint was encountered before the trace trigger plus delay condition occurred. In this case the buffer contains the 4k history before the breakpoint and is called the break trace buffer. It is possible that the trace trigger might have occurred, but the breakpoint was encountered before the delay was completed. In the (b) part of the figure, the breakpoint was encountered after the trace trigger buffer was latched. Note that the trace trigger buffer is latched after the delay past the trace trigger. This delay can be any number of cycles from 0 to 65535. The two buffers are shown in part (c), but in this case the trace qualifier starts recording in the buffer when event D occurs and stops when event C occurs. The 4k cycles in the buffers will span more than 4k bus cycles because some of the cycles were not qualified. The same qualification applies to both buffers - actually the qualification applies to the buffer which is active in recording cycles. Part (d) shows the recording in both buffers when the recording is qualified by event D. In this case if event D is true at any cycle, then that cycle is entered in the buffer.

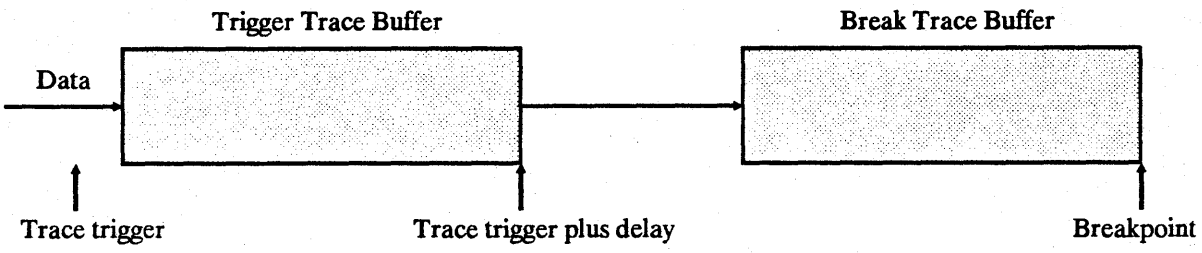
F. FREEZE TRACE

The freeze trace is a manual operation executed from the trace menu by typing CNTRL-QB. This keyboard command stops the trace buffer, but the emulator continues to run. You can then view the trace buffer, and you can modify the event and sequence menus. Although the emulator continues to run in the freeze trace condition, the break and trigger functions are disabled. CNTRL-QG is the resume trace command which causes the emulator to return to normal operation. With this feature you can alter the break emulation or trace trigger conditions without stopping the processor.

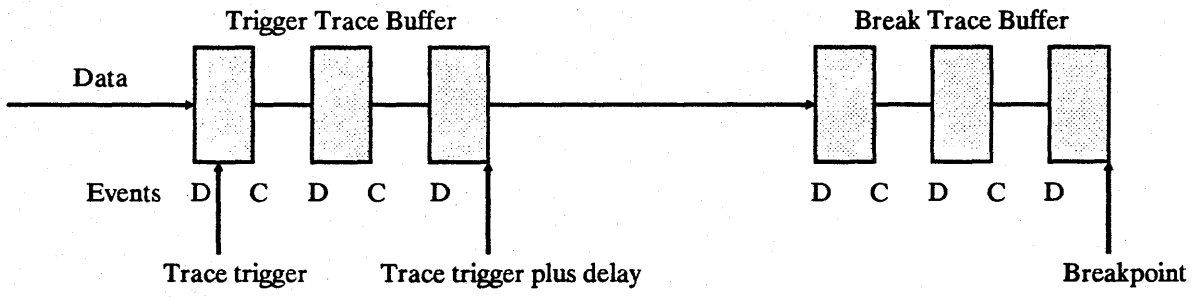
As an example of the use of the freeze trace command, you might have a system that you want to examine when the program reaches address 1000. You could set the break emulation condition for address 1000. Assume that the program does not reach address 1000, and that you do not want to stop the system. You could then freeze the trace and redefine the event and sequence for the break emulation function so that the break would occur at some other address, perhaps 1c30. When you type



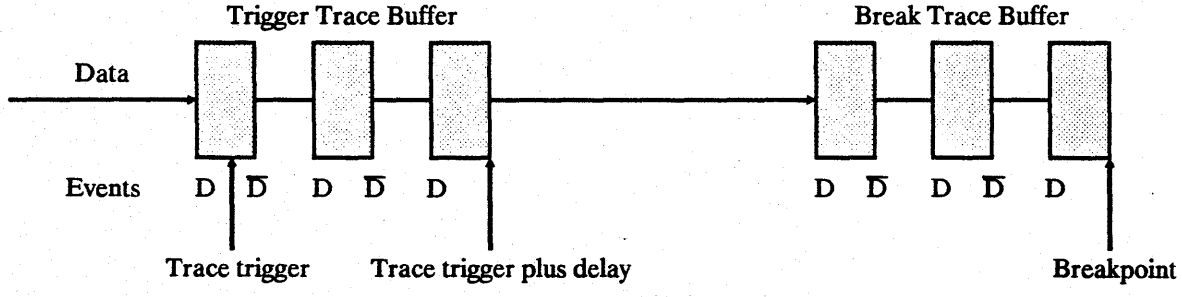
(a) No trace trigger specified, or specified but not reached



(b) Trace trigger and delay specified and reached before breakpoint



(c) Trace buffer with event D turning trace on and event C turning it off



(d) Buffers with trace only during event D ($\overline{D}$ turns trace off)

Fig. II - 4 Trace buffer control

CNTRL-QG to resume trace, the emulator would continue running, but would now be looking for a break at 1c30.

To make full use of the freeze trace function, you need to understand its relation to the trigger buffer. If the trigger buffer is full when the freeze trace command is typed, the freeze trace will display the trigger trace buffer, not the break trace buffer. There are two possible uses of the freeze trace command.

(1) If you want to see the trigger trace buffer without stopping the processor, wait until the status line shows "trace triggered" and then type the freeze command. You can examine the trigger trace buffer, and, if you wish, define a new trigger condition before you resume.

(2) If you want to see the most recent instructions when you issue the freeze command without stopping the processor, toggle the trace trigger on the Sequence Menu to OFF before starting the emulator.

III. INSTALLATION AND EXTERNAL CONNECTIONS

A. UNPACKING AND INSPECTION

Check to see that the following items were received:

1. 68000 Emulator with a protective cap over the 64 pins of the emulator's probe,
2. External trace leads with keyed connector,
3. A.C. power cord,
4. Diskette (if optional software was purchased)

Inspect these items to verify that everything survived shipping.

B. INTERFACING TO THE HOST

The HMI-200 series emulators have two RS-232 ports (marked Main RS-232 and Auxiliary RS-232). Connect a RS-232 cable from the main RS-232 port on the emulator to the RS-232 port on the host computer or terminal and configure switch S1 for your particular use as described in Appendix A.

Turn on the host and the emulator. If the host is a terminal, then the initialization depends on the configuration of switch S1. If S1 is configured for manual baud rate selection, then the emulator will sign- on after the reset button is pressed. If auto-baud is selected, then the emulator will sign on after pressing the reset button and typing a period at the terminal. If the host is a computer and the HMI communication software (ECS68K.EXE) is being used, then switch S1 should be configured for auto-baud. In this case, the communication software will take care of

transmitting the period to the emulator at which time the emulator will sign on to the computer.

The sign-on message will be of the form:

```
HMI SERIES 200 MICROPROCESSOR DEVELOPMENT SYSTEM 68000 UNIT
COPYRIGHT (c) 1987 BY HUNTSVILLE MICROSYSTEMS, INC. Vn.m
```

At this point, software development and debugging can begin without being plugged into the target system by using the emulator's internal clock source (see Configuration Menu).

C. INTERFACING TO TARGET SYSTEM

With the emulator and target system powered off, remove the 68000 CPU chip from the target system and insert the emulator's 68000 probe into the socket in the target system. Pin 1 of the probe is indicated by the "1" on the cover plate of the emulator.

***** WARNING *** Before powering up the emulator or target system, verify that the emulator probe is oriented correctly. If the probe is plugged in backwards, damage may occur to the emulator or target system.**

The emulator should always be powered up after the target system and powered down before the target system. Therefore, power up the target system and then the emulator and select target system clock (see Configuration Menu).

If you wish to trace signals from the target system other than the address, data and status signals which are available from the processor, then connect the external trace leads to the external trace input in the emulator. The leads are color coded (see Event Menu) for easy use. Although the emulator's ground and the target system's ground are connected through the emulator cable, it may still be desirable to connect a trace lead ground near a trace lead signal to reduce any noise on the signal which might be induced from the target system.

*****WARNING*** The external trace leads should be connected to TTL level signals only.**

D. TRIGGERING EXTERNAL LOGIC

Two programmable trigger output signals are available for triggering external logic or test equipment. Both signals are capable of driving a 50 ohm or 75 ohm load.

The first trigger is the pulse output. The pulse is programmed to occur on one of the four events (A, B, C or D) with a positive or negative going pulse (see Sequence Menu). The duration of this signal is approximately the width of the inactive portion of the data strobe (UDS/ and LDS/) signal from the 68000.

The second trigger is the level output. This signal is a low to high transition programmed in the sequence menu to occur on a predefined event sequence. Once triggered, this signal will remain high until it is reset by breaking emulation with the stop command, freeze trace command, reset command or by executing a breakpoint.

E. CONVERTING to 68010 or 68008 PROCESSORS

The HMI-200-68000 unit is normally shipped configured with a 68000 processor, but can be field modified to support the 68010 or the 68008 processors.

To modify the HMI-200-68000 emulator, follow the procedure listed below.

1. Remove the emulator cable cover plate by removing the two phillips head screws from the top of the unit.

2. Carefully remove the emulator cable from the unit by gently rocking the ends of the cable header back and forth until the cables are released.

3. Turn the unit over and remove the four screws from the rubber feet.

4. Slide the cover off of the unit and set it aside.

5. Remove the 68000 processor by turning the screw counter clockwise on the processors' ZIF socket.

6. To configure the unit as a 68010 emulator, simply insert a 68010 processor in the ZIF socket, lock it down and reassemble the unit.

7. To configure the unit as a 68008 emulator, follow these steps:

   a. Remove the DIP chips marked (JP1 68000 U81) and (JP2 68000 U82).

   b. Replace with the DIP chips by inserting (JP1 68008 U81) in socket U81 and (JP2 68008 U82) in socket U82.

   c. Insert the 68008 processor board (optional) in the 68000's ZIF socket and lock it down with the ZIF's screw. Note, the square notch in the processor board should line up with the ZIF's screw.

   d. Place the 68008 adapter board on the end of the emulator cable. Note, Pin 1 for the 68008 socket is now on the opposite end of the header from the 68000's pin 1.

   e. Reassemble the emulator unit.

8. After powering up the emulator in its new configuration, execute the CONFIG command to reinitialize the unit.

IV. SYSTEM OVERVIEW

A. OPERATION OVERVIEW

System operation can be easily performed from six convenient menus. Two of these are used primarily for initial setup or hardware configuration. The other four make up the heart of the HMI-200 series' unique Human Interface system (see "Operation From Menus").

There is also available a command line mode of operation. It may be used instead of the menu mode for many of the emulator functions. In this mode the user enters commands and parameters the same way he would under Debug on an IBM-PC or any other command line oriented computer. For a summary of emulator commands from the command line mode see "Command Line Operation".

There are several mechanisms that are used throughout the system.

1. The "ESC" Key

Depressing the "ESC" key will cause the emulator to switch command modes (command line/menu), unless the user is in the middle of a menu command setup, in which case the "ESC" key will abort the setup. When switching from command line mode to menu mode, the system will display the last menu used. When switching from menu mode to command line mode, the system will clear the screen and display a single dash as the prompt character.

2. The "HOME" Key

Depressing the "Home" key will move the cursor to the top line of the menu screen. This line is dedicated to menu selection and consists of the menu name abbreviations:

CONFIG COMMAND EVENT SEQUENCE TRACE INTERFACE

To move from one selection to the next, use the left and right arrow keys. When the selected menu name has been highlighted, depress the "Return" (or "Enter") key to activate the selection. The requested menu will be displayed. The "Home" key can also be used to switch from command line mode to the menu headers.

3. The Highlighted Areas

The highlighted areas of the menu screens are parameter fields. They signify areas of the screen that may be user modified. If the parameter has predefined selections, depressing the space bar will toggle to the next selection. If no predefined selections are available, then parameters must be entered directly from the keyboard.

4. Cursor Handling

Cursor movement, one character space at a time, may be performed by using the traditional up, down, left and right arrow keys or their duplicate control functions CNTRL-E, CNTRL-X, CNTRL-S and CNTRL-D respectively. There are two additional control functions that allow the cursor to move from anywhere in the current parameter block to the beginning of the next control block. The CNTRL-A function moves the cursor left one block, where the CNTRL-F function moves the cursor right one block.

5. Command Operation

Emulator Commands are designed to give the user a high degree of feedback and flexibility during Command operations. Each command (such as Fill, Read, Search, etc.) will display a "WORKING" message showing the current address block in which the command is operating. The Read and Write commands will display, in real time, the percentage of the file which has been transferred. Any command can be aborted before it has completed its task simply by entering a carriage return. At this time, an

"ABORT" message will be displayed showing the last address used by the command.

B. MEMORY MAP OVERVIEW

The Memory Map is used to organize the emulator memory to overlay target system memory. Emulator memory may be divided into four partitions. Each partition may start on any 8k word boundary in the 68000 address range. Each partition must be designated by type as either Supervisor Program (SP), Supervisor Data (SD), User Program (UP), User Data (UD) or any combination of these. A partition may have more than one type associated with it, however, no type may be used in more than one partition. Thus, if SP and SD are combined into one partition, the maximum number of partitions possible is three. The emulator's human interface will not allow the user to create a memory configuration that is not consistent with these restrictions.

Once configured, each partition is subdivided into 8K word segments. Each of these segments may be configured to be target memory (T) or emulator memory (E). Emulator memory segments (E) may be further configured as read/write (RW) or read only (RO) for write enabled or write protected respectively.

For a better understanding of Memory Mapping, see the Configuration Menu section.

C. EVENTS AND SEQUENCING OVERVIEW

There are four events (A ,B, C and D) that are used for break and trigger conditions. Each event is set as a bit pattern, including don't care conditions, of the following items:

* Address	24 bits
* Data	16 bits
* Status	16 bits
Read/Write (RW)	1 bit
Function codes (fc0, fc1 and fc2)	3 bits
Interrupt lines (ipl0, ipl1 and ipl2)	3 bits
Valid Memory Address (vma)	1 bit
Address error (AERR)	1 bit
Bus error (BERR)	1 bit
Valid Peripheral Address (vpa)	1 bit
Opcode Execution/Fetch (ox,f)	1 bit
(Reserved for future use)	4 bits
* External Trace Lines	16 bits
Total	72 bits

Each of the four events also has a 1 to 65535 count pass counter.

An event is activated when the selected parameters all occur simultaneously, the number of times specified by the pass counter. The user may specify up to four unique events (A, B, C and D).

Once the desired events have been specified, they may be sequenced (using AND, OR, THEN and parenthesis) to define a complex set of criteria that may be used to:

Break Emulation
Trigger Trace
Trigger External Level
Trigger Timer Start/Stop

A single event may be selected to generate the external pulse signal. Events may also be selected to serve as trace qualifications. The Event

and Sequencing Diagram in Fig. IV-1 shows events and their relationship to trigger and break points. As this figure shows, the external pulse can have either polarity, and corresponds to any one of the four defined events. The external pulse is not effected by the pass counters or the sequences defined on the sequence menu. The events, with their pass

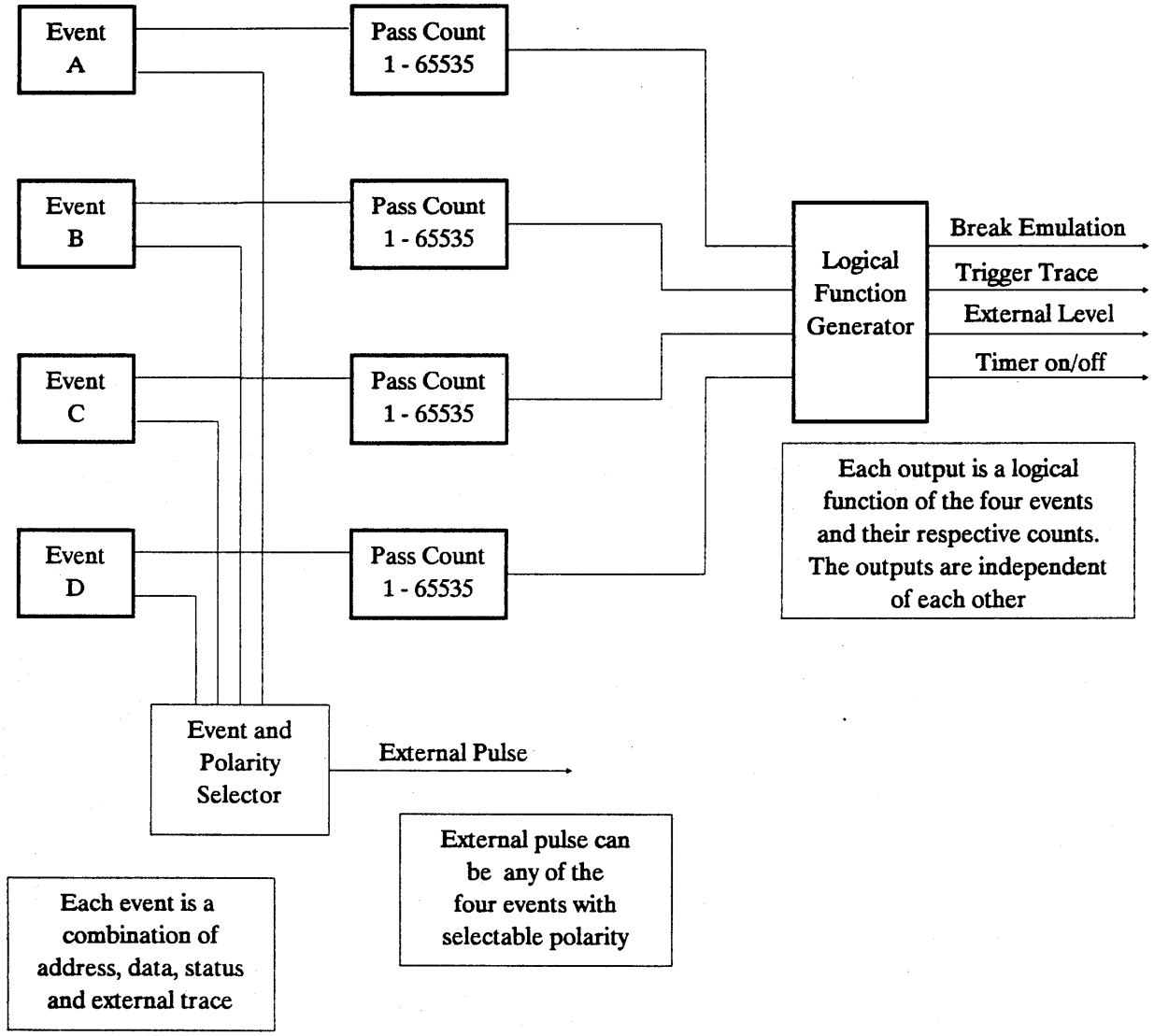


Fig. IV-1 Event and Sequencing Diagram

counts, are combined as you specify at the sequence menu to determine the break point, trace trigger, external level and timer on/off control.

D. TRACE OVERVIEW

The system has two 4k by 72 bit trace buffers. One is a traditional trace or execution history buffer called the "Break Trace". At a breakpoint this buffer will contain up to 4k bus cycles of execution history.

The other trace buffer is a trigger point trace buffer called "Trigger Trace". This trace buffer is handled just like the trace buffer in a logic analyzer. The trigger point can be configured with an event sequence much like a breakpoint is configured.

The trace trigger feature also has the usual delay counter. With the delay counter, the trace buffer is frozen a specified number of cycles after the trigger point occurs. In effect, this gives you a buffer which contains information before and after the trigger point. The trigger trace buffer is a window of executed code around the trigger point. The window effect, sometimes known as forward trace, is accomplished by configuring the "delay count" to specify the number of cycles, after the trigger point, which should be recorded. When the delay counter has counted down, the message "trace triggered" will be displayed, the trigger trace buffer will be preserved, and the emulator will begin recording in the break trace buffer until emulation is terminated. If emulation is terminated before the delay counter has counted down, then all of the recorded data will be in the break trace buffer.

The user may also establish trace qualification criteria, so that the trace buffer does not fill continually. One method is to specify a start trace on event D and a stop trace on event C. The other is to use event D to define a condition that would restrict trace except where the condition exists.

V. OPERATION FROM MENUS

This section contains detailed information on using the system from the six menus. The HMI-200 human interface system was designed to allow operation of the emulator with a minimum of instruction. Going through this section, using an emulator to experiment with each option or command as it is presented, should provide the user with a good working knowledge of the system. (If an emulator is not available, then refer to the drawings of the menu screens provided, as frequent references are made to items on these screens). The user will discover other subtle, but powerful, capabilities of the emulator through continued use and as specific needs arise.

A. CONFIGURATION MENU

The configuration menu, as shown in Fig. V-1, is used to establish the memory map and select hardware specific parameters.

1. Memory Mapping

Many users will simply configure all of the emulation memory into one block or partition containing all four of the following combine types:

SD - Supervisor Data
SP - Supervisor Program
UD - User Data
UP - User Program

In the upper left section of the menu is a one column field for "Block Number", a four column field for "Combine Type" (one column each for UD, UP, SD, SP respectively), and a one column field each for "Base Address", "Ending Address" and "Block Size".

1. Memory Mapping

```
.Idling.....................................................................
```

```
  Memory Map -      Unallocated Memory = 0000K      Processor: 68000
  Block    Combine      Base      Ending    Block  Clock      Emulator
  Number   Type         Address   Address   Size   Target System (BR,IPL0-2,
    1      UP UD SP SD  000000H   03FFFFH   0256K  BGACK,BERR):  Enabled
    2      -- -- -- --  000000H   000000H   0000K  Allow bus arbitration:
    3      -- -- -- --  000000H   000000H   0000K   Only During Emulation
    4      -- -- -- --  000000H   000000H   0000K  DTACK source is
BLK 1   S/U = SUPV/USER  P/D = PROG/DATA             From the Emulator
```

```
      E = Emulator, T = Target System, RO = Read Only, RW = Read/Write
000000 E RW  02C000 E RW
004000 E RW  030000 E RW
008000 E RW  034000 E RW
00C000 E RW  038000 E RW
010000 E RW  03C000 E RW
014000 E RW
018000 E RW
01C000 E RW
020000 E RW
024000 E RW
028000 E RW
```

Fig. V-1 Configuration Menu Screen

Combine Type

The system will not allow any of the parameters to be entered for a block, until at least one combine type is selected. To select a combine type for a given block, move the cursor to the highlighted field where the block number's row meets the selected combine types column and depress the space bar. That space should be replaced by the two letter abbreviation of the combine type.

If, for example, block number one and combine type UD were selected, the left most field of the combine type would contain the abbreviation "UD". Since the system will not allow a combine type to be used in more than one block, adding the UD combine type to block number two would remove it from block one.

Base Address

The base address is specified as the beginning address of a memory block.

Ending Address

The ending address field is used to specify the upper address for a memory block. Once the ending address has been entered, the system will compute and display the block size.

Block Size

A block's address boundaries may also be expressed by the base address and a block size. Any time block size is entered, the system will compute and display an ending address.

GENERAL NOTE

The user is required to specify blocks that begin on even 8k word address boundaries. The total memory allocated in defined blocks may not exceed total emulation memory. Unallocated memory is displayed at the top of the menu for quick reference. If the addresses or block size specified exceed the amount of unallocated memory or does not adhere to the boundary requirement, the system will change the entry to a value that will meet both criteria.

Block Segments

Each block (or partition) defined is divided into 8K word segments. The segments for a given block can be displayed in the lower half of the menu. There is a selection on the menu (just above the dividing line between upper and lower screen halves) called "BLK" with a number field to the right. This parameter allows the user to select the block for which segments will be displayed.

Select Block

To change the block number, place the cursor on the number field and depress the space bar. When the selected block appears, the segments for that block will be listed by beginning address in the bottom half of the screen. To move from the block number field to the segment fields, press the down arrow key.

Segment Parameters

Beside each segment will be displayed a memory source field. A "T" in that field directs the system to use the target system's memory and an "E" designates the use of emulator memory. All segments marked as emulator (E) memory will have an additional two character memory protection field. "RW" is used to represent read/write memory and "RO" represents read only (or write protected) memory.

Segment Parameter Selection

To change any of the fields, place the cursor on the selected parameter and depress the space bar. This will toggle the parameter from one state to the other. As an example, with the cursor on an "E", depressing the space bar will change it to "T" and the memory protection characters will disappear.

2. Processor Control Parameters

Several control signals need to be configured prior to emulation. These can be found in the upper right corner of the menu.

68000 Clock Source

The clock for the 68000 processor can be derived from the emulator's on board 10MHz oscillator by selecting "Emulator" in the Clock Field. The emulator clock is generally used for stand-alone operation (ie. when the emulator is used without being plugged into a target system). When plugged into a target system, it is generally advisable to run from the target system clock. Do this by selecting "Target System" in the Clock Field.

Control Signals

The control signals (BR, IPL0, IPL1, IPL2, BGACK and BERR) can be Enabled or Disabled. "Enabled" means that these signals are active during emulation. "Disabled" means that these signals are gated off to

the processor always. These signals should be Disabled when the emulator is not connected to a target system.

Allow Bus Arbitration

Bus Arbitration can be selected to be enabled "Only During Emulation" or "Always". If it is enabled "Only During Emulation", then bus arbitration can only be activated when real time emulation is in process (see the GO command in the Command Menu) and the Target System control signals have been enabled. If bus arbitration is selected to be active "Always", then it can occur at any point during emulator operation even if the Target System control signals have been disabled. This is particularly valuable for systems which use bus arbitration to perform refresh to dynamic memory. If bus arbitration is enabled Always, then a very active bus arbitration circuit may cause some noticeable speed difference in the operation of the emulator.

Data Acknowledge (DTACK)

The 68000 processor requires a DTACK signal to acknowledge all memory operations. This signal can come from the "Emulator's DTACK" or the "Target System's DTACK". When running the emulator stand alone without a Target System, DTACK should be selected to come "from the Emulator". When plugged into a Target System, DTACK can be selected to come "from the Target System" or "according to Memory Map" in which case DTACK will come from the Emulator when the 68000 makes a memory access to memory enabled in the Emulator, or from the Target System when the 68000 makes a memory access to memory mapped to the Target System.

B. COMMAND MENU

Emulation commands can be executed from the Command Menu, Fig. V-2. The current value of all of the registers, flags and disassembled current instruction are always present at the top of this menu. Each command can be selected by typing its one or two character abbreviation. The current values for the parameters of the selected command will be displayed in their respective fields.

```
.Idling.....Insert On.....................................................

D0 FFFFFFFF     D1 AAAAFFFF     D2 FFFDFFFF     D3 FFFFFFFF    SSP 04704002
D4 FFFFFFFF     D5 FFFFFFFF     D6 FFFFFFFF     D7 FFFFFFFF    USP FFFFFFFF
A0 31FC0000     A1 060442C0     A2 31FC5555     A3 06044EF8
A4 002031FC     A5 AAAA0604     A6 4EF80020     A7 04704002
   SR  2709  - NT S 17 NX NG NZ NV CY
 PC  00000120
        ORI.W #FF00.(A2)+
 X register change  D Dump   L List code  M Move     RD Read file
 SP Stop processor  F Fill   A Assemble   C Compare  W Write file  ^U Insert
 SS Single step     E Enter  G Go         SR Search  RS Reset          mode
   Dump  from  000100H  to  000200H  WORD      PAGED

        0    2    4    6    8    A    C    E
000100 0000 FF00 0000 FF02 0000 FF00 0000 FF00  ................
000110 00FF FFFF 00FE FFFB 00FF FFFF 00FF FFFF  ................
000120 005A FF00 0000 FF00 0008 FF00 0000 FF03  .Z..............
000130 00FF FFFF 00FF FF7E 00FF FFFF 00FF FFFF  .......~........
000140 0040 FF00 0000 FF00 0008 FF00 0000 FF04  .@..............
000150 00FF FFFF 005F FFDF 00FF FFFF 00FF FFFF  ....._..........
000160 0008 FF00 0000 FF02 0002 FF00 0000 FF11  ................
        Hit <CR> for more. "." <CR> to abort.
```

Fig. V-2 Command Menu Screen

There are two distinct types of parameter fields:

The Toggle Field - which will display the next possible option each time the space bar is depressed. Continuing to depress the space bar will eventually cycle through all of the options and begin again with the first option.

The Key Entered Field - which may be modified by overtyping the existing value with another value. The method in which characters are entered in this field may be altered by typing a CNTRL U to enable or disable INSERT mode. When INSERT mode is turned on, characters entered will shift the other characters, which lie on or to the left of the cursor, left one position. If the cursor had just landed on the right most character of this field, then the field will be zeroed when any characters are entered. This INSERT mode was designed to allow commands to be entered with the same key strokes as would occur by entering the same

command from the Command Line Mode. Therefore, a space or comma can be used to move to the next field.

When INSERT mode is turned off, then any character entered will simply overwrite the character in the current cursor position and the cursor will shift to the right. The parameter fields may be modified in any order, at any time until the command is executed or aborted.

Once all fields are properly configured, the command can be executed by depressing the return key. Any command can be aborted before it is executed by depressing the "ESC" key.

Commands will be shown in the following form:

Dump from 000000H to 000000H word (Supv. Prog. Memory) PAGED

- The single underline denotes a Key Entered Field.

- The double underline represents a Toggle Field.

- The parenthesis show a field that is dependent on the systems configuration. This field may not always appear with the command.

Since all parameter values are saved, the existing value will either be the initial value or the value of the last modification.

1. A (Assemble) Command

This command allows assembly language mnemonics to be entered into memory starting at the specified address (In-Line Assembly). Once the Assembly command is configured for the desired starting address, type return and the address will be displayed on the bottom of the screen with the cursor to the right of the address. At this point, 68000 assembly language instructions can be entered. After each entry, the emulator displays the next address and prompts for the next instruction. An error in the entered instruction will be flagged by displaying a question mark (?) and redisplaying the current address. A period (.) followed by a carriage return terminates the entry sequence and the command. The format for the Assemble command menu line is:

Assemble at Addr

where: Addr - is the hexadecimal starting address

For a complete listing of assembly language mnemonics and corresponding op-codes, see Appendix B.

2. C (Compare) Command

The Compare Command will compare one block of memory to another. The first word in the first block will be compared to the first word in the second block, and so on. Any non-comparisons will be displayed on the bottom of the screen showing the address of the first block with its data value and the address of the second block with its data value. If the two blocks match, then nothing will be displayed.

The format for the Compare command menu line is:

Compare (Data Type) from Baddr1 to Eaddr1 to (Data Type) at Baddr2

where:

Data Type - represents the block containing:

"Supv. Prog." = Supervisor Program Memory
"Supv. Data " = Supervisor Data Memory
"User Prog. " = User Program Memoru
"User Data " = User Data Memory

Note: The partitions (blocks) are defined in the Configuration Menu.

Baddr1- is the hexadecimal beginning address of the first compare block

Eaddr1 - is the hexadecimal ending address of the first compare block

Baddr2 - is the hexadecimal beginning address of the second compare block

The size of the first block sets the number of words to be compared.

For each word in block one that does not match the corresponding word in block two, a line is added to the display section of the screen.

Thisdisplay line has the format:

Addr1 Value1 Addr2 Value2

where:

Addr1 - is the address of the word in block one

Value1 - is the contents of addr1

Addr2 - is the address of the word in block two

Value2 - is the contents of addr2

3. D (Dump) Command

The Dump Command will display the contents of memory in the specified address range showing the hexadecimal data with its ASCII equivalent in the display section of the screen.

The Format for the Dump Command menu line is:

Dump from Baddr to Eaddr Mode (Data Type) Display

where:

Baddr - is the hexadecimal beginning address of the block of memory to be displayed

Eaddr - is the hexadecimal ending address of the block of memory to be displayed

Mode - "Word" = Dump to be displayed as word values

"Byte" = Dump to be displayed as byte values

Data Type - represents the block containing:

"Supv. Prog. Memory"
"Supv. Data Memory"
"User Prog. Memory"
"User Data Memory"

Note: The partitions (blocks) are defined in the Configuration Menu.

Display - "Paged" = Dump to be displayed one Page at a time until Eaddr is reached or a period is typed.

"Cont" = Dump to be displayed Continuously until Eaddr is reached or display is terminated.

The two formats for the Dump output are shown in Fig. V-3. The first is in the "Word" mode and the second shows the "Byte" mode.

```
Word Mode Dump:

001000 1234 1234 1234 1234 1234 1234 1234 1234  .4.4.4.4.4.4.4.4
001010 1234 1234 1234 1234 1234 1234 1234 1234  .4.4.4.4.4.4.4.4
001020 1234 1234 1234 1234 1234 1234 1234 1234  .4.4.4.4.4.4.4.4
001030 1234 1234 1234 1234 1234 1234 1234 1234  .4.4.4.4.4.4.4.4
001040 1234 1234 1234 1234 1234 1234 1234 1234  .4.4.4.4.4.4.4.4
001050 1234 1234 1234 1234 1234 1234 1234 1234  .4.4.4.4.4.4.4.4
001060 1234 1234 1234 1234 1234 1234 1234 1234  .4.4.4.4.4.4.4.4
001070 1234 1234 1234 1234 1234 1234 1234 1234  .4.4.4.4.4.4.4.4


Byte Mode Dump:

001000 12 34 12 34 12 34 12 34 12 34 12 34 12 34 12 34  .4.4.4.4.4.4.4.4
001010 12 34 12 34 12 34 12 34 12 34 12 34 12 34 12 34  .4.4.4.4.4.4.4.4
001020 12 34 12 34 12 34 12 34 12 34 12 34 12 34 12 34  .4.4.4.4.4.4.4.4
001030 12 34 12 34 12 34 12 34 12 34 12 34 12 34 12 34  .4.4.4.4.4.4.4.4
001040 12 34 12 34 12 34 12 34 12 34 12 34 12 34 12 34  .4.4.4.4.4.4.4.4
001050 12 34 12 34 12 34 12 34 12 34 12 34 12 34 12 34  .4.4.4.4.4.4.4.4
001060 12 34 12 34 12 34 12 34 12 34 12 34 12 34 12 34  .4.4.4.4.4.4.4.4
001070 12 34 12 34 12 34 12 34 12 34 12 34 12 34 12 34  .4.4.4.4.4.4.4.4
```

Fig. V - 3 Sample Dump Output

4. E (Enter) Command

The Enter Command is used to examine memory at the specified location with the option to change its contents. After executing the Enter Command, the selected address will appear at the bottom of the screen with the current data. The data can be altered by entering new data and pressing a carriage return, or it can remain unchanged by just typing a carriage return. In either case the next address will be displayed with its data. This command can be terminated by typing a period (.) in the data field followed by a carriage return.

The Format for the Enter Command menu line is:

Enter word at Addr (Data Type)

where:

Addr - is the hexadecimal address of the data value to be displayed for possible substitution

Data Type - represents the block containing:

"Supv. Prog. Memory"
"Supv. Data Memory"
"User Prog. Memory"
"User Data Memory"

Note: The partitions (blocks) are defined in the Configuration Menu.

5. F (Fill) Command

The Fill Command is used to fill memory with a constant. The given data value will be written to each memory location within the specified range.

The format for the Fill Command menu line is:

Fill from Baddr to Eaddr Mode (Data Type) Value Val

Where:

Baddr - is the hexadecimal beginning address of the memory block to be filled

Eaddr - is the hexadecimal ending address of the memory block to be filled

Mode - "WORD" = defines constant as a word value

"BYTE" = defines constant as a byte value

Data Type - represents the block containing:

"Supv. Prog. Memory"
"Supv. Data Memory"
"User Prog. Memory"
"User Data Memory"

Note: The partitions (blocks) are defined in the Configuration Menu.

Val - Constant value to be used to fill the memory block

6. G (Go) Command

The Go Command is used to start real-time emulation where any break conditions which occur are due to the configuration of the events from the Event Menu and the breakpoint sequence as defined in the Sequence Menu.

The format of the Go command is:

Go from Saddr

Where:

Saddr - is the hexadecimal address where execution (real-time emulation) will begin

7. L (List Code) Command

This command is used to disassemble the program memory data into assembly language mnemonics. All data and addresses are displayed in hexadecimal.

The format for the List Command menu line is:

List from Baddr to Eaddr (Data Type) Display

Where:

Baddr - is the hexadecimal beginning address of the memory data to be disassembled

Eaddr - is the hexadecimal ending address of the memory to be disassembled

Data Type - represents the block containing:

"Supv. Prog. Memory"
"Supv. Data Memory"
"User Prog. Memory"
"User Data Memory"

Display - "Paged" = List to be displayed one Page at a time until Eaddr is reached or a period is typed.

"Cont" = List to be displayed Continuously until Eaddr is reached or display is terminated.

Note: The partitions (blocks) are defined in the Configuration Menu.

8. M (Move) Command

The Move Command is used to copy the contents of one block to the contents of another block (non- destructive move). Block one is defined with its starting and ending addresses being the first two address fields displayed, and block two is defined with its starting address being the third address field. The contents of the first memory location of block one is copied to the first memory location of block two, and so on.

The format of the Move command menu line is:

Move (Data Type) from Baddr1 to Eaddr1 to (Data Type) at Baddr2

where:

Data Type - represents the block containing:

"Supv. Prog." = Supervisor Program Memory
"Supv. Data " = Supervisor Data Memory
"User Prog." = User Program Memory
"User Data " = User Data Memory

Note: The partitions (blocks) are defined in the Configuration Menu.

Baddr1- is the hexadecimal beginning address of the data "Source" block

Eaddr1 - is the hexadecimal ending address of the data "Source" block

Baddr2 - is the hexadecimal beginning address of the data "Destination" block

The size of the "Source" block defines the number of bytes to be copied into the "Destination" block.

The Move command menu line will remain on the screen until the copy is complete.

9. RD (Read) Command

This command is used to transfer a hexadecimal file from a host computer to the emulation memory. The file name and RS232 port (main or auxiliary) can be specified with this command. When downloading a file using the Huntsville Microsystems, Inc. communication program (ECS68K), the ASCII file is converted to binary before being transferred to the emulator. This essentially doubles the speed of the transfer over the normal ASCII transfer. When downloading from a host which doesn't use ECS68K, a standard ASCII transfer will occur.

The file formats accepted by the ECS68K software are Motorola S-Record, Textronix Extended Tex Hex, Microtec Research Absolute and Intel Hex. The file formats accepted by the emulator without going through the ECS68K software are the same except for the Microtec Research Absolute. This does not pose a restriction since the Microtec Research software is capable of generating several of the accepted formats.

Once the Read Command has been executed, it can be terminated by typing a CNTRL-X.

The format for the Read command is selected by toggling the Port Type parameter ("main" or "aux"). The two formats for the Read command menu line are:

Read (Data Type) with offset Val from Main port file Filename

Read (Data Type) with offset Val from Aux port

where:

Data Type - represents the block containing:

"Supv. Prog." = Supervisor Program Memory
"Supv. Data " = Supervisor Data Memory
"User Prog." = User Program Memory
"User Data " = User Data Memory

Note: The partitions (blocks) are defined in the Configuration Menu.

Val- is a hexadecimal value representing the address offset

Filename - is a 14 character file name, valid to the host computer, that designates the file to be down loaded to the emulator. If no extension is specified, the default .ABS will be used.

Note: The down load speed is directly proportional to the baud rate of the RS232 interface.

10. RS (Reset) Command

This command will issue a reset to the 68000 processor. All registers will be updated to the value which they contain upon a processor reset. The Program Counter is initialized to the contents of memory locations 4H and 6H and the Stack Pointer is initialized to the contents of memory locations 0H and 2H.

11. SP (Stop) Command

This command is used to stop emulation in order to regain control of the 68000 processor. Upon entering this command, emulation will break, all registers will be updated to their current values, and the message "PROCESSOR IS IDLING" will be displayed at the bottom of the screen.

12. SR (Search) Command

This command can be used to search memory between two addresses for a particular data pattern. When a match is found, the address is displayed. To search for the next occurrence, hit the Carriage Return key. To terminate this command, type a period and then return. If no match is found, then nothing is displayed.

The format for the Search command menu line is:

Search (Data Type) from Baddr to Eaddr value Mode Val

where:

Data Type - represents the block containing:

"Supv. Prog. Memory"
"Supv. Data Memory"
"User Prog. Memory"
"User Data Memory"

Note: The partitions (blocks) are defined in the Configuration Menu.

Baddr- is the hexadecimal beginning address of the block of memory to be searched

Eaddr - is the hexadecimal ending address of the block of memory to be searched

Mode - "Word" defines the search data value to be a word

"Byte" defines the search data value to be a byte

Val - is the data value to be searched for and will be either a byte or word value according to Mode

13. SS (Single Step) Command

This command is used to execute one instruction at a time. Each time a step is performed, the registers are updated and the address and instruction are displayed at the bottom of the screen. Therefore, multiple steps will display a history of program flow on the bottom half of the screen.

There are three options for operating the Single Step Command. The first option is to single step every time a carriage return is entered. To terminate this mode, type a period and then a carriage return. The second option is to single step a specified number of times. The third option is to single step continuously until a carriage return is entered.

The option can be selected by toggling the first parameter. The three formats for the Single Step command menu line are:

Single Step once and wait for <CR>

Single Step with iteration count of Count

Single Step continuously until <CR>

14. W (Write) Command

This command is used to transfer files from the emulation memory to a host computer. The size of the block being transferred is specified by the two address fields activated by this command. When using the Huntsville Microsystems, Inc. communication software (ECS68K), the block will be written as a Motorola S-record file on disk under the specified file name.

The format for the Write command is selected by toggling the Port Type parameter ("Main" or Aux").

The two formats for the Write command menu line are:

Write (Data Type) from Baddr to Eaddr to Main port file Filename

Write (Data Type) from Baddr to Eaddr to Aux port

where:

Data Type - represents the block containing:

"Supv. Prog." = Supervisor Program Memory
"Supv. Data " = Supervisor Data Memory
"User Prog." = User Program Memory
"User Data " = User Data Memory

Note: The partitions (blocks) are defined in the Configuration Menu.

Baddr- is the hexadecimal beginning address of code to be written

Eaddr - is the hexadecimal ending address of code to be written

Filename -is a 14 character filename, valid to the host computer, that designates the file to be uploaded (written) to the host computer. If no extension is specified, the default .ABS will be used.

15. X (Examine) Command

This command will allow the user to examine and change any register. After entering the X, the user will be prompted with a register field. After selecting the desired register and typing a carriage return, the selected register with its current value will be displayed at the bottom of the screen. At this point, the register contents can be modified or left unchanged simply by entering a carriage return.

If an invalid register name is selected, the system displays a question mark ("?") and terminates the command.

The format for the Examine Command is

Examine Register Reg

where:

Reg - is the name of the processor register to be examined/modified.

C. EVENT MENU

The Event menu, as shown in Fig. V-4, is used to establish the parameters that define an event. When emulation is initiated with the Go Command, the system will utilize the configuration of the four events (A, B, C and D). The fields that make up each of the four events are described below.

```
.Idling.....................................................................

      Event Definition Menu     Event A

     ADDRESS XXXXXXH  XXXXXXXX XXXXXXXX XXXXXXX0B
       DATA   XXXXH             XXXXXXXX XXXXXXXXB
    EXTERNAL   XXXXH
      STATUS   .XXXH

                          Pass Count [ 001AH ]

  ^Y to clear this event to "don't cares"
  ^O to clear all events to "don't cares"
```

	bits status		bits extern.	wire/tip color
00	X	R/W'	X ext0	B/BLK
01	X	FC0	X ext1	B/BRN
02	X	FC1	X ext2	B/RED
03	X	FC2	X ext3	B/ORNG
04	X	IPL0'	X ext4	B/YELW
05	X	IPL1'	X ext5	B/GRN
06	X	IPL2'	X ext6	B/BLUE
07	X	VMA'	X ext7	B/PURP
08	X	VPA'	X ext8	W/BLK
09	X	AERR'	X ext9	W/BRN
10	X	BERR'	X ext10	W/RED
11	X	OF'/X	X ext11	W/ORNG
12	.		X ext12	W/YELW
13	.		X ext13	W/GRN
14	.		X ext14	W/BLUE
15	.		X ext15	W/PURP

W/ = White B/ = Blue
Black/Black = Ground

Fig. V-4 Event Menu Screen

1. Address and Data

A unique definition for each of the four events can be displayed by toggling the event field at the top center of the menu. The four configuration parameters for each event are the Address, Data, External bits, and

Status. Each of these parameters can be configured as hexadecimal values in the hex fields or as binary values in the binary fields. The binary fields for the Address and Data are just to the right of the hex fields. The binary fields for the External and Status parameters are in the vertical bit definitions on the right side of the screen.

Any hexadecimal digit (0-F) and X are allowable characters for any hex field, where X represents the don't-care. Allowable binary values are 0, 1 or X for any binary field.

2. Status

A prime (') after a character in the status definition implies active low. That is, r/w' means that a "1" in this bit field will select a read operation, and a "0" in this field will select a write operation and an "x" in this field will select either. The only status bit which requires special attention is the op-code fetch/op-code execution (of'/x) bit. If this bit is set to a "0", then an op-code fetch will trigger the event. A "1" in this bit field will prevent the event from triggering until the actual execution of the selected instruction occurs.

3. Pass Count

Each event has a Pass Count field. Therefore, each event can be configured to trigger from 1 to 65535 times before activating the selected function in the Sequence Menu.

4. External Trace Bits

Each of the external trace bits have a wire color and tip color associated with it, making identification and correlation of the menu and the physical probes simple.

5. Common Trigger Conditions

Some of the trigger conditions that can be established with the status bits are not immediately obvious. The following is a list of some of the trigger conditions commonly used. It is by no means a complete list due to the many status bit combinations possible.

Conditions	Field	Value
Read only	R/W'	1
Write only	R/W'	0
Read or Write	R/W'	X
OPCODE Execution	OF'/X	1
OPCODE Fetch	OF'/X	0
Specific Interrupt (0-7)	ipl0, ipl1, ipl2	000-111
Interrupt Acknowledge	fc0, fc1, fc2	111
Access User Data	fc0, fc1, fc2	001
Access User Program	fc0, fc1, fc2	010
Access Supervisor Data	fc0, fc1, fc2	101
Access Supervisor Program	fc0, fc1, fc2	110
Any Memory Access	R/W', OF'/X	XX

D. SEQUENCE MENU

The Sequence Menu is where an event or a sequence of events can be specified to trigger various emulator operations. The screen for this menu is shown in Fig. V-5. It is also used to configure the various functions of the emulator which are to be enabled. Some of the functions controlled by this menu are breakpoint, trace trigger, interval timer and output signals.

When assigning a sequence to one of the functions, the user may use one of several methods. He may simply type in the event or event sequence from the keyboard. He may select one of the 14 predefined sequences and enter the sequence number. The user also has the option of replac-

```
.Idling.....................................................................
```

Sequence Menu		Predefined Sequences
Break Emulation ON A OR B Restart 00H time(s) XX for continuous Wait for <CR> before resuming ? NO	Trace Trigger OFF A THEN B Trace Delay 0000H *Trace Qualifier NONE	1. A 2. A OR B 3. A OR B OR C 4. A OR B OR C OR D 5. A AND (B OR C) 6. A AND B 7. A AND B AND C 8. A OR (B THEN C) 9. B AND (C THEN D) 10. A OR B OR C THEN D 11. A THEN B 12. B THEN C THEN D 13. A OR (B THEN C) 14. A THEN B WITHOUT C * 11. A THEN B
Interval Timer - Start A ON - Stop B OR C Time = 001.434 mSec.		
External Level Out ON C		
* NEGATIVE External Pulse On Event D		
Pass Counts A=F9D2/0001H B=F9D1/0021H C=DC40/0001H D=DC40/0001H	* Pass Counts Do Not Apply	Functions AND & OR + THEN T ()

Fig. V-5 Sequence Menu Screen

ing any of the predefined sequences with a sequence that he more commonly uses. Since all setup information is saved in battery back-up memory, these new sequences will be maintained for future use. Event sequences may contain AND, OR, THEN, parenthesis and the four events A, B, C and D.

When emulation is initiated with the Go Command, then the system will respond to the events as defined in the Event Menu and will trigger the appropriate operation as defined in this menu.

There are four operations which can occur as defined by a sequence of up to four events. These are Break Emulation, Trace Trigger, Interval Timer and External Level Out. Any field which requires a sequence can be configured by entering a sequence or the number of a predefined sequence in the appropriate field followed by a carriage return.

Remember that as a sequence is occurring during emulation, that pass counts apply to each event. That is, if a sequence of A THEN B is selected, and Event A has a pass count of 5 and Event B has a pass count

of 3, then the actual operation will be: A occurs 5 times and then B occurs 3 times before the conditions of the selected operation are met.

1. Break Emulation

The Break Emulation parameter can be configured to handle a breakpoint in several ways. The traditional breakpoint can be configured by toggling the Break Emulator field to "ON" and entering the desired sequence in the sequence field just below. An automatic Restart function can be configured by entering a two digit Hexadecimal value in the Restart field. This function will display the registers and resume emulation the number of times specified or forever if "xx" is placed in the restart field. The field titled "Wait for <CR> before resuming?" acts as a switch. In the "OFF" position it allows automatic restarts, but in the "ON" position the user must depress the return key each time to resume emulation. In either case the number of restarts allowed is controlled by the Restart field.

2. Trace Trigger

The Trigger Trace buffer can be activated by toggling the Trace Trigger field "ON" and selecting the desired sequence for triggering. The triggering of this trace buffer can be delayed after the conditions of the sequence are met by inserting a count of 0 to FFFFH Bus cycles in the Trace Delay Counter field. Trace data is always recorded into the Break Trace Buffer. When the Trace Trigger sequence is executed and the delay count has expired, then the current trace buffer is tagged as the Trigger Trace Buffer and trace recording continues in the Break Trace Buffer.

3. Trace Qualification

A trace qualifier is available for both trace buffers (Trigger Trace and Break Trace). The qualifier can be programmed to trace everything that occurs from Event D until Event C. It can also be configured to record

only occurrences during Event D. The Trace Qualifier field will toggle through these two options and "None". The pass counts of events C and D do not apply to trace qualification as the asterisk (*) and note on the sequence menu reminds you.

4. Interval Timer

The timer can be configured to start on a sequence of events and to stop timing after another sequence of events. The time elapsed between the Start and Stop sequences will be displayed below the Stop sequence. The display will update real-time and will stop when emulation is terminated or when the Stop sequence is reached. The timer has a micro second resolution and a maximum time of 71 minutes. This function can be enabled or disabled by using the "ON"/"OFF" toggle field.

5. External Level Out

The BNC for external level will generate an active high (low to high) signal when the sequence is reached. The External Level Out parameter can be configured with an event sequence and has a field to toggle it "ON" or "OFF".

6. External Pulse Out

The External Pulse parameter can be configured to be a "POSITIVE" or "NEGATIVE" pulse by toggling the first field. The second field is for event selection. It will toggle through the four events. The BNC for external pulse will generate either a positive or negative pulse each time the selected event is active. The Pulse Out will be enabled whenever the processor is emulating. That is, the emulator will transmit a pulse out on every processor cycle in which the conditions of the selected event are met (ignoring the pass counter). The width of the pulse is the width of the high portion of the combined date strobe signals. The external pulse out depends only on the event selected, not on a sequence of

events, and not on a pass count, as the asterisk (*) and note on the event menu reminds you.

7. Pass Counts

The pass count for each event is displayed in the form cccc/pppp, where cccc is the current pass count for the event and pppp is the preset value of the pass count for this event as set in the Event Menu. The default for the preset value is 1.

8. Predefined Sequences

On the right side of the Sequence Menu, there are 14 predefined sequences followed by an extra highlighted line. This extra line is actually two fields. The first will toggle through the numbers "1" to "13" (by depressing the space bar). The second field is for you to generate a sequence to replace a sequence in the predefined list. If the sequence is valid, it will replace the old sequence. If invalid, the sequence entry will flash.

9. User Defined Sequences

Any character can be entered into a sequence field. The validity of the sequence is checked only when the cursor leaves the sequence field and the function is turned "ON". An invalid sequence will flash and an error message will appear on the status line. A sequence is generally valid if it consists of the four events A, B, C and D , the operations OR, AND, THEN and WITHOUT and parentheses. A list of specific rules for the construction of sequences is given below.

Only one level of parentheses is permitted.

During emulation, once a sequence has become true, it remains true until emulation is terminated or until the sequence is altered while in the Freeze Trace condition.

Any or all of the four events may be used in any combination with AND or OR operations. For example A OR C OR B AND D is valid and corresponds to the intuitive meaning A + C + BD where the AND opera-

tion has higher priority than the OR operation. It is important to understand that AND and OR have a different meaning than the combinational logic operations in Boolean algebra. In Boolean algebra these operators imply simultaneous values, for example X AND Y (usually written XY) is true if X is true and if Y is true at the same time. The sequence A AND B is true if the event A, with its pass count, has become true, and if the event B, with its pass count, has become true. The events A and B are different and become true at different times. In this case A AND B means that both have become true in either order, A then B or B then A. If the order of the events is important, you should use the THEN operator described below. If you wanted to detect the simultaneous occurrence of A and B, you would just define a new single event, say C, as the set of conditions you were looking for in A and B.

The events connected by the THEN operator must be in alphabetical order. The sequence B THEN C is valid but the sequences B THEN D and B THEN A are not valid. Up to three THEN operators can be used to define a sequence, A THEN B THEN C THEN D is valid. An expression before or after THEN is not valid. The sequences (A or B) THEN C and A THEN (B OR C) are not valid, but the sequence (A THEN B) OR D is valid. The THEN operator uses the first event to arm the emulator to start looking for the second event. Thus, in the sequence A THEN B, when A becomes true the emulator starts looking for event B and the sequence is true when B has become true.

It is possible to use a sequence to activate one function and a different sequence to activate another function. However, some sequences are incompatible (that is, the definition of one sequence may make another sequence invalid). For example, if the Trigger Trace was set to occur on A THEN B and the External Level Out was to occur when Event B happened then this combination would generate an error message. The reason is that Event B will not be enabled until A has occurred so the Level Out has an invalid sequence. This condition will result in an Incompatible Sequences message in the top status line and the sequences that need attention will flash.

The predefined sequence number 14, A THEN B WITHOUT C, is a special case which cannot be modified. This sequence looks for the event A, not including its pass count. The sequence then becomes true if B, without its pass count, becomes true before C, without its pass count, becomes true. If C does occur before B, then the sequence will start over, looking for event A. Notice the asterisk (*) on sequence 14 and the cor-

```
.Idling.....................................................................

^W Line up    ^R Screen up     ^QR Top of buffer    ^QF Find       BREAK TRACE
^Z Line down  ^C Screen down   ^QC Bottom of buffer ^N  Find next     DISASM
^QB Freeze trace               ^QG Resume trace     ^T  Find trigger  ^P Print
Status: B=BERR', A=AERR', P=VPA', M=VMA', IPL2'-0', FC2-0, R=R/W'

Cycle  Addr.  Data  Ext.          Status: BA PM IPL FC  R  Bus Activity

0039   001084   MOVEM.W [00]001808(PC),2                  RD 78CD < 001808
                                                          RD 9002 < 00180A
0040   00180A   SWAP D2
0043   00108C   MOVE.W A4,D2
0044   00108E   MOVEM.W [00]001808(PC),D3
                                                          RD 78CD < 001808
                                                          RD 90D2 < 00180A
0047   001094   SWAP D3
004A   001096   MOVE.W A6,D3
004B   001098   BRA.S [00]001056
004D   001056   ADDI.L #00010002,[00]001800.W
                                                          RD 182D < 001800
                                                          RD 3046 < 001802
```

Fig. V-6 Trace Menu Screen

responding note on the sequence menu which reminds you that the pass count does not apply.

E. TRACE MENU

The primary purpose of the Trace Menu, as shown in Fig V-6, is to allow the user to view and search through the two 4k by 72 bit trace buffers (Trigger Trace and Break Trace) and to initiate and view the Freeze Trace. Note that the Trigger Trace will only exist if the user has setup a trigger point condition and during emulation has met that condition (activating the trigger point). For further explanation of trigger point see the Trace Overview and Sequence Menu sections.

1. Viewing the Trace Buffer

The Trace Menu is composed of two areas; the Command Summary (the enclosed top portion of the screen) and the Trace Display (the bottom portion of the screen). To select the trace buffer to be displayed, toggle the trace selection field to either "Break Trace" or "Trigger Trace". Either trace may be displayed as disassembled instructions only "DISASM", bus state only "STATE" or both disassembly and state "DISASM/STATE" by toggling the field in the upper right corner of the menu. Anytime a change is made to either of these fields, the trace display is dynamically updated to reflect the new selection.

Once the selected trace has been displayed, the user may move through the 4K of data using the following control commands (these are listed for quick reference in the Command Summary area of the menu):

CNTRL-W	Scroll up one line
CNTRL-Z	Scroll down one line
CNTRL-R	Page up one screen
CNTRL-C	Page down one screen
CNTRL-Q R	Move to top of buffer
CNTRL-Q C	Move to bottom of buffer
CNTRL-T	Move to trigger point

2. Searching the Trace Buffer

There is also a "FIND" command for searching through the currently selected trace buffer. By entering the command CNTRL-Q F the system will display five fields on the first line of the Trace Display area. The fields are (from left to right) Bus Cycle, Address, Data, External Bits and Status Pattern. They are used to define the search criteria and may include don't care conditions (X).

The search is initiated by entering a <CR>. The search always begins at the top of the current page in the trace buffer. Therefore, if a complete search of the buffer is needed, move to the top of the trace buffer prior to initiating a search. The search ends when a match is found or when the bottom of the buffer is reached. When a match is found the Trace Display area will update to contain the trace information found.

When a match is not found the Trace Display area will show the bottom of the trace buffer.

Once the user has entered a search condition, it is maintained in memory and may be used again by entering the CNTRL-N command. This command is therefore used as a "FIND NEXT" occurrence command.

3. Freeze Trace

The Freeze Trace function may be executed by entering the CNTRL-QB command from the Trace Menu. This function allows the user to view the trace buffer and to modify the Event and Sequence Menus, while the processor continues to run. While the "freeze" is active, tracing, break and trigger functions are disabled. To resume these functions, the Resume Trace command may be executed by entering the CNTRL-Q G command from the Trace Menu.

Neither the "freeze" nor the "resume" disturb the running of the processor. When the "resume" is executed, the current Event and Sequence Menu parameters are reprogrammed. Thus, providing the user a mechanism to dynamically modify these parameters. The Freeze Trace can be viewed using the same commands as the Break Trace or the Trigger Trace.

This feature is particularly useful where the target system is using a watchdog timer and needs to make parameter changes without triggering a time-out.

4. Print Trace

Any portion of the trace displays can be transmitted to the auxiliary port (like to a printer or computer) by entering CNTRL - P while in the trace menu and filling in the fields for the beginning and ending cycle numbers. The trace buffer and format for the trace which is to be transmitted to the auxiliary port is determined by the current mode of the display. i.e. if you are currently viewing the Trigger Trace in State Only mode, then that is what will be printed.

From a terminal:

When the Print Trace command is executed, three lines will appear just below the trace menu. The first line will be used to specify the beginning and ending cycle numbers for defining a block of trace which is to be transmitted out the auxiliary port. The second line is used to define a prefix string which will be transmitted out the auxiliary port before the actual trace information is transmitted. The third line is used to define a sufix string which will be transmitted out the auxiliary port after the trace information is transmitted. The prefix and sufix strings are typically used to open and close a file on a computer or to set special printer functions.

Any control characters which need to be transmitted on the prefix or suffix strings are specified by a back slash followed by the 2 character hexadecimal representation of the control code.

eg. \0D\0A will transmit a carriage return followed by a line feed

\1B will transmit an Escape character

The format for the Trace Print is defined in the Interface Menu.

From a host computer:

When operating from a host computer using the Huntsville Microsystems communication software, the CNTRL - P command will be used to transfer Trace information directly to the host on the Main serial port. Here again, the information transmitted to the host will be the currently displayed trace data, including symbolic and source code if available. The user must specify the page format and the file name which is to be opened for the trace information, which can also be a printer.

Trace printing can be aborted by typing CNTRL X.

F. INTERFACE MENU

The Interface Menu shows the current configuration of the Main and Auxiliary RS232 ports, see Fig V-7. The attributes of the main port can not be changed from this menu, but are shown to reflect the switch setting for switch S1 (see Appendix A). The configuration for the Auxiliary port is configurable from this menu. The handshaking is also selectable between XON/XOFF and CTS/RTS.

When the auxiliary port is connected to a printer, the page format can be defined from this menu for the Trace Print command (see Trace Menu).

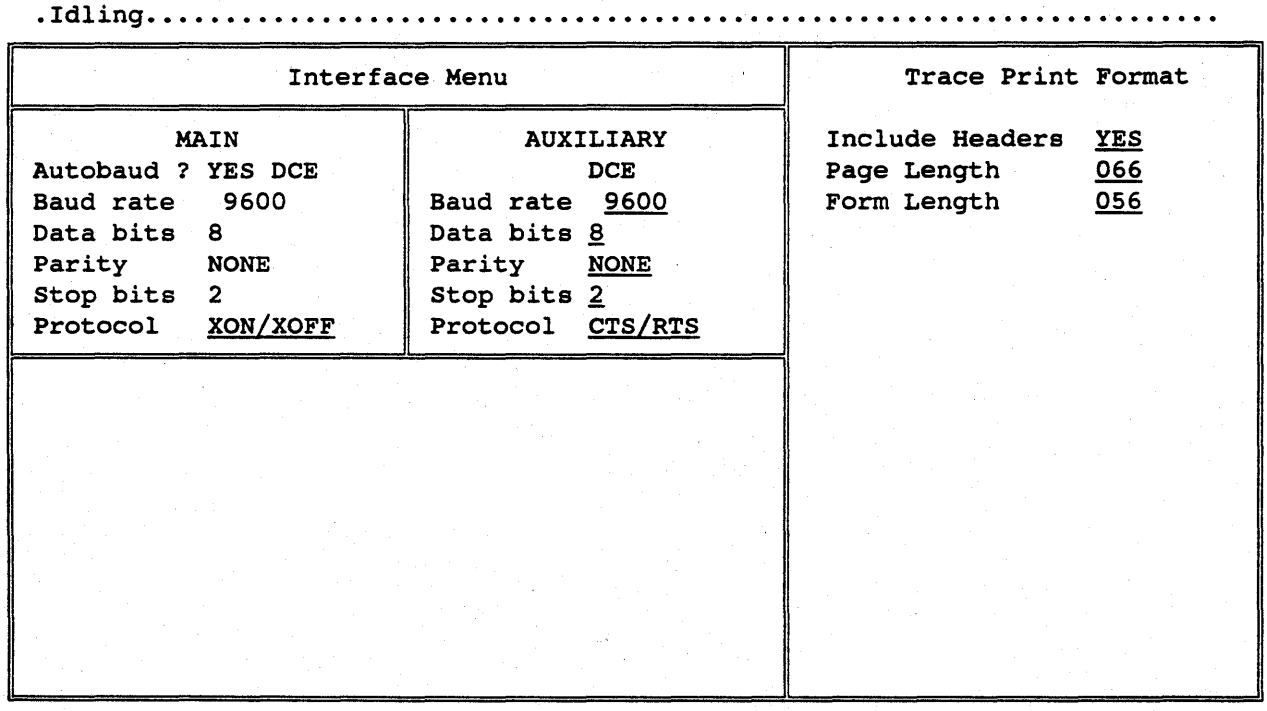


Fig. V-7 Interface Menu Screen

VI. COMMAND LINE OPERATION

The HMI-200 series emulator has been developed with a very fast and easy to use Menu Driven Operating System. The control and flexibility of the menu coupled with the power of the features available in this system make it a very complex development tool.

Therefore, when the user desires to enter a quick and less complicated development session, he can enter into a command line mode in which the emulator will function as a somewhat more conventional debugger. The advantage to this mode is that, once a familiarity with the command structure is established, it is quite a bit quicker to execute the commands from this mode and the whole screen is available for examining data.

To enter the command line mode from the menu mode, depress the "ESC" key. The screen will blank and a dash (-) prompt will appear for a command. To return to menu operation, depress the "ESC" key again.

A. COMMAND LINE EDITING

There are several editing functions available in the HMI-200 command line mode. They consist of a number of control characters (depressing a character while holding the control key down) listed and defined below. In the command line mode, the system is always in the insert mode and characters typed at the keyboard are displayed on the screen and added to the command line.

1. Traditional Editing Commands

< CR > -Carriage Return (or Enter) initiates execution of currently displayed command line. The position of the cursor on the line does not make any difference. Thus, the user may edit the middle of the line and when the line is correct, may enter a < CR > without returning to the end of the line.

CNTRL-U - Abort the current line by moving the cursor to the next line. This leaves the aborted command line displayed on line above for user reference.

CNTRL-R - Re-display the valid characters of the current line on the next line. This is primarily a teletype command to allow the user to clean up the line after several delete and enter sequences.

CNTRL-X- Delete all characters to the left of the cursor and position cursor at the beginning of line.

DEL -Traditional delete with an echo of the deleted character to the screen.

2. Advanced Editing Commands

<LF> - Line Feed has the same function as <CR>.

CNTRL-S- Move cursor one character position to the left.

CNTRL-D- Move cursor one character position to the right.

CNTRL-A - Move cursor to the beginning of the line.

CNTRL-F - Move cursor to the end of the line.

CNTRL-B - Toggle between beginning of the line and the end of the line. If the cursor is at any character position other than the beginning of the line, CNTRL-B will position it to the beginning of the line. If the cursor is positioned at the beginning of the line, CNTRL-B will position it at the end of the line.

TAB - Tab character repositions cursor (by inserting spaces) to the next tab position. Tabs are set every 8 character positions on the command line.

CNTRL-W- Restores the previously executed command line on the current line. If the cursor is positioned at the beginning of the line, the entire command is displayed. If the cursor is located at some other position, only the portion of the previous command that would normally have appeared to the right of the cursor will be displayed.

CNTRL-G - Delete the character at the current cursor position.

CNTRL-K - Delete all characters to the right of and including the current cursor position.

CNTRL-T - Has the same function as CNTRL-K.

CNTRL-Y - Abandon the current command line by clearing the line and repositioning the cursor to the beginning of the line.

B. COMMAND ENTRY

In Command Line Mode, the command abbreviation and the first parameter are not required to be separated, but may be separated by one or more spaces. An exception to this is the G command, see the G command description later in this chapter. If multiple parameter fields are used, they must be separated by a recognized delimiter. Allowed delimiters include a comma, a single space or multiple spaces. If a comma is used as the delimiter, it may be preceded and followed by one or more spaces. This allows the user some flexibility in entering the commands in a form that is comfortable and also allows for minor typing errors.

C. COMMANDS THAT SET GLOBAL PARAMETERS

A global parameter is a default value that is used in all subsequent commands, where it would normally apply and has not been overridden on the command line.

In commands where word or byte values can be specified for input or output, the following commands can be used to set the global parameter.

1. BM (Byte Mode) Command

Set the default for input or output values to byte.

2. WM (Word Mode) Command

Set the default for input or output values to word.

In commands that utilize memory references to the four partitions (Supervisor Program, Supervisor Data, User Program, User Data) you can establish a default by using one of the following commands:

3. DM (Data Mode) Command

Sets the default memory category to be data memory.

4. PM (Program Mode) Command

Sets the default memory category to be program memory.

5. SM (Supervisor Mode) Command

Sets the default memory area to be the supervisor area.

6. UM (User Mode)

Sets the default memory area to be the user area.

D. COMMAND LINE OVERRIDES

An override is a optional command parameter that allows the user to temporarily override the established global (default) parameter. Overrides are entered on the command line in brackets or parenthesis. More than one override may be specified in the brackets and may be entered immediately adjacent to each other without separating spaces or delimiters.

Command line overrides do not apply to all commands. They are represented in the command format as a square bracket enclosing the override codes that apply to the command (the override codes and a brief description of each are given in the table below). Some commands have a predefined or default override. The Assemble Command for example, assumes that the memory type will be "program" unless overridden on the command line. These default overrides will be noted in the formats and their override code letter will be shown underlined.

Some of the overrides shown below are paired with another. The pairs consist of mutually exclusive overrides. It makes no sense to select both word and byte mode, for example. The user would only want to use an override if he did not want to use the global parameter, was not sure how the global parameter was set or the command has a default override that he does not choose to use.

1. S (Supervisor Mode)

Memory addresses given in the command are assumed to be references into either the Supervisor Data or Supervisor Program memory partition. Which one is used will be determined by other overrides or global parameters.

2. U (User Mode)

Memory addresses given in the command are assumed to be references into either the User Program or User Data memory partition. Which one is used will be determined by other overrides or global parameters.

3. P (Program mode)

Memory addresses given in the command are assumed to be references into either the User Program or Supervisor Program memory partition. Which one is used will be determined by other overrides or global parameters.

4. D (Data Mode)

Memory addresses given in the command are assumed to be references into either the User Data or Supervisor Data memory partition. Which one is used will be determined by other overrides or global parameters.

5. W (Word Mode)

The optional data size of input parameter or output (display) value is selected to be word.

6. B (Byte Mode)

The optional data size of input parameter or output (display) value is selected to be byte.

7. T (Target System)

Memory will be accessed in the Target System's memory space overriding the memory map as defined in the configuration menu.

E. EMULATION COMMANDS

To view a summary of the command line mode commands, simply enter a question mark (?) and a carriage return (see ? (Help) Command at the end of this section).

Commands will be shown in the following form:

FF ([SUPDWBT]) (Baddr) (Eaddr)

- The underline denotes an address field.

- The brackets [ ] enclose available overrides.

- Any override shown underlined (as P is in the example) is a command default.

- The parenthesis denote optional fields.

Since some parameter values are saved, optional parameters may be subject to default to a previously used value. Consult the individual command for an explanation of optional parameter handling.

- Any address may be specified as a one to six digit hexadecimal number or as a register name by typing an R followed by a register.

Example: Assume Register D2 = 42355CE2, then
DRD2
will Dump memory starting from address 355CE2

1. A (Assemble) Command

This command allows assembly language mnemonics to be entered into memory starting at the specified address (In-Line Assembly). Once the Assembly command is entered, the address will be displayed with the cursor to the right. At this point, 68000 assembly language instructions can be entered. After each entry, the emulator displays the next address and prompts for the next instruction. An error in the entered instruction will be flagged by displaying a question mark (?) and redisplaying the current address. A period (.) followed by a carriage return terminates the entry sequence and the command.

The format for the Assemble command menu line is:

A ([SUPDT]) Addr

where: Addr - is the hexadecimal starting address

Note: The default override "P" (Program memory area) exists for the Assemble command.

2. C (Compare) Command

The Compare Command will compare one block of memory to another. The first word in the first block will be compared to the first word in the second block, and so on. Any non-comparisons will be displayed on the screen showing the address in the first block with its data value and the address in the second block with its data value. If the two blocks match, then nothing will be displayed.

The format for the Compare command is:

C ([SUPDT]) Baddr1 Eaddr1 ([SUPDT]) Baddr2

where:

Baddr1- is the hexadecimal beginning address of the first compare block

Eaddr1 - is the hexadecimal ending address of the first compare block

Baddr2 - is the hexadecimal beginning address of the second compare block

The size of the first block sets the number of words to be compared.

For each word in block one that does not match the corresponding word in block two, a line is displayed on the screen.

Thisdisplay line has the format:

Addr1 Value1 Addr2 Value2

where:

Addr1 - is the address of the word in block one
Value1 - is the contents of addr1
Addr2 - is the address of the word in block two
Value2 - is the contents of addr2

3. CONFIG (Initial Configuration) Command

This command will return the emulator menus, ports, memory map, predefined sequences and other battery backed-up configuration values to the factory configuration.

4. DL (Disable Latch) Command

This command is used to restore the emulators address latches to a mode in which all addresses which are generated by the processor are placed on the target system address bus. In this mode, the emulator will be reading from various address in the target system. If this is unacceptable, use the Enable Latch (EL) command.

5. D (Dump) Command

The Dump Command will display the contents of memory in the specified address range showing the hexadecimal data with its ASCII equivalent on the screen.

If no ending address is given, then one screen of data will be displayed and the command will terminate. If no starting address is specified then the last ending address (whether user entered or default) will be used as the starting address. Using this method of defaulting allows the user to dump a display of memory from some starting address and get the next and subsequent pages of data dumped by simply typing the "D" and a carriage return.

The Format for the Dump Command is:

D ([SUPDWBT]) (Baddr) (Eaddr)

where:

Baddr - is the hexadecimal beginning address of the block of memory to be displayed

Eaddr - is the hexadecimal ending address of the block of memory to be

The hexadecimal portion of the display can be selected as either byte values separated by spaces or word values separated by spaces. Based on the mode selected by the global parameters or overrides.

6. EL (Enable Latch) Command

The emulator is designed to be as flexible as possible in its interface to a wide variety of target system designs. Generally, when the emulator is plugged into the target system, the address lines are changing, reflecting the emulators on-board monitor which the processor is executing. For the most part, this does not interfere with the operation of the target system.

There are a few cases, however, where a target system will not function properly when random addresses (reads) are placed on its bus. An example of this type of situation would occur when the user has configured the emulator to use Target System DTACK and the target systems memory has parity check. When the emulator and target system are powered up, the emulator processor begins to run internal code while the target system interprets the emulators operation as memory reads. When memory reads are performed on uninitialized memory with parity, then parity errors are bound to occur. The parity error may cause the target system to fail to return a DTACK signal, thus causing the emulator to time-out.

The Enable Latch (EL) command is available to remedy this situation. The EL command will force the emulator to latch out the last valid address used by the target system. For example, after a Reset command is issued, the address 00006 will remain on the target system address bus. Another example would be if memory were Dumped from 100 to 150, then the address 00150 would be latch on the target system address bus.

If the EL command has been activated and the emulator tries to read memory where no DTACK signal will be returned, then the emulator will time out. Instead of latching out the address which caused the DTACK time-out, the emulator will force a dummy read to a known good address, initially 00000 (see TA command), thus latching out a valid address which will always return a DTACK on each memory read which the emulator performs.

7. E/EN (Enter) Command

The Enter Command is used to examine memory at the specified location with the option to change its contents. After executing the Enter Command, the selected address will appear on the screen with the current data. The data can be altered by entering new data and pressing a carriage return, or it can remain unchanged by just typing a carriage return. In either case the next address will be displayed with its data. This command can be terminated by typing a period (.) in the data field followed by a carriage return.

There is a variation of the Enter command (EN) that does not advance to the next address. Whether you enter a change and a carriage return or just a carriage return, it will display the value at the specified address only. This feature is very useful in monitoring a memory mapped I/O location.

The Format for the Enter Command is:

E ([SUPDWBT]) Addr
or
EN ([SUPDWBT]) Addr

where:

Addr - is the hexadecimal address of the data value to be displayed for possible substitution

8. F (Fill) Command

The Fill Command is used to fill memory with a constant. The given data value will be written to each memory location within the specified range. The value may be a byte or a word in length, determined by the global parameters and overrides used.

The format for the Fill Command is:

F ([SUPDWBT]) Baddr Eaddr Val

Where:

Baddr - is the hexadecimal beginning address of the memory block to be filled

Eaddr - is the hexadecimal ending address of the memory block to be filled

Val - Constant value to be used to fill the memory block

9. G (Go) Command

The Go Command is used to start real-time emulation. The options for this command are to run without a breakpoint, or to run with temporary breakpoints as set in this command.

If the option of running without a breakpoint is selected, then any break, trigger, trace, timer or output conditions which have been specified on the Sequence Menu are active.

If the option for running with breakpoints from this command is selected, then it is important to note that the breakpoint is configured as an address field on op-code execution. Also, the configuration of the Event and the Sequence menus become inactive for this option.

In either case emulation will begin at the start address given or default to the current program counter. If a starting address is to be used it must be immediately adjacent to the G, otherwise it will be taken as a break-point address.

The format for utilizing the Event and Sequence menu configurations is:

G(Saddr)

The format for setting temporary breakpoints is:

G(Saddr) Braddr (Braddr2 Braddr3 Braddr4)

Where:

Saddr - is the hexadecimal address where execution (real-time emulation) will begin

Braddr,
Braddr2,
Braddr3,
Braddr4 - is a hexadecimal address where execution (real-time emulation) will stop (or break) on op-code execution

10. I (Input) Command

This command is used to read and display the same memory address each time the carriage return key is depressed. This function is very useful when monitoring a memory mapped I/O location. The Input Command is terminated by entering a period followed by a carriage return.

The format for the Input Command is:

I ([SUPDWBT]) Addr

Where:

Addr - is the hexadecimal address of the data value to be read.

11. L (List Code) Command

This command is used to disassemble the program memory data into assembly language mnemonics. All data and addresses are displayed in hexadecimal.

If no ending address is specified, then one screen of disassembled mnemonics will be displayed and the command will terminate. If no beginning address is specified then the last ending address (whether entered or default) will be used as the beginning address. This allows the user to disassemble code one display screen at a time by entering the command and an initial starting address for the first display and for the next and each subsequent page only the "L" and a carriage return need be entered.

If a range is given, disassembled mnemonics will be displayed from the starting address to the ending address, scrolling the screen if necessary. To pause a scrolling screen enter CNTRL-S and to resume scrolling another CNTRL-S or a CNTRL-Q. To terminate the command while scrolling depress the space bar or enter a carriage return.

The format for the List Command is:

L ([SUPDT]) (Baddr) (Eaddr)

Where:

Baddr - is the hexadecimal beginning address of the memory data to be disassembled

Eaddr - is the hexadecimal ending address of the memory to be disassembled

12. M (Move) Command

The Move Command is used to copy the contents of one block to the contents of another block (non- destructive move). Block one is defined with its starting and ending addresses being the first two address fields displayed, and block two is defined with its starting address being the third address field. The contents of the first memory location of block one is copied to the first memory location of block two, and so on. The next prompt character is not displayed until the move is complete.

The format of the Move Command is:

M ([SUPDT]) Baddr1 Eaddr1 ([SUPDT]) Baddr2

where:

Baddr1- is the hexadecimal beginning address of the data "Source" block

Eaddr1 - is the hexadecimal ending address of the data "Source" block

Baddr2 - is the hexadecimal beginning address of the data "Destination" block

The size of the "Source" block defines the number of bytes to be copied into the "Destination" block.

13. ME (Menu Mode) Command

This command switches the system from the current command line mode of operation to the menu mode of operation. There is an optional parameter to the menu command that allows the user to select the menu that will be displayed.

If the parameter is not used, then the last menu used will be displayed by default.

The format for the Menu Mode Command is:

ME(Selection)

Where:

Selection - is one letter code representing the menu to be displayed .

The one letter codes are:

M - Configuration Menu

C - Command Menu

E - Event Menu

S - Sequence Menu

T - Trace Menu

I - Interface Menu

14. MT/MTL (Memory Test) Command

This command can be used to test memory (RAM) within the given address range.

The short memory test (MT) writes, reads and compares two separate bit patterns for each word within the range. The bit patterns are 5555H and AAAAH, thus cycling all bits of the word through both one and zero.

The long memory test (MTL) is a walking bit pattern. This test writes, reads and compares the word while having only one bit in the word on at a time. This is done by using the Hexadecimal values 1, 2, 4, 8, 10, 20 ---- 8000 for each word within the range.

In either test, any read not comparing with what was written into a location will cause the address, data written and data read to be displayed on the next line of the screen.

The format for the Memory Test command is:

MT ([SUPDT]) Baddr Eaddr

The format for the long Memory Test command is:

MTL ([SUPDT]) Baddr Eaddr

Where:

Baddr - is the hexadecimal beginning address of the memory block to be tested.

Eaddr - is the hexadecimal ending address of the memory block to be tested.

15. O (Output) Command

This command is used to write to a memory address each time the carriage return key is depressed. This function does not increment the address and is very useful when testing a memory mapped I/O location. The output command is terminated by entering a period followed by a carriage return.

After the command has been executed, the last value entered will be output each time a carriage return is entered. The data value can be changed simply by typing a new value before entering a carriage return.

The format for the Output command is:

0 [(SUPDWBT)] Addr, Val

Where:

Addr - is the hexadecimal address of the memory location which is to be written to.

Val - is the constant value which is to be written.

16. PD (Programmable DTACK) Command

This command is used to program the emulator to insert a predefined number of wait states into each memory cycle. Each wait state is one clock cycle in length. The wait states are inserted on any memory cycle in which the emulator is supplying the DTACK signal. The wait states are programmed by typing:

PD count

Where:

Count - is the number of wait states to be inserted within the range 0-254.

Typing PD < CR > will show the current number of wait states selected.

17. PT/PTX (PassThru) Command

This command is used to simulate a wire connecting the emulator's Main and Auxiliary RS-232 ports together. This allows the terminal connected to the emulator's Main port to control the device connected to the emulator's Aux port.

When the PT command is given, the emulator responds with the following:

> Please enter a string of characters to be used to terminate Pass-Thru mode. (Use ^W to recall the last string used)

The emulator will now accept a string of up to eighty displayable ASCII characters. A carriage return terminates the entry of the string.

PassThru mode now commences. Any characters typed at the emulator's terminal will pass through the emulator from the Main port to the Aux port. Likewise, any characters received at the emulator's Aux port will be passed through to the Main port. The emulator is doing its best to simulate a wire connecting the two serial ports together.

The emulator is also watching all the characters entered through the Main serial port. When a string of characters is received which exactly matches the pass through mode termination string, pass thru mode is canceled, and the emulator returns to command mode.

Note: The pass thru mode termination string is also sent out the Aux port, as it is entered. When pass thru mode terminates, a carriage return character is sent out the Aux port. Therefore, the chosen termination string should be something which is harmless to the device connected to the emulator's Aux port.

The PTX version of this command enables the automatic transmission of XON/XOFF characters when the emulator's internal Main and/or Aux serial port buffers are about to overflow. (Otherwise, hardware handshaking is done over the RS-232 RTS line.)

18. RD (Read) Command

This command is used to transfer a hexadecimal file from a host computer to the emulation memory. The file name and RS232 port (main or auxiliary) can be specified with this command. When downloading a file using the Huntsville Microsystems, Inc. communication program (ECS68K), the ASCII file is converted to binary before being transferred to the emulator. This essentially doubles the speed of the transfer over the normal ASCII transfer. When downloading from a host which doesn't use ECS68K, a standard ASCII transfer will occur.

The file formats accepted by the ECS68K software are Motorola S-Record, Intel Hex, Textronix Extended Tex Hex and Microtec Research Absolute. The file formats accepted by the emulator without going through the ECS68K software are the same except for the Microtec Research Absolute. This does not pose a restriction since the Microtec Research software is capable of generating several of the accepted formats.

Once the Read Command has been executed, it can be terminated by typing a CNTRL - X.

When reading a file from the main RS232 port (only allowed if using HMI Communications Software - ECS68K) the format is:

RD ([SUPDT]) (Offset), Filename

When reading a file from the auxiliary RS232 port the format is:

RDX ([SUPDT]) (Offset), (String)

where:

Offset - is a hexadecimal value representing the address offset

Filename - is a file name, valid to the host computer, that designates the file to be down loaded to the emulator. If no extension is specified, the default .ABS will be used.

String - is any string of characters, followed by a carriage return, which the user wants to be transmitted out the auxiliary port.

T = U2D TARGET'S MEMORY

Typically this string would be a command to a computer to transmit a file to the emulator.

Note: The default override "P" (Program memory area) exists for the read command.

19. RS (Reset) Command

This command will issue a reset to the 68000 processor. All registers will be updated to the value which they contain upon a processor reset. The Program Counter is initialized to the contents of memory locations 4H and 6H and the Stack Pointer is initialized to the contents of memory locations 0H and 2H.

20. SP (Stop) Command

This command is used to stop emulation in order to regain control of the 68000 processor. Upon entering this command, emulation will break, and the message "PROCESSOR IS IDLING" will be displayed along with the current values of the registers.

21. SR/SRN (Search) Command

This command can be used to search memory between two addresses for a particular data pattern. When a match is found, the address is displayed. To search for the next occurrence, hit the Carriage Return key. To terminate this command, type a period and then return. If no match is found, then nothing is displayed.

There is also a Search Not option available. In this mode the value is searched for, but the command displays the address of all memory locations in the given range that do not match the search value. This is useful when checking a block of memory that should contain a constant value, such as zero or the value of a previous Fill command.

The format for the Search command is:

S ([SUPDWBT]) Baddr Eaddr Val

The format for the Search Not command is:

SRN ([SUPDWBT]) Baddr Eaddr Sval

where:

Baddr- is the hexadecimal beginning address of the block of memory to be searched

Eaddr - is the hexadecimal ending address of the block of memory to be searched

Val - is the data value to be searched for and will be 1-20 byte values or 1-10 word values (each value separated by a comma or a space) or a 1-20 ASCII character string enclosed in quotes.

Sval - is the data value to be searched for and will be a byte value, a word value or one or two ASCII characters enclosed in quotes.

22. SS/SSX (Single Step) Command

This command is used to execute one instruction at a time. Each time a step is performed, the registers, the address and instruction are displayed on the screen. Therefore, multiple steps will display a history of program flow.

There are three options for operating the Single Step Command. The first option is to single step every time a carriage return is entered. To terminate this mode, type a period and then a carriage return. The second option is to single step a specified number of times. The third option is to single step continuously until a carriage return is entered.

The format for Single Stepping once for each carriage return entered is:

SS

The format for Single Stepping a specified number of times is:

SSCount

The format for Single Stepping continuously until a carriage return is entered is:

SSX

Where:

Count - is the number of iterations to single step

23. TA (Target Address) Command

If the Enable Latch (EL) command has been activated and the emulator tries to read memory where no DTACK signal will be returned, then the emulator will time out. Instead of latching out the address which caused the DTACK time-out, the emulator will force a dummy read to a known good address, initially 00000. If this default address is not a good address in which to perform a dummy read, then the Target Address can be changed by typing:

TAaddr

Where:

addr - is the address which is to be latched onto the target system's address bus after any DTACK time-out.

Typing TA <CR> will show the current Target Address.

24. TERM (Terminal selection) Command

This command is used to select a particular terminal which is to be connected to the emulators Main serial port. After invoking this command, a list of currently supported terminals will be displayed. Enter the letter corresponding with the desired terminal. The default terminal is the Liberty Freedom One.

When connecting to a computer using the HMI-100 communication package, it is not necessary to select a terminal through this command because the HMI-100 will configure the emulator automatically.

When connecting to a DEC terminal or compatible, disable Xon-Xoff in the terminal. Also, on a DEC terminal, function key PF1 can be used as the ESC key to enter and leave menus and function key PF2 can be used as the HOME key.

25. VER (Version) Command

This command displays the current version and release date of the emulator firmware.

26. W/WX (Write) Command

This command is used to transfer files from the emulation memory to a host computer. The size of the block being transferred is specified by the two address fields activated by this command. When using the Huntsville Microsystems, Inc. communication software (ECS68K), the block will be written as a Motorola S-record file on disk under the specified file name.

The format for writing to the main RS232 port is:

W ([SUPDT]) Baddr Eaddr Filename

The format for writing to the auxiliary RS232 port is:

WX ([SUPDT]) Baddr Eaddr

where:

Baddr- is the hexadecimal beginning address of code to be written

Eaddr - is the hexadecimal ending address of code to be written

Filename -is a filename, valid to the host computer, that designates the file to be uploaded (written) to the host computer. If no extension is specified, the default .ABS will be used.

Note: The default override "P" (Program memory area) exists for the write command.

27. X/XFL (Examine Registers and Flags) Command

This command allows the user to examine and modify the registers and flags (status register). There are three options to the examine command. First is to display all registers. The second option is to display a specific register and allow it to be modified. The last option is to display the flags (status register) and the condition code status (shown in the table below). This option also allows the flags to be modified by typing in the condition code status.

The format for displaying the registers is:

X

The format for displaying a specific register and allowing it to be modified is:

XReg

The format for displaying the flags (status register condition codes) is:

XFL

Where:

Reg - is the two character register name, such as A0 or D4

Bit Location in Status Register	Flag or Condition Code Name	Not Active Code	Active Code
15	Trace Mode	NT	TR
13	Supervisor State	U	S
8-10	Interrupt Mask	-	I0-I7
4	Extend	NX	XT
3	Negative	PS	NG
2	Zero	NZ	ZR
1	Overflow	NV	OV
0	Carry	NC	CY

STATUS REGISTER CODES

28. ? (Help) Command

This command displays a help summary of the Command Line Mode commands. It will display the first page of the command summary and wait for the user to enter a carriage return before displaying the next and each subsequent page.

The command summary display is shown below and continued on the following pages.

68000 Emulator Shorthand:

aa - beginning address
bb - beginning address, 2nd block
cc - count
ee - ending address
dd - data
rr - register
Registers may be substituted for addresses (aa,bb,ee) by preceeding them with R (e.g. RD0).

%	DISPLAY MACRO HELP MENU
Aaa	ACTIVATE IN-LINE ASSEMBLER AT aa
BM	BYTE DISPLAY MODE
Caa,ee,bb	COMPARE MEMORY BLOCK AT aa THROUGH ee WITH BLOCK AT bb
CONFIG	RECALL FACTORY CONFIGURATION
Daa,ee	DISPLAY MEMORY BLOCK. aa AND ee OPTIONAL
DL	DISABLE ADDRESS LATCHING TO TARGET SYSTEM
DM	DATA MEMORY MODE
Eaa	EXAMINE/CHANGE MEMORY AT aa
EL	ENABLE ADDRESS LATCHING TO TARGET SYSTEM
ENaa	EXAMINE/CHANGE MEMORY AT aa WITH NO ADDRESS INCREMENT
Faa,ee,dd	FILL MEMORY BLOCK AT aa THROUGH ee WITH dd
G	START PROGRAM EXECUTION FROM CURRENT LOCATION

Command Summary Help Pages

G,ee	START PROGRAM EXECUTION FROM CURRENT LOCATION AND BREAK AT ee
Gaa	START PROGRAM EXECUTION AT aa
Gaa,ee	START PROGRAM EXECUTION AT aa AND BREAK AT ee
Iaa	NON-INCREMENTAL ADDRESS READ
Laa,ee	DISASSEMBLE MEMORY BLOCK. aa AND ee OPTIONAL
Maa,ee,bb	MOVE MEMORY BLOCK FROM aa THROUGH ee TO BLOCK AT bb
MTaa,ee	TEST BLOCK OF MEMORY FROM aa THROUGH ee
MTLaa,ee	LONG MEMORY TEST FOR BLOCK FROM aa THROUGH ee
Oaa,bb	NON-INCREMENTAL ADDRESS WRITE
PDccc	PROGRAMABLE WAIT STATES FOR DTACK (DECIMAL 0 - 254)
PM	PROGRAM MEMORY MODE
PT	ENTER PASS-THRU MODE
RD,filename	READ HEX FILE (COMMUNICATIONS SOFTWARE ONLY)
RDaa,filename	READ HEX FILE WITH BIAS aa (COMMUNICATIONS SOFTWARE ONLY)
RDX	READ HEX FROM THE AUXILIARY SERIAL PORT
RDXaa	READ HEX WITH BIAS aa FROM THE AUXILIARY SERIAL PORT
RS	RESET TARGET SYSTEM
SM	SUPERVISOR MEMORY MODE
SP	STOP PROGRAM EXECUTION
SRaa,ee,dd	SEARCH BLOCK OF MEMORY FROM aa THROUGH ee FOR DATA dd
SRNaa,ee,dd	SEARCH BLOCK OF MEMORY FROM aa THROUGH ee FOR NOT DATA dd
SS	SINGLE STEP
SSX	SINGLE STEP CONTINUOUSLY
SScc	SINGLE STEP cc CYCLES
ST	SHOW PROCESSOR STATUS
TAaa	DEFINE TARGET ADDRESS
TERM	TERMINAL SETUP MENU
UM	USER MEMORY MODE
VER	DISPLAY VERSION NUMBER

Command Summary Help Pages

Waa,ee,filename	WRITE MEMORY BLOCK FROM aa THROUGH ee IN HEX FILE FORMAT
WXaa,ee	WRITE MEMORY BLOCK TO AUXILIARY SERIAL PORT
WM	WORD DISPLAY MODE
X	DISPLAY REGISTERS
Xrr	EXAMINE/CHANGE REGISTER rr
XFL	EXAMINE/CHANGE FLAGS

If you are using the Huntsville Microsystems communication software, ECS68K, then you will also get the following additional listing of commands in response to the ? command.

Communications Software Enhanced Functions:

+ filename	Open output file
+ + filename	Open output file for append
: symbol[= [value]]	Show/Delete/Edit symbol value
?	Display enhanced help message
EXIT	Return to the operating system
N filename[,ABCHMS]	Establish default filename for RD and W
/	Pause/Resume a batch file
:: label	Label a line in a batch file
; comment	Comment a line in a batch file
< filename	Open a batch file
> filename	Open a keyboard capture file
d:	Change default disk drive
CHDIR [directory]	Perform DOS CHDIR function
DIR [filename]	Perform DOS DIR function
TYPE filename	Perform DOS TYPE function

Command Summary Help Pages

F. MACROS

A macro is a sequence of emulator commands which can be executed under one command known as a macroname. A macroname is a sequence of characters defined by the user. Up to eight macros can be created.

1. %L macroname (Open)

This is the learn command which is used to open a macro under the user defined macroname. After this command is executed, any characters typed at the keyboard will go into the macro as an emulator command.

2. % (Close/Help)

This command actually has two functions. If the macro learn command (%L) had previously been entered, then the % command is the termination character which will close a macro. Otherwise the % command can be used to call up a macro help menu.

3. % macroname (Execute)

The percent sign followed by a previously created macro will execute the macro. Each command will be displayed and immediately executed until the macro is complete.

4. %V macroname (Save)

This command will save the configuration of the emulator under a macroname. Once saved, the configuration of the emulator can be restored by executing this macro.

5. %S (Show Directory)

This command will show a directory of previously defined macros.

6. %D macroname (Delete)

This command will delete the macro defined by macroname from the macro library.

7. %C (Clear)

This command will clear all macros from the macro library.

Note: For obvious reasons, L, V, S, D and C cannot be used as macro-names.

APPENDIX A

A. Switch settings for S1 and S2.

Switch S1: Main RS232 Port Configuration

1 - up DTE. This makes Pin 2 the data input to the emulator, Pin 3 the data output from the emulator, Pin 4 the CTS/ input and Pin 5 the RTS/ output.

1 - down DCE. This makes Pin 2 the data output from the emulator, Pin 3 the data input to the emulator, Pin 4 the RTS/ output and Pin 5 the CTS/ input.

Note: Pins 6, 8, and 20 on the RS232 connector are pulled high through a 10K pull-up to + 12V. Pins 1 and 7 are connected to ground.

2 - up Manual Baud Rate Selection. The Baud rate will be determined by switches 7-10.

2 - down Auto Baud Rate Selection. The Baud rate will be determined when a period (.) is transmitted to the emulator after a push-button reset.

3 - up Parity enabled

3 - down Parity disabled

4 - up Parity even

4 - down Parity odd

5 - up 7 data bits

5 - down 8 data bits

6 - up 1 stop bit

6 - down 2 stop bits

Switch S1				
10	9	8	7	Baud Rate
---	--	--	--	----------
up	up	up	up	19.2K
up	up	up	down	38.4K
up	up	down	up	19.2K
up	up	down	down	9600
up	down	up	up	4800
up	down	up	down	2400
up	down	down	up	1200
up	down	down	down	600
down	up	up	up	300
down	up	up	down	150
down	up	down	up	110
down	up	down	down	9600
down	down	up	up	9600
down	down	up	down	9600
down	down	down	up	9600
down	down	down	down	9600

Switch S2: Auxiliary RS232 Configuration

1 - up DTE. This makes Pin 2 the data input to the emulator, Pin 3 the data output from the emulator, Pin 4 the CTS/ input and Pin 5 the RTS/ output.

1 - down DCE. This makes Pin 2 the data output from the emulator, Pin 3 the data input to the emulator, Pin 4 the RTS/ output and Pin 5 the CTS/ input.

Note: Pin 6, 8, and 20 on the RS232 connector are pulled high through a 10K pull-up to +12V. Pins 1 and 7 are connected to ground.

2 - up Use the configuration from the Interface Menu for configuring this port.

2 - down Use switches 3-10 to configure this port.

3 - up Parity enabled

3 - down Parity disabled

4 - up Parity even

4 - down Parity odd

5 - up 7 data bits

5 - down 8 data bits

6 - up 1 stop bit

6 - down 2 stop bits

10	9	8	7	Baud Rate
---	--	--	--	----------
up	up	up	up	19.2K
up	up	up	down	38.4K
up	up	down	up	19.2K
up	up	down	down	9600
up	down	up	up	4800
up	down	up	down	2400
up	down	down	up	1200
up	down	down	down	600
down	up	up	up	300
down	up	up	down	150
down	up	down	up	110
down	up	down	down	9600
down	down	up	up	9600
down	down	up	down	9600
down	down	down	up	9600
down	down	down	down	9600

B. Factory Configuration

Switch S1	Switch S2
1 - down	1 - down
2 - down	2 - down
3 - down	3 - down
4 - down	4 - down
5 - down	5 - down
6 - down	6 - down
7 - down	7 - down
8 - down	8 - down
9 - down	9 - down
10 - down	10 - down

APPENDIX B

A. 68000 Mnemonics and Corresponding Opcodes

Mnemonic		Opcode
ABCD.B Dx,Dy		1100 yyy1 0000 0xxx
ABCD.B -(Ax),-(Ay)		1100 yyy1 0000 1xxx
ADD.z Dx,Dy		1101 yyy0 LW00 0xxx
ADD.z e,Dy		1101 yyy0 LWmm mrrr + X
ADD.z Dx,e		1101 xxx1 LWmm mrrr + X
ADDA.w e,Ay		1101 yyyL 11mm mrrr + X
ADDI.z #n,Dy		0000 0110 LW00 0yyy + N
ADDI.z #n,e		0000 0110 LWmm mrrr + N + X
ADDQ.z #n,e		0101 ooo0 LWmm mrrr + X
ADDX.z Dx,Dy		1101 yyy1 LW00 0xxx
ADDX.z -(Ax),-(Ay)		1101 yyy1 LW00 1xxx
AND.z Dx,Dy		1100 yyy0 LW00 0xxx
AND.z e,Dy		1100 yyy0 LWmm mrrr + X
AND.z Dx,e		1100 xxx1 LWmm mrrr + X
ANDI.z #n,Dy		0000 0010 LW00 0yyy + N
ANDI.z #n,e		0000 0010 LWmm mrrr + N + X
ANDI.B #n,CCR		0000 0010 0011 1100 + N
ANDI.W #n,SR	(P)	0000 0010 0111 1100 + N
ASL.z Dx,Dy		1110 xxx1 LW10 0yyy
ASL.z #n,Dy		1110 ooo1 LW00 0yyy
ASL.W e		1110 0001 11mm mrrr + X
ASR.z Dx,Dy		1110 xxx0 LW10 0yyy
ASR.z #n,Dy		1110 ooo0 LW00 0yyy
ASR.W e		1110 0000 11mm mrrr + X
Bcc.S a		0110 cccc dddd dddd
Bcc.L a		0110 cccc 0000 0000 + D
BCHG.B Dx,e		0000 xxx1 01mm mrrr + X
BCHG.L Dx,Dy		0000 xxx1 0100 0yyy
BCHG.B #n,e		0000 1000 01mm mrrr + B + X
BCHG.L #n,Dy		0000 1000 0100 0yyy + B
BCLR.B Dx,e		0000 xxx1 10mm mrrr + X
BCLR.L Dx,Dy		0000 xxx1 1000 0yyy
BCLR.B #n,e		0000 1000 10mm mrrr + B + X
BCLR.L #n,Dy		0000 1000 1000 0yyy + B
BSET.B Dx,e		0000 xxx1 11mm mrrr + X
BSET.L Dx,Dy		0000 xxx1 1100 0yyy

```
BSET.B #n,e                  0000 1000 11mm mrrr + B + X
BSET.L #n,Dy                 0000 1000 1100 0yyy + B

BTST.B Dx,e                  0000 xxx1 00mm mrrr + X
BTST.L Dx,Dy                 0000 xxx1 0000 0yyy
BTST.B #n,e                  0000 1000 00mm mrrr + B + X
BTST.L #n,Dy                 0000 1000 0000 0yyy + B

CHK.W Dx,Dy                  0100 yyy1 1000 0xxx
CHK.W e,Dy                   0100 yyy1 10mm mrrr + X

CLR.z Dy                     0100 0010 LW00 0yyy
CLR.z e                      0100 0010 LWmm mrrr + X

CMP.z e,Dy                   1011 yyy0 LWmm mrrr + X
CMPA.w e,Ay                  1011 yyyL 11mm mrrr + X
CMPI.z #n,Dy                 0000 1100 LW00 0yyy + N
CMPI.z #n,e                  0000 1100 LWmm mrrr + N + X
CMPM.z (Ax)+,(Ay)+           1011 yyy1 LW00 1xxx

DBcc.L Dx,a                  0101 cccc 1100 1xxx + D

DIVS.W Dx,Dy                 1000 yyy1 1100 0xxx
DIVS.W e,Dy                  1000 yyy1 11mm mrrr + X

DIVU.W Dx,Dy                 1000 yyy0 1100 0xxx
DIVU.W e,Dy                  1000 yyy0 11mm mrrr + X

EOR.z Dx,Dy                  1011 xxx1 LW00 0yyy
EOR.z Dx,e                   1011 xxx1 LWmm mrrr + X
EORI.z #n,Dy                 0000 1010 LW00 0yyy + N
EORI.z #n,e                  0000 1010 LWmm mrrr + N + X
EORI.B #n,CCR                0000 1010 0011 1100 + N
EORI.W #n,SR           (P)   0000 1010 0111 1100 + N

EXG.L Dx,Dy                  1100 xxx1 0100 0yyy
EXG.L Dx,Ay                  1100 xxx1 1000 1yyy
EXG.L Ax,Dy                  1100 yyy1 1000 1xxx
EXG.L Ax,Ay                  1100 yyy1 0100 1xxx

EXT.w Dy                     0100 1000 1L00 0yyy

ILLEGAL                      0100 1010 1111 1100

JMP.j e                      0100 1110 11mm mrrr + X

JSR.j e                      0100 1110 10mm mrrr + X

LEA.L e,Ay                   0100 yyy1 11mm mrrr + X
```

```
LINK Ax,#d                  0100 1110 0101 0xxx + D

LSL.z Dx,Dy                 1110 xxx1 LW10 1yyy
LSL.z #n,Dy                 1110 ooo1 LW00 1yyy
LSL.W e                     1110 0011 11mm mrrr + X


LSR.z Dx,Dy                 1110 xxx0 LW10 1yyy
LSR.z #n,Dy                 1110 ooo0 LW00 1yyy
LSR.W e                     1110 0010 11mm mrrr + X

MOVE.z e,Dy                 00ZZ yyy0 00mm mrrr + X

MOVE.z e,ey                 00ZZ yyye eemm mrrr + X + Xy

MOVE.W CCR,Dy     (68010)   0100 0010 1100 0yyy
MOVE.W CCR,e      (68010)   0100 0010 11mm mrrr + X
MOVE.W Dx,CCR               0100 0100 1100 0xxx
MOVE.W e,CCR                0100 0100 11mm mrrr + X
MOVE.W Dx,SR          (P)   0100 0110 1100 0xxx
MOVE.W e,SR           (P)   0100 0110 11mm mrrr + X
MOVE.W SR,Dy                0100 0000 1100 0yyy
MOVE.W SR,e                 0100 0000 11mm mrrr + X
MOVE.W SR,Dy    (P,68010)   0100 0000 1100 0yyy
MOVE.W SR,e     (P,68010)   0100 0000 11mm mrrr + X
MOVE.L USP,Ay         (P)   0100 1110 0110 1yyy
MOVE.L Ax,USP         (P)   0100 1110 0110 0xxx
MOVEA.w e,Ay                00ZZ yyy0 01mm mrrr + X
MOVEQ.L #n,Dy               0111 yyy0 nnnn nnnn

MOVEC.L SFC,sy  (P,68010)   0100 1110 0111 1010 + syyy 0000 0000 0000
MOVEC.L DFC,sy  (P,68010)   0100 1110 0111 1010 + syyy 0000 0000 0001
MOVEC.L USP,sy  (P,68010)   0100 1110 0111 1010 + syyy 1000 0000 0000
MOVEC.L VBR,sy  (P,68010)   0100 1110 0111 1010 + syyy 1000 0000 0001
MOVEC.L sy,SFC  (P,68010)   0100 1110 0111 1011 + syyy 0000 0000 0000
MOVEC.L sy,DFC  (P,68010)   0100 1110 0111 1011 + syyy 0000 0000 0001
MOVEC.L sy,USP  (P,68010)   0100 1110 0111 1011 + syyy 1000 0000 0000
MOVEC.L sy,VBR  (P,68010)   0100 1110 0111 1011 + syyy 1000 0000 0001

MOVEM.w q,e                 0100 1000 1Lmm mrrr + Q + X
MOVEM.w e,q                 0100 1100 1Lmm mrrr + Q + X

MOVEP.w Dx,d(Ay)            0000 xxx1 1L00 1yyy + D
MOVEP.w d(Ax),Dy            0000 yyy1 0L00 1xxx + D

MOVES.z sx,e    (P,68010)   0000 1110 LWmm mrrr + sxxx 1000 0000 0000 + X
MOVES.z e,sy    (P,68010)   0000 1110 LWmm mrrr + syyy 0000 0000 0000 + X

MULS.W Dx,Dy                1100 yyy1 1100 0xxx
MULS.W e,Dy                 1100 yyy1 11mm mrrr + X
```

```
MULU.W Dx,Dy              1100 yyy0 1100 0xxx
MULU.W e,Dy               1100 yyy0 11mm mrrr + X

NBCD.B Dy                 0100 1000 0000 0yyy
NBCD.B e                  0100 1000 00mm mrrr + X

NEG.z Dy                  0100 0100 LW00 0yyy
NEG.z e                   0100 0100 LWmm mrrr + X


NEGX.z Dy                 0100 0000 LW00 0yyy
NEGX.z e                  0100 0000 LWmm mrrr + X
NOP                       0100 1110 0111 0001

NOT.z Dy                  0100 0110 LW00 0yyy
NOT.z e                   0100 0110 LWmm mrrr + X

OR.z Dx,Dy                1000 yyy0 LW00 0xxx
OR.z e,Dy                 1000 yyy0 LWmm mrrr + X
OR.z Dx,e                 1000 xxx1 LWmm mrrr + X
ORI.z #n,Dy               0000 0000 LW00 0yyy + N
ORI.z #n,e                0000 0000 LWmm mrrr + N + X
ORI.B #n,CCR              0000 0000 0011 1100 + N
ORI.W #n,SR          (P)  0000 0000 0111 1100 + N

PEA.L e                   0100 1000 01mm mrrr + X

RESET                (P)  0100 1110 0111 0000

ROL.z Dx,Dy               1110 xxx1 LW11 1yyy
ROL.z #n,Dy               1110 ooo1 LW01 1yyy
ROL.W e                   1110 0111 11mm mrrr + X

ROR.z Dx,Dy               1110 xxx0 LW11 1yyy
ROR.z #n,Dy               1110 ooo0 LW01 1yyy
ROR.W e                   1110 0110 11mm mrrr + X

ROXL.z Dx,Dy              1110 xxx1 LW11 0yyy
ROXL.z #n,Dy              1110 ooo1 LW01 0yyy
ROXL.W e                  1110 0101 11mm mrrr + X


ROXR.z Dx,Dy              1110 xxx0 LW11 0yyy
ROXR.z #n,Dy              1110 ooo0 LW01 0yyy
ROXR.W e                  1110 0100 11mm mrrr + X

RTD #d              (68010)  0100 1110 0111 0100 + D

RTE                  (P)  0100 1110 0111 0011
```

```
RTE              (P,68010) 0100 1110 0111 0011

RTR                        0100 1110 0111 0111

RTS                        0100 1110 0111 0101

SBCD.B Dx,Dy               1000 yyy1 0000 0xxx
SBCD.B -(Ax),-(Ay)         1000 yyy1 0000 1xxx

Scc.B Dy                   0101 cccc 1100 0yyy
Scc.B e                    0101 cccc 11mm mrrr + X

STOP #n                (P) 0100 1110 0111 0010 + N


SUB.z Dx,Dy                1001 yyy0 LW00 0xxx
SUB.z e,Dy                 1001 yyy0 LWmm mrrr + X
SUB.z Dx,e                 1001 xxx1 LWmm mrrr + X
SUBA.w e,Ay                1001 yyyL 11mm mrrr + X
SUBI.z #n,Dy               0000 0100 LW00 0yyy + N
SUBI.z #n,e                0000 0100 LWmm mrrr + N + X
SUBQ.z #n,e                0101 ooo1 LWmm mrrr + X

SUBX.z Dx,Dy               1001 yyy1 LW00 0xxx
SUBX.z -(Ax),-(Ay)         1001 yyy1 LW00 1xxx

SWAP.W Dy                  0100 1000 0100 0yyy

TAS.B Dy                   0100 1010 1100 0yyy
TAS.B e                    0100 1010 11mm mrrr + X

TRAP #n                    0100 1110 0100 nnnn

TRAPV                      0100 1110 0111 0110

TST.z Dx                   0100 1010 LW00 0xxx
TST.z e                    0100 1010 LWmm mrrr + X

UNLK Ay                    0100 1110 0101 1yyy
```

a - Address
d - Displacement (may be calculated from 'a')
D - 16-bit displacement (may be calculated from 'a')

B - Bit number: 0000 0000 nnnn nnnn (calculated from 'n')

cc - Condition Codes:

cccc	cc	Bcc	DBcc	Scc
0000		RA	T	T
0001		SR	F RA	F
0010	HI			
0011	LS			
0100	CC HS			
0101	CS LO			
0110	NE			
0111	EQ			
1000	VC			
1001	VS			
1010	PL			
1011	MI			
1100	GE			
1101	LT			
1110	GT			
1111	LE			

e - Effective address:

e - Alternate (Destination) 'm'
m - Mode
r - Register
X - optional eXtension word(s)

	e	mmm	rrr	X
	Dx	000	xxx	
	Ax	001	xxx	- Only if 'z' is 'W' or 'L'
	(Ax)	010	xxx	
	(Ax)+	011	xxx	
	-(Ax)	100	xxx	
	d(Ax)	101	xxx	dddd dddd dddd dddd
	d(Ax,sy.w)	110	xxx	syyy L000 dddd dddd
	a.W	111	000	aaaa aaaa aaaa aaaa
	a.L	111	001	aaaa aaaa aaaa aaaa
				aaaa aaaa aaaa aaaa
	a(PC)	111	010	dddd dddd dddd dddd
	a(PC,sy.w)	111	011	syyy L000 dddd dddd
Byte	#n	111	100	0000 0000 nnnn nnnn
Word	#n	111	100	nnnn nnnn nnnn nnnn
Long	#n	111	100	nnnn nnnn nnnn nnnn
				nnnn nnnn nnnn nnnn

```
n - Data
N - Data extension. Byte = 0000 0000 nnnn nnnn
                    Word = nnnn nnnn nnnn nnnn
                    Long = nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn

o - Data range 0-7, representing 8,1-7 (calculated from 'n')

(P) - Privileged instruction
(68010) - 68010 instruction

q - Register List Mask:
        Consists of ranges (sx-sy) or registers (sx) separated by
/'s

        if mmm is 100, Q = D0 D1 D2 D3  D4 D5 D6 D7  A0 A1 A2 A3  A4 A5
                           A6 A7
        otherwise,     Q = A7 A6 A5 A4  A3 A2 A1 A0  D7 D6 D5 D4  D3 D2
                           D1 D0

s - Register name. 0 if 'D', 1 if 'A'

x,y - Register number 0-7

b - 'z' minus 'L' (i.e. 'B' or 'W')
w - 'z' minus 'B' (i.e. 'W' or 'L')
z - Optional length identifier: 'B', 'W', 'L'
L - 1 if 'z' is 'L', 0 otherwise
W - 1 if 'z' is 'W', 0 otherwise
ZZ - Special 'z':
        'B' - 01
        'W' - 11
        'L' - 10

j - Short or Long Jump:
        'S' - one word absolute (m.r = 7.0)
        'L' - one long absolute (m.r = 7.1)

Note:  A7 equals SP and either USP or SSP depending on the 'S' mode bit
```

HMI 110 SERIES
SOURCE DEBUGGER
FOR THE HMI-200-68000/68020 EMULATOR

Huntsville Microsystems, Inc.

August 15, 1990

IBM PC/XT/AT SOURCE DEBUG COMMUNICATIONS SOFTWARE FOR HMI-200-68K EMULATORS AND PC-DOS v2.0 OR HIGHER

Supplied on your IBM compatible master disk are the following files:

ECS68KS.EXE	Executable Source Debug Communications Software program which uses the IBM-PC/XT/AT COM1 or COM2 serial port.
ECS68KS.DOC	This file.

Starting ECS68KS:

The standard ECS68KS command line is formatted as follows:

```
ECS68KS [-baud] [-port] [-Ddownloadport] [-Llinesym] [<hexfile>]
```

All command line arguments, as indicated by the square brackets ('[' and ']'), are optional.

The command line arguments are as follows:

-baud Select the serial communications port baud rate.

The available baud rate selections are: 38400, 19200, 9600 (the default), 4800, 2400, 1200, 600, 300, 150, 110, and None. The selection 'N' or 'None' means that no serial port initialization is performed by ECS68KS. The last baud rate programmed for the serial port by some external program (i.e. MODE) is used as the communication software's serial port baud rate.

-port Select the serial communications port.

The available port selections are: COM1, COM2, and AUX (the default). AUX is either COM1 or COM2, depending on the computer's installed hardware. If a COM1 port is installed in the system, then the AUX port is COM1, even if a COM2 port is also installed. If there is no COM1 port, but there is a COM2 port, then AUX becomes COM2. If both COM1 and COM2 ports are installed on the system, ECS68KS will always use COM1 for its communications port, unless specifically told to use COM2 on the command line, or through the ECS68KPORT environment variable (described below).

-Ddownloadport Select the parallel download port.

The available port selections are: LPT1 (the default), LPT2 and LPT3. This option is only available for use with those emulators equipped with a parallel download port.

-Llinesym Select the format for line number symbols.

In order to enable source-level debugging, ECS68KS must know the relationship between the lines in the source files on disk, and the assembly language code in the emulator. This relationship is defined through the Line Number Symbols.

The default format for line number symbols recognized by ECS68KS is "L_%u". This is a scanf-formatted string indicating that line number symbols are those symbols which consist of one capital 'L' character, followed by an '_' character, followed by a unsigned decimal number. The unsigned decimal number is the line number. The value associated with the symbol is the address of the first assembly-language instruction generated by that line of source code.

This command line switch allows source-level debugging using any compiler that produces line number symbols. All that needs to be done is specify the compiler's line number symbol format as a scanf string, either on the command line, or through the ECS68KLINE environment variable (see "Environment Variables" below).

<hexfile> Set up the named file as the default file for all emulator hex file Read and Write operations. The hexfile name is a standard DOS filename, with optional drive, path, and extent portions. If not specified, the drive and path default to the currently logged drive and directory. The file's extent, if not given, defaults to ".ABS".

When ECS68KS is started, it will display a sign-on message and attempt to establish a communications link with the HMI series 200 emulator connected to the designated (or default) serial communications port. If the emulator has been Reset (by depressing the small black button on the front of the emulator's case), then the emulator's sign-on message will also be displayed, followed by the emulator's Command Line Prompt (a single 'dash' or minus sign character ['-']). If the emulator has not been reset, then only the command line prompt will appear.

If ECS68KS cannot establish the communications link to the

emulator, it will time-out and display an error message, along with a prompt to try again. Typing any key will cause ECS68KS to reset itself, and try again to initialize the communications link with the emulator. Entering a Control-C or a Control-Break character instead will cause ECS68KS to give up and exit back to the operating system.

To exit ECS68KS under normal circumstances, use the "EXIT" command while at the command line prompt.

Startup Examples:

```
ECS68KS                 ;Start ECS68KS

ECS68KS -19200          ;Set 19200 baud

ECS68KS -COM2           ;Use COM2

ECS68KS -DLPT3          ;Download to LPT3

ECS68KS -LLN%u          ;Set the line number symbol format to
                        ;"LN%u"

ECS68KS TEST            ;Set the default hex file name to
                        ;"TEST.ABS"

ECS68KS TEST.HEX        ;Set the default hex file name to
                        ;"TEST.HEX"

ECS68KS -19200 -COM2 -DLPT3 -LLN%u TEST ;All options used
```

Note:

When using ECS68KS, the HMI series 200 emulator's Main serial port switches must be set for the autobaud mode with 8 data bits and no parity. This is how ECS68KS will set up the computer's COM port.

Environment Variables:

ECS68KS recognizes the following DOS environment variables:

ECS68KBAUD ECS68KPORT ECS68KLINE ECS68K_LPT

These environment variables are used to change the default settings of the ECS68KS command line switches.

The ECS68KBAUD environment variable is used to set the default serial communications port baud rate. The legal values which this variable can be set to are the same values defined for

the '-baud' command line switch; namely 38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 150, 110, and None.

The ECS68KPORT environment variable is used to set the default serial communications port. The legal values for this environment variable are: COM1, COM2, and AUX. (The same as the '-port' command line switch.)

The ECS68KLINE environment variable is used to set the default line number symbol format. This format must be given as a scanf string which will decode exactly one 2-byte integer value.

The ECS68K_LPT environment variable is used to set the default parallel download port. The legal values for this variable are the same as for the '-Ddownloadport' command line switch, namely: LPT1, LPT2, and LPT3.

Environment String Examples:

The following environment strings change the ECS68KS port, baud rate, line number symbol format, and download port from their default values of AUX, 9600, L_%u, and LPT1, respectively, to the new default values of COM2, 19200, LN%u, and LPT2:

```
ECS68KPORT=COM2
ECS68KBAUD=19200
ECS68KLINE=LN%u
ECS68K_LPT=LPT2
```

Note that these new defaults can still be overridden by the appropriate command line switches.

ECS68KS commands:

Using ECS68KS in conjunction with an HMI-200-68K emulator provides the following additional commands to the emulator's command set:

General Commands

+ [filename]	Open the named file and capture all screen output into it.
++ [filename]	Open the named file and append all screen output to it.
?	Display the emulator's help screens, with

	enhancements.
EXIT	Exit the source debugger. Return to the operating system.
N [filename][,ABCDHSMZ]	Establish a default filename for emulator RD and W commands, with option control flags.
RD [bias][,filename][,ABCDHSMZ]	Read a hex file, with option control flags.
W saddr,faddr[,filename][,HS]	Write a hex file, with option control flags.

Batch File Commands

< [filename]	Open a batch file for execution by the emulator.
> [filename]	Open a keyboard capture (macro) file. This file will record all keystrokes for later replay as a batch file.
/ [comment]	Pause or Resume execution of a batch file.
:: label	Mark a label in a batch file.
; [comment]	Batch file comment.

Symbol Table Manipulation Commands

: [symbol]	Show the value of the named symbol (or all symbols if none is named).
: symbol=	Delete the named symbol from the symbol table.
: symbol=value	If the named symbol exists, assign it the given value.
: [(type)]symbol=value	If the named symbol does not exist, assign it the given value and type.
:=C [module(s)]	Clear the named module(s) (or all modules if none is named) from the symbol table.

Source Debug Control Commands

:=A [module(s)]	Set the named module(s) (or all modules if none is named) to assembly level debug mode.
:=M [module(s)]	Set the named module(s) (or all modules if none is named) to mixed debug mode (assembly and source).
:=S [module(s)]	Set the named module(s) (or all modules if none is named) to source-only debug mode.
:=D [module(s)]	Display the source debug information for the named module(s) (or all modules if none is named).

DOS Commands

d:	Change the default (logged) drive.
CHDIR [dir]	Perform the DOS CHDIR function
DIR [filename]	Perform the DOS DIR function.
TYPE filename	Perform the DOS TYPE function.

Notes:

[...]	Bracketed items are optional.
filename	A DOS file name, with optional drive, directory path, and extent. DIR accepts wildcard filenames.
bias	A hex file read bias, to be added to the load address of all the data records of a hex file.
saddr	The starting address of a hex file write.
faddr	The ending address of a hex file write.
comment	A comment, ending with a carriage-return, line-feed sequence (CRLF).
label	A batch file label, starting with the first non-space character following the "::" and ending with the first space character seen after that.
symbol	A user-defined program symbol, or symbol path.

value A constant, symbol, or expression.

module A user-defined program module.

d A DOS disk drive letter.

dir A DOS directory.

ECS68KS command details:

General Commands

+ [filename]

The '+' command opens the named file, and causes ECS68KS to route all information destined for the screen to this file as well as the screen.

Output to the '+' file continues until a new '+' file is opened, ECS68KS is exited, or a '+' command is encountered without a filename. In the latter case, the current '+' file is closed without a new one being opened.

The '+' file's filename is formatted as usual with DOS. No default extent is assumed for the file name.

++ [filename]

The '++' command is functionally identical to the '+' command, with the following exception: The named file is opened, and all screen output is appended to the existing contents of this file; rather than the '+' command's behavior, which is to overwrite the existing contents (if any) of the named file.

?

The emulator's '?' command is enhanced by ECS68KS to include a short summary of the enhanced emulator commands available through ECS68KS.

EXIT

The 'EXIT' command causes ECS68KS to terminate operations, close any existing open files, and return to the operating system.

N [filename][,ABCDHSMZ]

The 'N' command Names a default file for use with subsequent emulator 'RD' and/or 'W' commands. The letters ABCDHSMZ represent the following individual option control flags:

A Download hex files to the emulator using ASCII file transfer.

B Download hex files to the emulator using Binary file transfer.

C Concatenate any symbols found reading a hex file with those which already exist in the internal symbol table (i.e. do not clear the symbol table on a 'RD' command).

D Use only the digits presented when looking up a symbol by value for display. (Opposite of 'Z'.)

H Upload/Download only the hex data records present in a hex file. Ignore all symbol records.

S Upload/Download only the symbol records of a hex file. Ignore all hex data records.

M Treat a file containing 'S' records as a Hitachi hex file. (Otherwise, a file containing 'S' records is considered to be a Motorola hex file.)

Z Pad a number with leading zeros when doing a lookup by value for symbol display. (Opposite of 'D'.)

1 Establish LPT1 as the new default parallel download port.

2 Establish LPT2 as the new default parallel download port.

3 Establish LPT3 as the new default parallel download port.

The default option control flag settings are "BZ".

A word about the 'D' and 'Z' option controls:

These two option control flags control the algorithm used to lookup a symbol by value for display when disassembling code. Their use is best illustrated by the following example:

Let us say that the symbol table contains the symbol SYM100, with the associated value $00000100, and the emulator is going to disassemble the following instruction:

```
MOVE.L #00,D0
```

Only two digits are presented following the '#' character: Zero and zero. As it happens, the last two digits of the value of SYM100 are also zero and zero, therefore, with the 'D' option selected, the following will be displayed:

```
MOVE.L #00:SYM100,D0
```

However, with the 'Z' option selected, the value 00 will be internally padded with leading zeroes and converted to $00000000, which does not match the $100 value assigned to SYM100; therefore, the following will be displayed:

```
MOVE.L #00,D0
```

RD [bias][,filename][,ABCDHSMZ]

The 'RD' command, which can be given on the emulator's command line or through the emulator command menu, causes ECS68KS to open the named file and read it as a series of hex and/or symbol records.

Unless inhibited by an option control, hex records are downloaded to the emulator, and symbol records are read into the ECS68KS internal symbol table. Hex records will be transferred in binary, unless the 'A' option control is used. Symbol records overwrite existing symbol table entries (if any), unless the 'C' option control is used.

If the optional read bias value is given, this bias will be added to the load address of all hex records downloaded.

If a filename is not given for the 'RD' command, it will use the last file name given to any preceding 'RD', 'W', or 'N' command. The default extent for a file name is ".ABS".

'bias' can be a hexadecimal number, a symbol, or an expression.

RDP [bias][,filename][,ABCDHSMZ123]

The 'RDP' command, which can be given on the emulator's command line only if the emulator is equipped with a parallel download port, functions in all respects as the regular 'RD' command, with the following exception: All hex records downloaded to the emulator are transmitted via the computer's parallel port, for greater speed in downloading.

W saddr,faddr[,filename][,HS]

The 'W' command, which can be given on the emulator's command line or through the emulator command menu, causes ECS68KS to open the named file and write a series of hex and symbol

records to it.

Unless inhibited by an option control, hex records covering the range of memory addresses from 'saddr' through 'faddr' are uploaded from the emulator and written to this file in Motorola 'S' record format. Additionally, the contents of the ECS68KS internal symbol table will also be written to this file.

If a filename is not given, the 'W' command uses the last file name given for a 'RD', 'W', or 'N' command; lacking one of these, it will use the default file name established on the ECS68KS command line. The default extent for this file name is ".ABS".

'saddr' and 'faddr' can be a hexadecimal number, a symbol, or an expression.

Batch File Commands

< [filename]

The '<' command opens the named file for batch processing by the emulator. The contents of this file are presented to the emulator as if they had been typed in at the keyboard. Any emulator command may be given in a batch file.

A '<' command given in a batch file causes ECS68KS to close the current batch file and open the new file for batch processing. Batch file processing does not return to the old batch file when the new file finishes. A '<' without a file name causes ECS68KS to close the current batch file without opening a new one.

Typing a carriage-return on the keyboard while a batch file is running will abort the batch file processing and return emulator control to the keyboard.

The default extent for a batch file name is ".BAT".

> [filename]

The '>' command opens the named file as a "Keyboard Capture File". All characters entered from the keyboard will be recorded in this file for later replay as a batch file, until another '>' command is entered, or the ECS68KS program is exited.

A '>' command entered while a keyboard capture file is open causes ECS68KS to close the current keyboard capture file and open the new one. A '>' without a file name closes the current capture file without opening a new one.

The default extent for a keyboard capture file is

".BAT".

/ [comment]

The '/' command Pauses the execution of a batch file (if one is running), or Resumes execution of a batch file (if one is Paused).

Once a batch file is paused, emulator control reverts to the keyboard, where it remains until a '/' command is entered, causing batch file execution to resume; or a '<' command is entered, to either close the current batch file or open and execute a new one.

If the 'comment' portion of a pause command starts with one or more decimal digit characters, then the batch file goes into a "Timed Pause", which will delay the execution of the batch file the given number of 1/10th seconds.

While a timed pause is in effect, typing the space bar will convert the timed pause into an ordinary (or "hard") pause, which will require the entry of a '/' command from the keyboard before batch file execution will resume. Entering any other character from the keyboard will cause immediate resumption of batch file processing.

:: label

The '::' command is treated as a comment command by the emulator and ECS68KS, and is used to mark a label in a batch file.

; [comment]

A ';' command causes the remainder of the line to be treated as a comment by ECS68KS and the emulator. This line is ignored.

Symbol Table Manipulation Commands

: [symbol]

The ':' command is used to perform a number of actions, all relating to the ECS68KS internal symbol table.

A ':' command by itself will cause ECS68KS to display the entire contents of its internal symbol table.

A ':' command followed by a symbol, or a symbol path, will generate a display of the name and value of the first

matching symbol found.

A ':' command followed by an empty path (i.e. ':/'), will generate a display of all the modules named in the ECS68KS internal symbol table.

A ':' command followed by an incomplete symbol path (i.e. one consisting of just a module name), will cause the display of all the symbols associated with the named module.

A ':' command followed by the path delimiter character ('/') followed by a module name will display the contents of the named module, and rotate the list of modules in the symbol table such that the named module moves to the top of the list, just below the dummy module "**_None_**". Since all symbol table lookups are performed by ECS68KS on a "first come, first served" basis, this allows the user to decide just which module does come first.

When the ':' command is used to display portions of the ECS68KS internal symbol table, it will accept the wildcard characters '?' and '*'. A '?' character in a symbol name will match any character in that position in the name. A '*' character in a symbol name (which must be the last character in the name) will match any number of characters, starting with the position of the '*' character in the symbol name. All symbols which match the wildcard symbol specification will be displayed by ECS68KS.

: symbol=

A ':' command, followed by a symbol or a symbol path, followed by an equals sign, results in the deletion of the named symbol from the symbol table.

This command does not accept wildcards.

: symbol=value

A ':' command, followed by a symbol or symbol path, followed by an equals sign, followed by some numeric value, results in the assignment of the given value to the named symbol.

If the named symbol exists, its value is updated.

If the named symbol does not exist, it is created at the location indicated by symbol path (if any), and assigned the given value. If no symbol path is given for the creation of a new symbol, it is assigned to a dummy module named "**_None_**". This dummy module is always first in the ECS68KS symbol lookup search algorithm.

The value assigned to a symbol may be a hexadecimal

number, another symbol's value, or an expression.

: [(type)]symbol=value

This form of the symbol assignment command may be used to create new symbols and assign them a type, as well as a value.

If no type is given, the default type assigned is "AG".

The following types are defined. One from each column may be chosen:

Segment	Scope
P	G
C	L
D	
U	
A	
S	

These letters assign type information to a symbol as follows:

P - Program address symbol
C - Code address symbol (same as Program)
D - Data address symbol
U - Uninitialized data address symbol (for 'C' programs).
A - Absolute address symbol
S - Scalar (numeric) symbol

G - Global (public) symbol
L - Local symbol

:=C [module(s)]

There are a number of ':=' commands, most of them having to do with source level debug controls, however, the ':=C' command causes ECS68KS to delete the entire contents of its internal symbol table (if no module is named), or otherwise just delete the named module(s) and all its (their) contents from the symbol table.

Source Debug Control Commands

:=A [module(s)]

The ':=A' command sets the named module(s) (or all modules if none is named) to Assembly Level Debug mode.

This is the default mode for all modules when ECS68KS is started.

:=M [module(s)]

The ':=M' command sets the named module(s) (or all modules if none is named) into Mixed Debug mode.

Once a module has been enabled for mixed debug mode, that module's source file will be opened, and the source lines for that module will be inserted at appropriate points into the output of all commands which disassemble code (i.e. the 'L' [list] command and the 'SS' [single-step] command), all register displays, and the Trace Menu display.

:=S [module(s)]

The ':=S' command sets the named module(s) (or all modules if none is named) for Source-only Debug mode.

When source-only mode is selected for a particular module, all assembly language displays for that module will be suppressed. Only source code lines from that module's source code file will be shown. This applies to the output of the list command, the single-step command, and the Trace Menu displays.

Additionally, single-step counts are by source lines, rather than assembly language instructions, such that, for example, the command 'SS6' will single-step through six source code lines, rather than six assembly language instructions (and as such represents single-stepping through an indeterminate number of assembly language lines).

:=D [module(s)]

The ':=D' command displays source debug information for the named module(s) (or all modules if none is named).

Source debug information consists of the line number map acquired by ECS68KS through examination of the module's line number symbols (whose format is set with the ECS68KS command line switch '-L'), the module's source code file name, and the current debug mode set for that module (Assembly, Mixed, or Source-only).

A word about the ':=M' and ':=S' commands:

When the ':=M' and ':=S' commands are used, they must be able to locate and successfully open each named module's source code file.

These source code files are assumed to reside in the current

directory, and their names are assumed to be the module's name, with the extent of ".C".

If any of these assumptions are false, ECS68KS will prompt for a source code file name, giving the name of the module for which it cannot find the source code file.

If the CHDIR command is to be used, then the prompt for the source code file name should be answered with the full path name of the file (and drive designator).

If no source code file is available, entering no file name at the prompt will cause ECS68KS to set that module back to Assembly mode debug (which requires no source code file), and continue on to the next module to be set to Source-only or Mixed debug mode.

DOS Commands

The following DOS commands are made available for use while ECS68KS is executing:

d:

Change the current logged disk drive.

CHDIR [dir]

Perform the DOS CHDIR function, with the usual DOS arguments and restrictions.

DIR [filename]

Perform the DOS DIR function, with the usual DOS arguments and restrictions.

Note that the DIR function will accept wildcard file names.

TYPE filename

Perform the DOS TYPE function, with the usual DOS arguments and restrictions.

A word about the DOS commands:

The DOS commands are provided by executing a copy of COMMAND.COM as a child process of ECS68KS. These commands will not run if the ECS68KS internal symbol table occupies too much

memory, or if COMMAND.COM cannot be found.

Since these commands run as a separate process from the ECS68KS program, their output will not appear in the screen capture files produced by the '+' and '++' commands.

Using symbols in commands:

Once a symbol table has been loaded into ECS68KS with the 'RD' command, or some symbols have been created with the ':' command, those symbols become available for use when entering numeric values into ECS68KS and emulator commands.

To use a symbol (or a mathematical expression involving symbols) in a command, prefix the symbol (or expression) with a ':' character at the point in the command where the symbol's value is to be used. For example, to begin listing code from the address given by the value of the symbol SYM100, use the following command:

```
L:SYM100
```

In a menu, symbols may also be used to enter numeric values into fields which are capable of accepting them. To enter a symbol's value into a field, position the cursor to any position in that field, and enter a ':' character. The cursor will then move to the bottom of the screen and echo the ':' as a prompt. Now enter the symbol. After the symbol is entered, a carriage-return will return the cursor to the field, and the symbol's value will be entered there.

Once Mixed or Source-only debug mode is activated for a particular module, that module's line numbers become available as special symbols. To use the address of a particular line of source code in a command, use the line number desired as if it were a symbol, and prefix this line-number-symbol with the name of the module it is to come from, separated from the line number with a '/'. For example, to begin listing code from line number ten in the module "GORF", do the following:

```
L:GORF/10
```

Line numbers are decimal numbers.

Symbol Path

A various points in this document, the phrase "symbol path" has been used. In general, a symbol path and a symbol name are synonymous. A symbol path consists of the following:

[module/]symbol

The symbol portion of a symbol path is required. The module portion is not, unless the symbol itself is not uniquely defined in the symbol table (i.e. it is a local symbol, and the same local symbol appears in more than one module).

When dealing with line numbers, it is best to always prefix the appropriate module name, since line numbers are all the same, and exist in practically every module.

An incomplete symbol path (one with a module name, but without the trailing '/' character nor symbol name) may be used only in the symbol display version of the ':' command.

Symbol Expressions

In addition to symbols, and symbol paths, symbol expressions may be used in any command where a symbol can be used (except on the left side of an '=' in the ':' command).

A symbol expression always starts with ":symbol" or ":module/symbol". After the initial symbol, a '+' character (for addition) or a '-' character (for subtraction) may be used to indicate the addition or subtraction of an offset from the preceding symbol's value. Following the '+' or '-' character comes a numeric value which is either a constant, or another symbol.

Constants are hexadecimal by default, unless one of the following characters is used to prefix the constant's value:

%	Binary override prefix.
@	Octal override prefix.
&	Decimal override prefix.
.	Decimal override prefix.
$	Hexadecimal override prefix.
0x	Hexadecimal override prefix.

Symbol Expression Examples:

```
:SYM100              ;SYM100 plus zero offset (just SYM100)
:GORF/SYM100         ;Same as above
:SYM100+10           ;SYM100 plus 10 hex
:SYM100+.16          ;SYM100 plus 16 decimal
:SYM100+%10000       ;SYM100 plus 10000 binary
:SYM100+@20          ;SYM100 plus 20 octal
:SYM100+:SYM100      ;SYM100 times 2
:SYM100-:SYM100      ;Zero
```

The IBM keypad and function keys

The following functions are performed by the keys on the IBM keypad:

Home	Generates a HOME character
End	Generates an ESC character
PgUp	Page Up (^R) in the Trace Menu
PgDn	Page Down (^C) in the Trace Menu
Left	Move Cursor Left
Right	Move Cursor Right
Up	Line Up (^W) in the Trace Menu, otherwise move Cursor Up (^E) in any menu
Down	Line Down (^Z) in the Trace Menu, otherwise move Cursor Down (^X) in any menu.
Insert	Toggle Insert Mode On and Off in the Command Menu
Delete	Delete Left character (^H) in line editor; move Cursor Left in any menu.
Control-PgUp	Move to Top of Buffer (^QR) in Trace Menu
Control-PgDn	Move to Bottom of Buffer (^QC) in Trace Menu
Control-Left	Move to Left Field Block (^A) in any menu
Control-Right	Move to Right Field Block (^F) in any menu

The following functions are performed by the function keys on the IBM keyboard:

F3	Repeat the last line entered
F5	Go to the CONFIG menu
F6	Go to the COMMAND menu
F7	Go to the EVENT menu
F8	Go to the SEQUENCE menu
F9	Go to the TRACE menu
F10	Go to the INTERFACE menu

SOURCEGATE USER'S MANUAL

FOR THE HMI-200-68000 EMULATOR

Huntsville Microsystems Inc.

August 13, 1990

Version 1.31

The contents of this document are believed to be accurate. However, Huntsville Microsystems, Inc. assumes no responsibility for the use of this document and issues no warranties, implied or otherwise, on the contents herein. Furthermore, Huntsville Microsystems, Inc. reserves the right to revise this publication and to make changes to the contents, without obligation to notify any persons of such revisions or changes.

Thank you for your purchase of SourceGate.

SourceGate is a window driven source level debugger designed to interface with the HMI in-circuit emulators. The combination of SourceGate and the HMI-200 series of in-circuit emulators is a high performance development system which combines the control of an in-circuit emulator with the power of a logic analyzer to provide a complete debugging environment for hardware and software on microprocessor based systems.

Organization of this Manual

Chapter 1 contains the installation procedures

Chapter 2 describes how to initialize the software

Chapter 3 describes the window interface

Chapter 4 is the Command Reference section

Chapter 5 contains a tutorial example

Appendices contain additional information about SourceGate errors along with information on using compilers and assemblers from various manufacturers.

The HMI 200 Series 68000 User's Manual provides more detail on the in-circuit emulator.

If problems should occur, contact HMI's technical support at (205)881-6005.

Note: Please fill out and return the SourceGate User Registration form on the following page. This will help to insure that you receive future updates to the SourceGate program.

Errata Sheet for HMI-200-68000 SourceGate Version 1.31

The following commands have not been implemented in version 1.31 but are listed in this manual or in the on-line help:

CNTRL-A

N filename, ABHMS

PM

RESTORE

SAVE

TR (68010 Only)

TRD (68010 Only)

TRE (68010 Only)

These commands will be added in future releases of SourceGate.

Note: This manual (including errata sheet) reflects the latest features of SourceGate version 1.31. If you have trouble executing a command, please reference this sheet and the SourceGate manual.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Chapter 1 Installation

1.0 Introduction

This chapter contains the setup and installation procedures for SourceGate. There are three steps in the installation process.

1. Unpacking the software
2. Make directory on system
3. Copy contents of diskettes

1.1 Unpacking the Software

SourceGate is shipped on three double sided, double density diskettes. Two of the diskettes contain the SourceGate files and demo programs and the third diskette contains object file converters for various compilers and assemblers. This "Object File Converter" diskette is provided free of charge to HMI customers to enable the user to develop software with the compiler of choice.

1.2 Make Directory on System

The files need to remain together in the same directory on the system. Make a directory on the system to copy the files into.

Example:

```
c>md HMI
c>cd HMI
```

Change to the directory so that the files can be copied into the directory.

1.3 Copy Contents of Diskettes

Copy the contents of the two SourceGate diskettes to the directory on the hard-disk:

```
Example:  HMI>copy a:sg68k.exe  <cr>
          HMI>copy a:68k.hlp <cr>  (6810.hlp for 68010 systems)
          HMI>copy a:68k.inx <cr>  (6810.inx for 68010 systems)
```

Do a directory to verify that all the files were copied to the HMI directory:

```
HMI>dir <cr>
     sg68k.exe
     68k.hlp
     68k.inx
```

Chapter 2 Initialization

2.0 Introduction

The communication software provides the communication between the emulator and computer system. SourceGate will operate only when an HMI-200 series in-circuit emulator is properly connected to the serial port of the host computer and the emulator is powered on.

Note: SourceGate allocates a number of temporary files when it is running. To ensure that your system has enough memory, it is recommended that all memory resident programs be removed from memory.

Note: The emulator will run up to 38.4K baud. It has been our experience that the 286 machines will support up to 19.2K baud and that the 386 machines will support up to 38.4K baud. Try lowering the baud rate if the emulator appears to have difficulty transferring the data.

Note: If you have been using the emulator with a terminal and want to switch to using a PC computer, the emulator needs to have a CONFIG command done on the command line of the terminal before going to the PC.

2.1 Initializing the Software

To initialize the software and enter the SourceGate program, the standard command line is formatted as follows:

```
SG68K [-baud] [-port] [-Llinesym]
```

All command line arguments, as indicated by the square brackets ('[' and ']'), are optional.

The command line arguments are as follows:

-baud Select communications port baud rate. Allowable selections are: N(No serial port initialization performed), 110, 150, 300, 600, 1200, 2400, 4800, 9600 (default), 19200, and 38400.

-port Select communications port. SourceGate automatically uses COM1 as its default communications port, unless COM1 is not available, in which case it will use COM2. The -port option will override the default SourceGate port selection and cause SourceGate to use only the port name. Allowable options are: COM1, COM2 and AUX.

-Llinesym In order to enable source-level debugging, SourceGate must know the relationship between the lines in the source file on disk, and the assembly language code in the emulator. This relationship is defined through the use of Line Number Symbols.

The default format for the line number symbols recognized by SourceGate is "LL%u". This is a scanf-formatted string indicating that line number symbols are those symbols which consist of two capital 'L' characters, followed by an unsigned decimal number. The unsigned decimal number is the line number. The value associated with the symbol is the starting address of the assembly-language instruction generated by that line of source code.

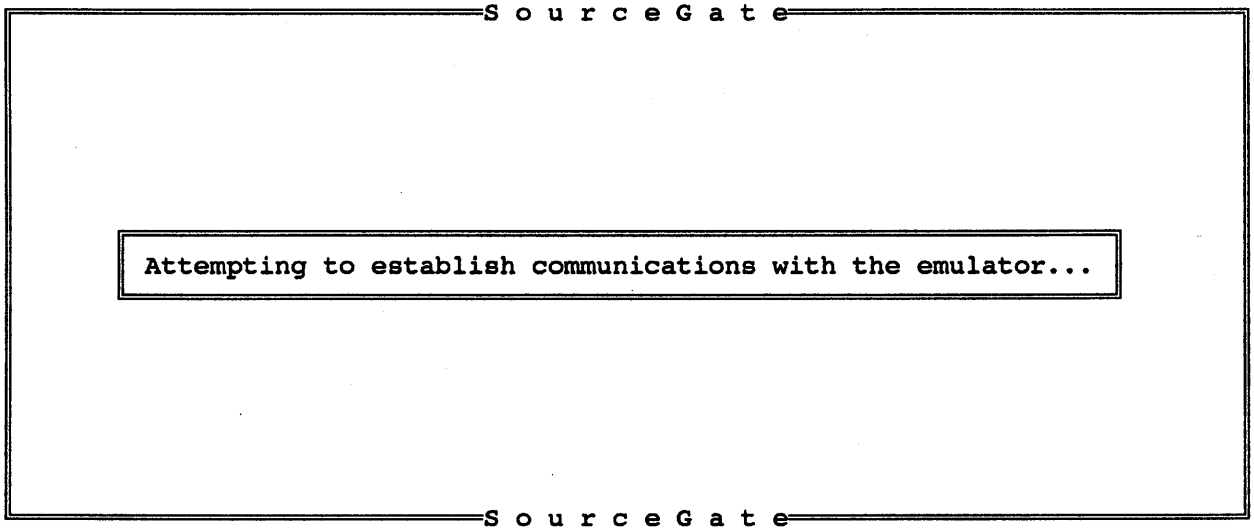
If, for example, your compiler produced line number symbols of the format LL_%u, then the -LLL_%u option should be used on the command line which initiates SourceGate. In this example, the -L tells SourceGate that the line number format is LL_%u. This command line switch allows source-level debugging using any compiler that produces line number symbols.

Note: To use SourceGate, the HMI 200 series emulator must be set up for the autobaud mode with 8 data bits and no parity (this is how SourceGate will initialize the COM port).

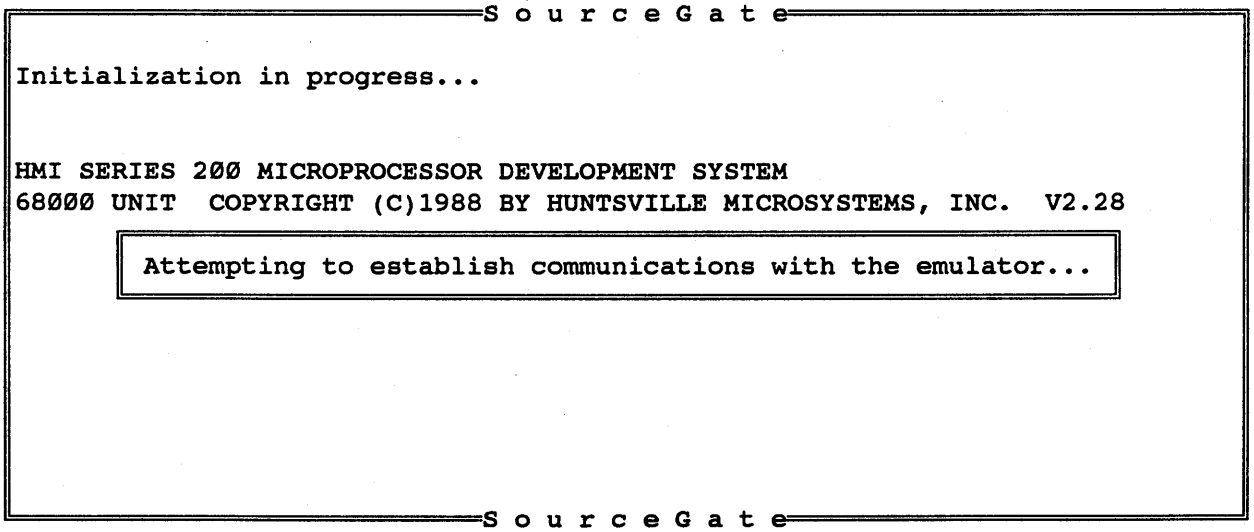
Examples:

```
SG68K                  = Start SourceGate
SG68K -19200           = Set 19200 Baud
SG68K -COM2            = Use COM2
SG68K -LLL_%u          = Set Line Number Symbol
                         format to "LL_%u"

SG68K -19200 -COM2 -LLL_%u = All Options used
```

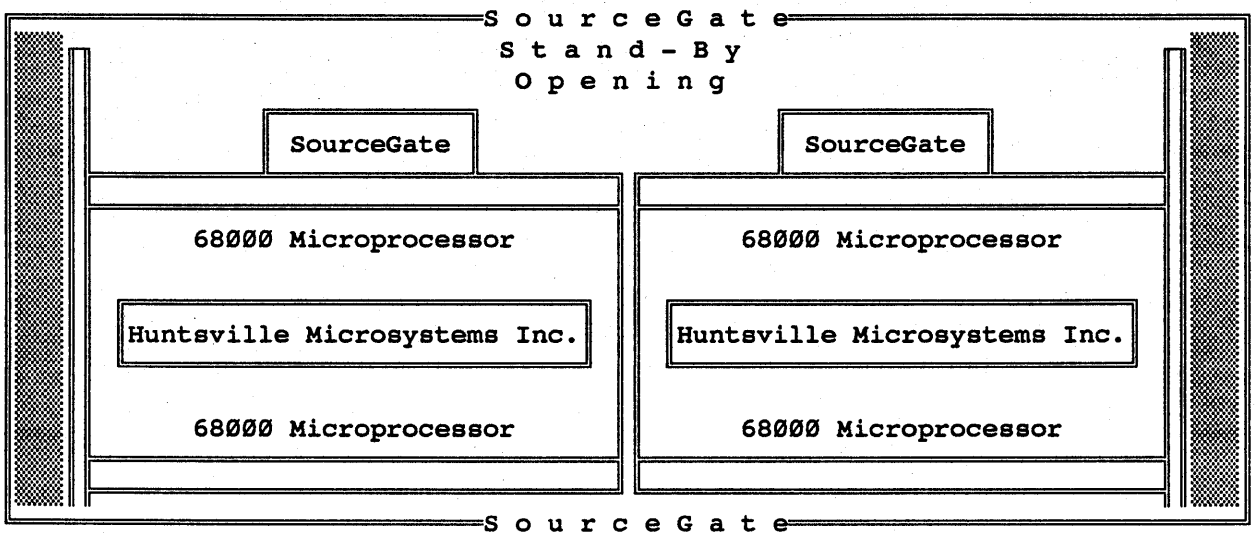
When SourceGate is started, it will display the following screen:



After communications have been established, the emulator will sign-on:



The screen will then display SourceGate "opening":



The messages "Setting Up Emulator" and "Reading Emulator Status" will appear. Once SourceGate is fully loaded, the Register Window and Command Window will appear along with a Status Window:

```
Status Window                ter Window  Alt(R)
For Help Information: Press F1  Ø
                             = 2709          NT S I7 NX NG NZ NV CY
DØ  = ØØØ4E71 D1  = 55550000 D2 = ØØØØØØØ1 D3 = ØØØØAAAA  VBR = ØØØØØØØØ
D4  = ØØØØØØØØ D5  = FFFFFFFF D6 = FFFFFFFF D7 = FFFFFFFF  SFC =         1
AØ  = 31FCØØØØ A1  = Ø6Ø442CØ A2 = 31FC5555 A3 = Ø6Ø44EF8  DFC =         Ø
A4  = ØØ2Ø31FC A5  = AAAAØ6Ø4 A6 = 4EF8ØØ2Ø A7 = ØØØØ1ØØØ

                          Command Window  Alt(C)
```

Chapter 3 Window Interface

3.0 Introduction

SourceGate is a true window driven software package. Windows can be popped up on the screen, moved and sized anywhere. This chapter describes all the windows and their functions.

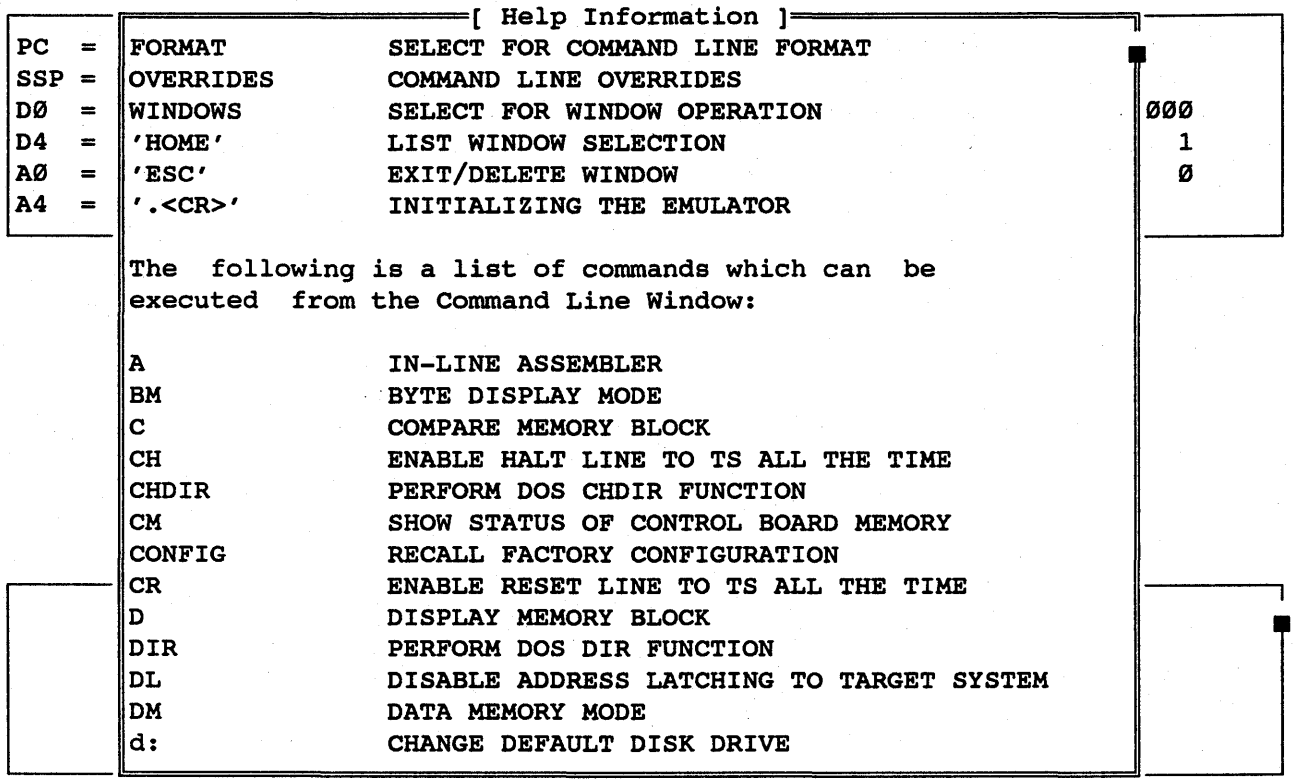
3.1 Description of Control Keys

The main keys to control the windows are the following:

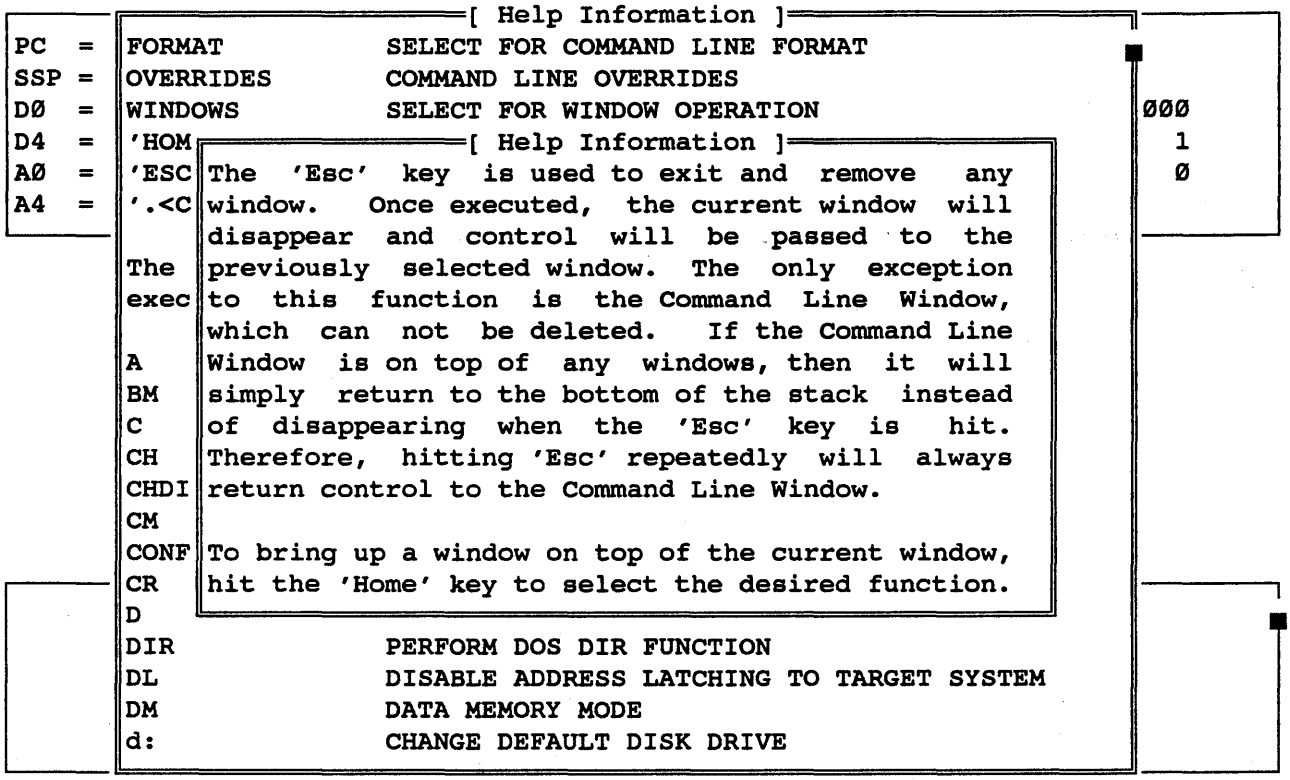
"HOME"	WINDOW SELECTION
"ESC"	DELETE WINDOW
"ALT"	AUTO SELECT
"ALT F1"	LIST VALID ALT KEYS
"F1"	HELP
"F2"	MOVE AND SIZE
"F4"	ZOOM WINDOW
"TAB"	ALL AVAILABLE WINDOWS DISPLAYED
"<CR>"	ENTER WINDOWS
CNTRL-A	PULL UP ALL WINDOWS
CNTRL-PgUp and CNTRL-PgDn	ROTATE THROUGH WINDOWS
CNTRL-@	DUMP COMMAND BUFFER
CNTRL-Return	DUPLICATE WINDOW
CNTRL-BREAK	EXIT SOURCEGATE
CNTRL-Home	CLEAR COMMAND WINDOW
SHIFT-TAB	DISPLAY PREVIOUS WINDOW
RESTORE	RESTORE CONFIGURATION FILE
SAVE	SAVE CONFIGURATION FILE
ALT-F2	MOVE SELECTION WINDOWS

3.1.1 The "F1" or "HELP" Key

Typing the "F1" key at any time will display a help window with text relative to the field or window from which it was requested. Typing the "F1" key while in the command window will display the main help window as shown below:



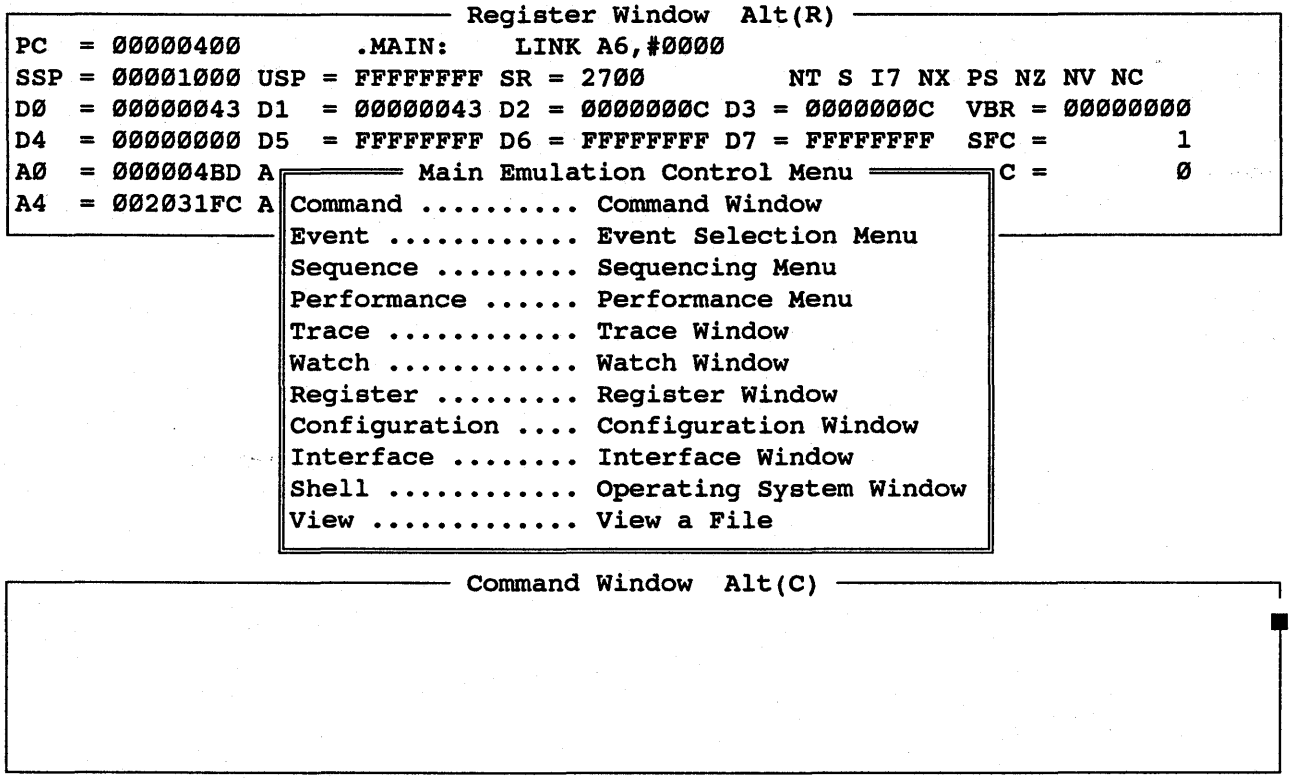
Some help windows have key-word fields from which additional help is available by typing the "F1" key again:



The arrow keys are used to scroll through the available text and the "ESC" key is used to exit the help window.

3.1.2 The "HOME" Key

Typing the "HOME" key will pull up a window selection list as shown below:



At this point, the current window will be left on the screen and a new window can be pulled up by selecting the appropriate window and entering <cr> or you can return to the previous window by typing the "ESC" key.

3.1.3 The "ESC" Key

Typing the "ESC" key will delete any window and return control to the previous active window. The only exception to this rule is that the Command Window cannot be deleted. Typing an "ESC" while in the Command Window will move that window to the bottom of the window stack, returning control to the previous active window.

3.1.4 The "ALT" Key

Typing any window's ALT key (found in the window's border) will also leave the current window on the screen, but will pull up the selected window without going through any window selection process. For example, to select the Register window press the "ALT" key along with R and it will take you to the Register window.

3.1.5 The "ALT F1" Key

Typing "ALT F1" will display a window that lists all valid ALT key combinations. These combinations correspond to the ALT keys that are shown in each window's border. Entering a <cr> on the desired window will pull up that window on the display.

3.1.6 The "F2" Key

Typing the "F2" key will display a help message at the bottom of the screen describing how to move or re-size the current window. Immediately after the F2 key is typed, the arrow keys can be used to move the highlighted window border. By typing the "END" key, the window can be resized to the maximum or minimum size which SourceGate will allow. Once the window is moved and sized to the desired position, the window is permanently placed by hitting the return key.

3.1.7 The "F4" Key

Typing the "F4" key will display two versions of a window. The version that was on the screen and the new version created after the window has been modified using the "F2" key. The window can be sized to be the full screen and then using the "F4" key zoomed in to a screen that contains a smaller version of that window, along with the other windows on the screen.

3.1.8 The "TAB" Key

The TAB key will display all the selections for the Event or Sequence window. By pressing TAB instead of return on Event or Sequence in the Main Emulation Control window, all 4 Event or Sequence Windows will be displayed at one time.

3.1.9 The <cr> Key

To enter one of the windows in the Main Emulation Control window, press the <cr> key and it will enter the selected window. For example, pressing return on the Trace window option will call up the Trace Window. For the Event, Sequence and Performance window, a selection window will appear and by pressing return on one of the options it will take you to the next window.

3.1.10 CNTRL-A

Typing a CNTRL-A while in a window selection menu will bring up all associated windows from the current cursor position down. This command can be used, for example, to bring up all sequence windows with one key stroke.

3.1.11 CNTRL-PgUp and CNTRL-PgDn

Active window control can be passed from the current window to the "next" or "last" window by typing the CNTRL-PgUp or CNTRL-PgDn keys. This gives the ability to cycle through the current windows on the screen without having to reselect a currently displayed window.

3.1.12 CNTRL-@

SourceGate offers a command buffer for type-ahead utilization. However, if too many keystrokes have been buffered causing the user to wait for command processing to end, the command buffer can be emptied by typing CNTRL-@ (or CNTRL-2).

3.1.13 CNTRL-Return

Duplicate windows can be displayed on the screen at the same time by typing CNTRL-Return instead of return while in the Main Emulation Control window. If the window already resides on the screen, then typing a return on the selected window name will bring that window to the top and become active. Typing CNTRL-Return will duplicate that window on the screen.

3.1.14 CNTRL-Break

Typing a CNTRL-Break will cause a direct abort of SourceGate and will not save any window modifications to the index file. Exiting SourceGate from the command window using the "EXIT" command will save any modified window positions and sizes in the SourceGate index file.

3.1.15 CNTRL-Home

Typing a CNTRL-Home will cause the command window to be cleared and the cursor will go to the top of the command window.

3.1.16 SHIFT-TAB

Typing in a SHIFT-TAB will display the previous window on the screen. For example, if in the Event A window and SHIFT-TAB is pressed, the main window will be displayed along with the preview window of A, B, C and D.

3.1.17 SAVE

The Save command will store onto disk the configuration file for the fields, registers, events, etc.

If changes are made when in SourceGate, the Restore command will load in the saved file.

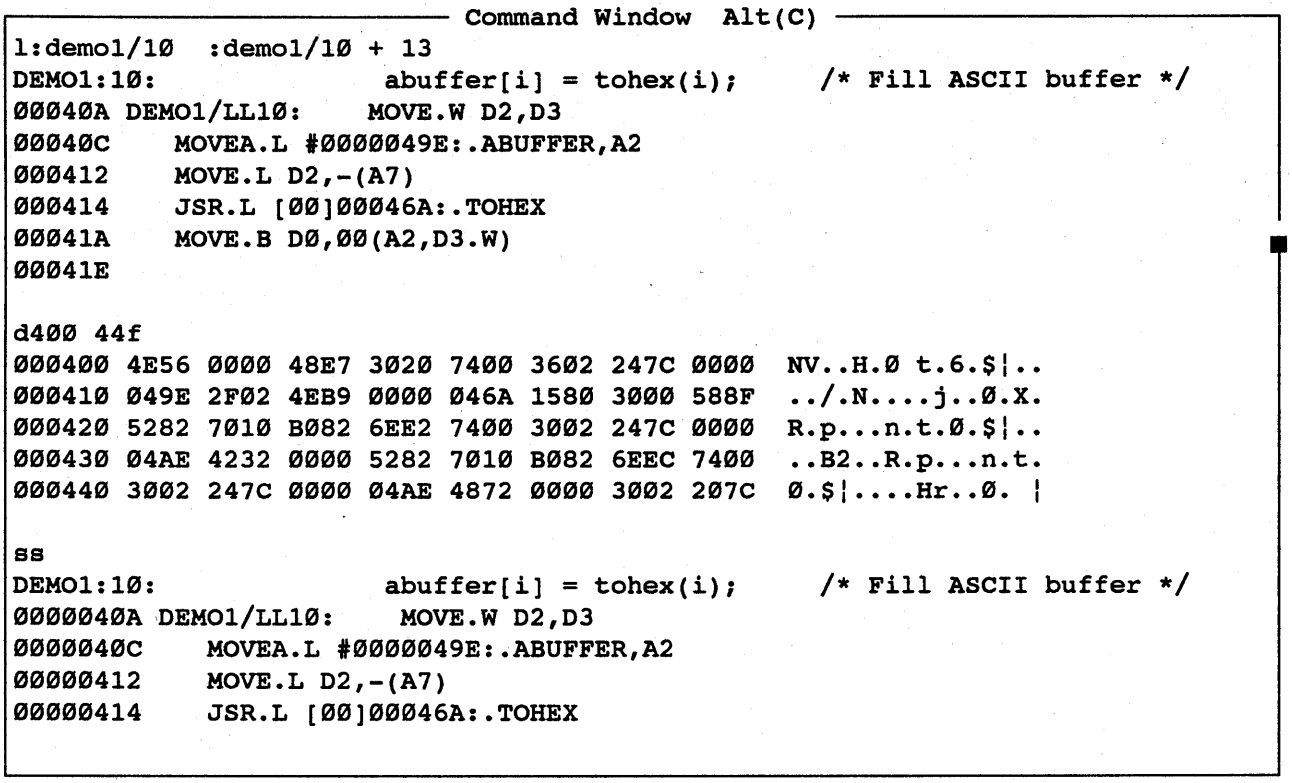
3.1.18 RESTORE

The Restore command will load in the file that was saved with the SAVE command.

3.1.19 ALT-F2

Typing in the ALT-F2 allows the selection window to be moved around on the screen.

3.2 The Command Window



The command window is the window from which the majority of emulator commands are executed. This is the only window which cannot be removed from the screen. While in this window, the command bar will be active. Commands which are typed in this bar are executed after a carriage return is entered. A command window history can be reviewed by moving the command bar back and forth in the command window using the arrow and PgUp/PgDn keys. A previous command can be re-executed by placing the command bar over it and hitting return.

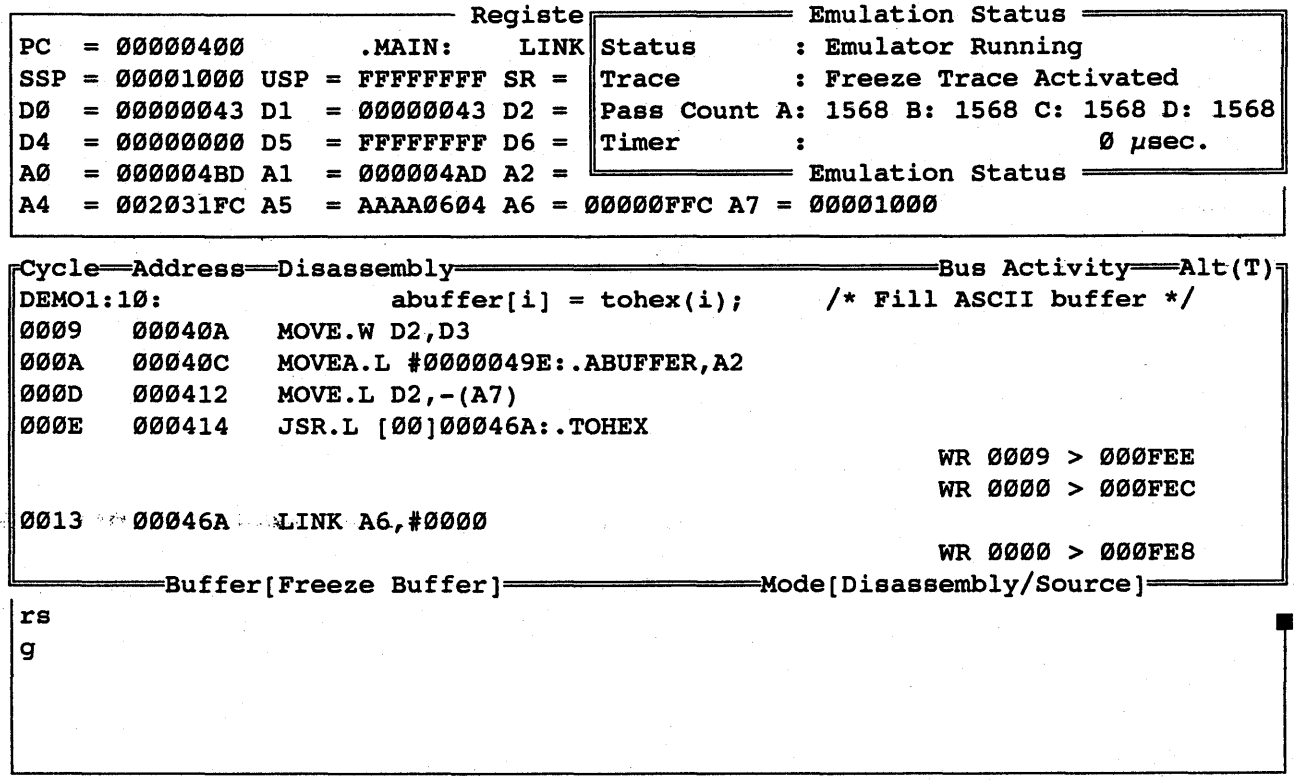
The following is a list of commands which can be executed from the Command Line Window:

A	ASSEMBLE
BM	BYTE DISPLAY MODE
C	COMPARE MEMORY BLOCK
CH	HALT ALWAYS ENABLED
CHDIR	PERFORM DOS CHDIR FUNCTION
CM	SHOW STATUS OF CONTROL BOARD MEMORY
CONFIG	RECALL FACTORY CONFIGURATION
CR	RESET ALWAYS ENABLED

D	DUMP MEMORY BLOCK
DIR	PERFORM DOS DIR FUNCTION
DL	DISABLE ADDRESS LATCHING TO TARGET SYSTEM
DM	DATA MEMORY MODE
d:	CHANGE DEFAULT DISK DRIVE
E	ENTER/CHANGE MEMORY
EH	ENABLE HALT DURING EMULATION ONLY
EL	ENABLE ADDRESS LATCHING TO TARGET SYSTEM
ER	RESET ENABLED DURING EMULATION ONLY
EXIT	RETURN TO THE OPERATING SYSTEM
F	FILL MEMORY BLOCK
G	BEGIN REAL-TIME EMULATION
I	INPUT COMMAND
L	LIST DISASSEMBLED CODE
M	MOVE MEMORY BLOCK
MT	TEST BLOCK OF MEMORY
N	ESTABLISH DEFAULT FILENAME FOR RD AND W
NH	DISABLE HALT LINE
NR	DISABLE RESET LINE
O	OUTPUT COMMAND
PD	PROGRAM WAIT STATES FOR DTACK COMMAND
PM	PROGRAM MEMORY MODE
QUIT	EXIT SOURCEGATE WITHOUT SAVING WINDOWS
RD	READ HEX FILE
RESET	RE-INITIALIZE EMULATOR
RS	RESET TARGET SYSTEM
SM	SUPERVISOR MEMORY MODE
SP	STOP PROGRAM EXECUTION
SR	SEARCH BLOCK OF MEMORY
SS	SINGLE STEP
ST	SHOW PROCESSOR STATUS
STF	TURN EMULATION STATUS WINDOW OFF
STI	SHOW EMULATION STATUS WINDOW DURING EMULATION
STO	TURN EMULATION STATUS WINDOW ON ALWAYS
TA	DEFINE TARGET ADDRESS
TR	SHOW STATUS FOR TRD/TRE
TRD	ADDRESS BUS TRI-STATE DISABLE
TRE	ADDRESS BUS TRI-STATE ENABLE
TYPE	PERFORM DOS TYPE FUNCTION
UM	USER MEMORY MODE
VER	DISPLAY VERSION NUMBER
W	WRITE MEMORY BLOCK
X	EXAMINE/CHANGE REGISTERS
:	SHOW/DELETE/EDIT SYMBOL VALUE
:=<ACDMS>	PERFORM SYMBOL MODULE FUNCTIONS (assembly, clear, display, mixed, source)
<	OPEN A BATCH FILE
!	TERMINATE AND STAY RESIDENT

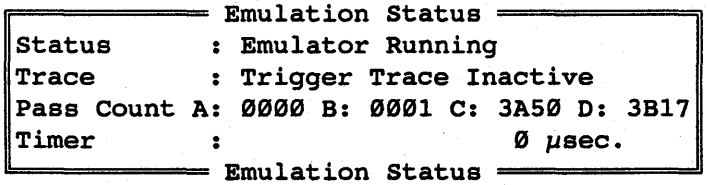
The Command Reference (Chapter 4) lists all the commands and gives a description of each.

3.3 The Emulation Status Window



The Emulation Status window will display messages regarding the status of the emulator, trigger trace, pass counters and interval timer. This window cannot be edited. The function of each field in the Emulation Status window is decribed as follows:

3.3.1 Status Field



The Status field will display messages regarding the status of the emulator. Messages such as "Emulator Running, Break Point Encountered, Stop Command Received", etc. will be displayed.

3.3.2 Trace Field

```
Emulation Status
Status       : Emulator Running
Trace        : Trace Trigger
Pass Count A: 0001 B: 60A0 C: 84C6 D: 858D
Timer        :                    0 µsec.
Emulation Status
```

The Trace field will display messages regarding the status of the trace. Messages such as "Trace Trigger" will be displayed. If the user is in the Trace window, and the "Freeze" command is executed, the Trace field will display the message "Freeze Trace Activated":

```
Emulation Status
Status       : Emulator Running
Trace        : Freeze Trace Activated
Pass Count A: 0001 B: EE3B C: F620 D: F61D
Timer        :                    0 µsec.
Emulation Status
```

3.3.3 Pass Count Field

```
Emulation Status
Status       : Emulator Running
Trace        : Trigger Trace Inactive
Pass Count A: 0001 B: 0002 C: 0003 D: 0004
Timer        :                    0 µsec.
Emulation Status
```

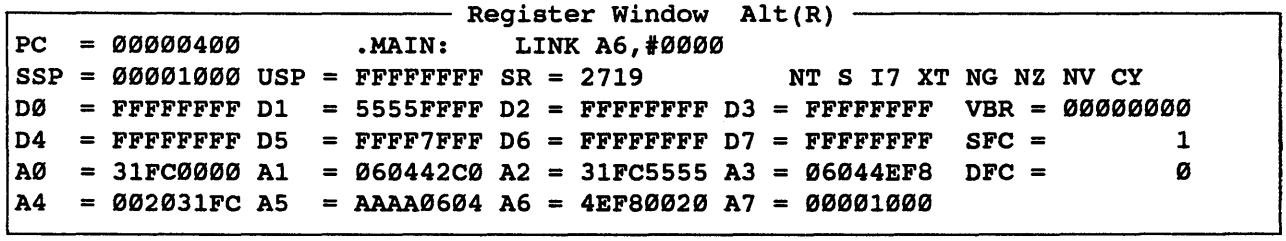
The Pass Count field shows the current pass count for each event, even if that event is not defined. Undefined events will have pass counts that show rapidly changing random numbers since undefined events consist of don't care conditions for all fields. The pass count fields are updated dynamically, so it is possible to watch the pass counts change for a certain event if that event occurs at a slow enough rate.

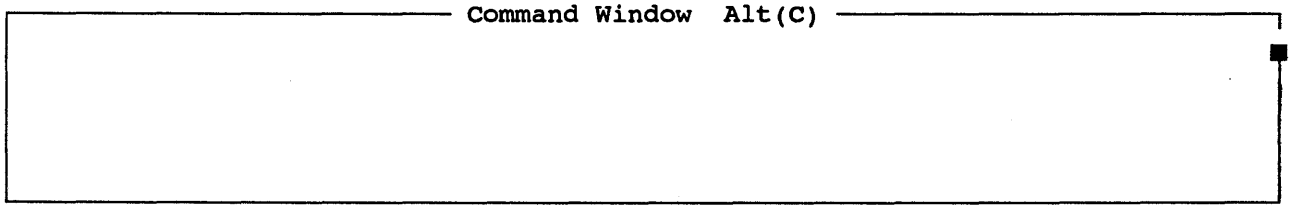
3.3.4 Timer Field

```
Emulation Status
Status       : Emulator Running
Trace        : Trace Trigger
Pass Count A: 0001 B: 0001 C: 6B59 D: 6BB9
Timer        :                 006 µsec.
Emulation Status
```

The Timer field shows the current reading obtained by the Interval Timer. If the "start" portion of the timer has been activated, then the timer value will be updated dynamically until the "stop" condition is reached. Therefore, it is possible to watch the timer value change for long timing loops.

3.4 The Register Window





The Register window shows the contents of the processor's registers. The register values are always accurate whenever emulation is not in progress, and are the last known value whenever emulation is in progress.

Registers can be altered in the register display window by placing the cursor on the desired register data field and entering a new value. Any time a register's value changes, the new value will be highlighted in the window until the next time that registers are updated.

The register window is automatically updated whenever the emulator is single-stepped, a breakpoint occurs, a stop command is executed or the emulator is reset.

Note: The Register Window examples shown are for the emulator running a 68010 processor. Certain fields (VBR, DFC, SFC, etc.) will not be shown when the emulator is running a 68000 or 68008 processor.

Each field in the Register window is described as follows:

3.4.1 PC Field

The Program Counter is a 32-bit register that contains the address of the next instruction to be executed by the processor. The Program Counter values can be modified by placing the cursor on the desired field and entering a new value.

The Program Counter Description field is the field right next to the PC field. This field shows the assembled instruction pointed to by the PC and is automatically updated by the emulator. This field cannot be edited.

3.4.2 SR (Status Register) Field

The Status Register is a 16-bit register in which each bit is used to indicate the results of the instruction just executed. The user byte containing the condition codes is the only portion of the Status Register information available at the user privilege level, and it is referenced as the CCR in user programs. In the supervisor privilege level, software can access the full Status Register, including the interrupt priority mask (three bits) as well as additional control bits. The Status Register values can be modified by placing the cursor on the desired field and entering a new value.

The individual condition codes are also displayed to the right of the hexadecimal SR value. An explanation of each condition code is listed below:

Bit Location in SR	Flag or Condition Code Name	Not Active Code	Active Code
15	Trace Mode	NT	TR
14	Not Defined	--	--
13	Supervisor/User State	U	S
12	Not Defined	--	--
11	Not Defined	--	--
10	Interrupt Priority Mask 2 ---\		
9	Interrupt Priority Mask 1 ----	I0 through I7	
8	Interrupt Priority Mask 0 ---/		
7	Not Defined	--	--
6	Not Defined	--	--
5	Not Defined	--	--
4	Extend	NX	XT
3	Negative	PS	NG
2	Zero	NZ	ZR
1	Overflow	NV	OV
0	Carry	NC	CY

3.4.3 SSP Field

The Supervisor Stack Pointer is a 32-bit register that contains the address of the next free location on the stack if the SSP is the active stack pointer. The SSP is only accessable at the Supervisor privilege level. The SSP value can be modified by placing the cursor on the desired field and entering a new value.

3.4.4 USP Field

The User Stack Pointer is a 32-bit register that contains the address of the next free location on the stack if the USP is the active stack pointer. The USP value can be modified by placing the cursor on the desired field and entering a new value.

3.4.5 VBR Field*

The Vector Base Register is a 32-bit register that contains the base address of the exception vector table in memory. The displacement of an exception vector is added to the value in this register to access the vector table. The VBR value can be modified by placing the cursor on the desired field and entering a new value.

3.4.6 D0 - D7 Fields

Registers D0 - D7 are 32-bit data registers that are used for bit, bit-field, byte, word, and long word operations. These register values can be modified by placing the cursor on the desired field and entering a new value.

3.4.7 A0 - A7 Fields

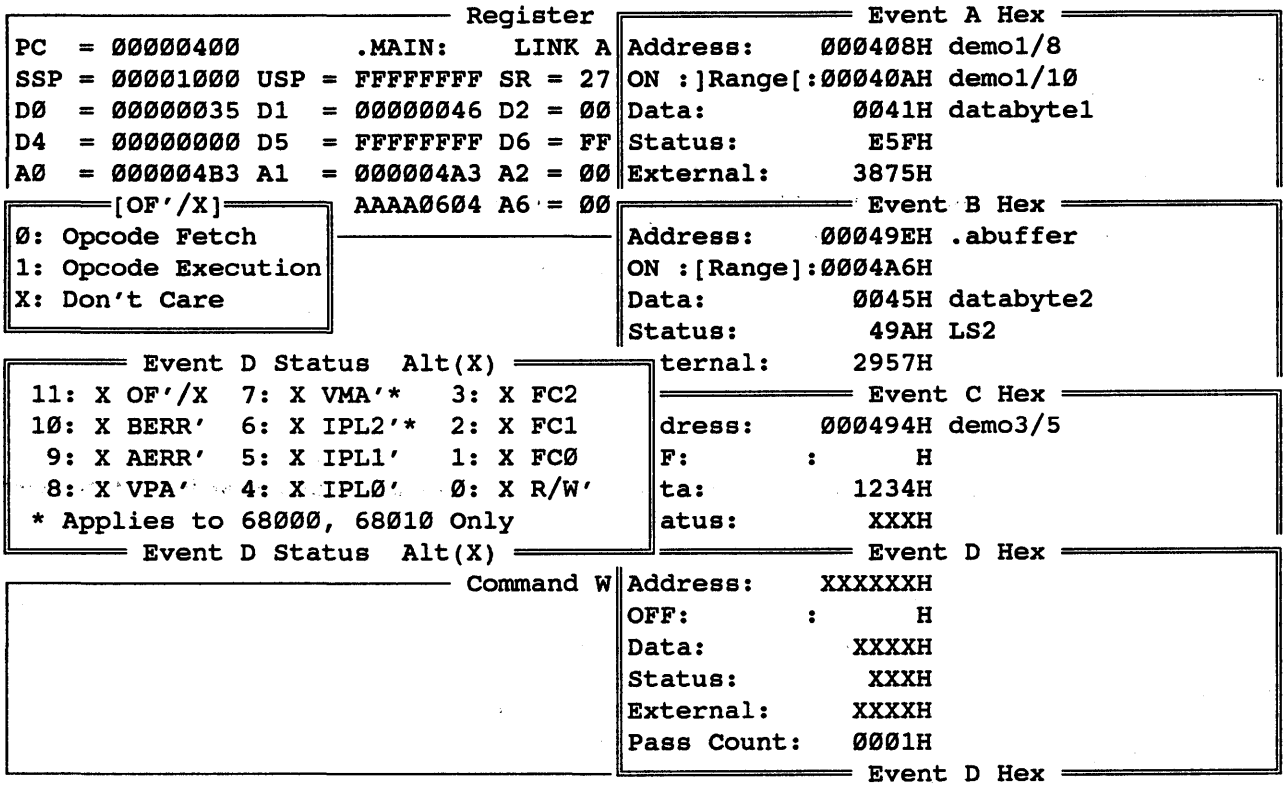
Registers A0 - A6 are 32-bit registers that may be used as software stack pointers or base address registers. Register A7 is a register designation that applies to the USP in the user privilege level, and to the SSP in the supervisor privilege level. These register values can be modified by placing the cursor on the desired field and entering a new value.

* - 68010 Only

3.4.8 DFC and SFC Fields*

The Alternate Function Code registers DFC and SFC contain 3-bit function codes. Function codes are automatically generated by the processor to select address spaces for data and program at the user and supervisor privilege levels, and a CPU address space used for processor functions (for example, coprocessor communications). Registers DFC and SFC are used by certain instructions to explicitly specify the function codes for operations.

3.5 The Event Windows

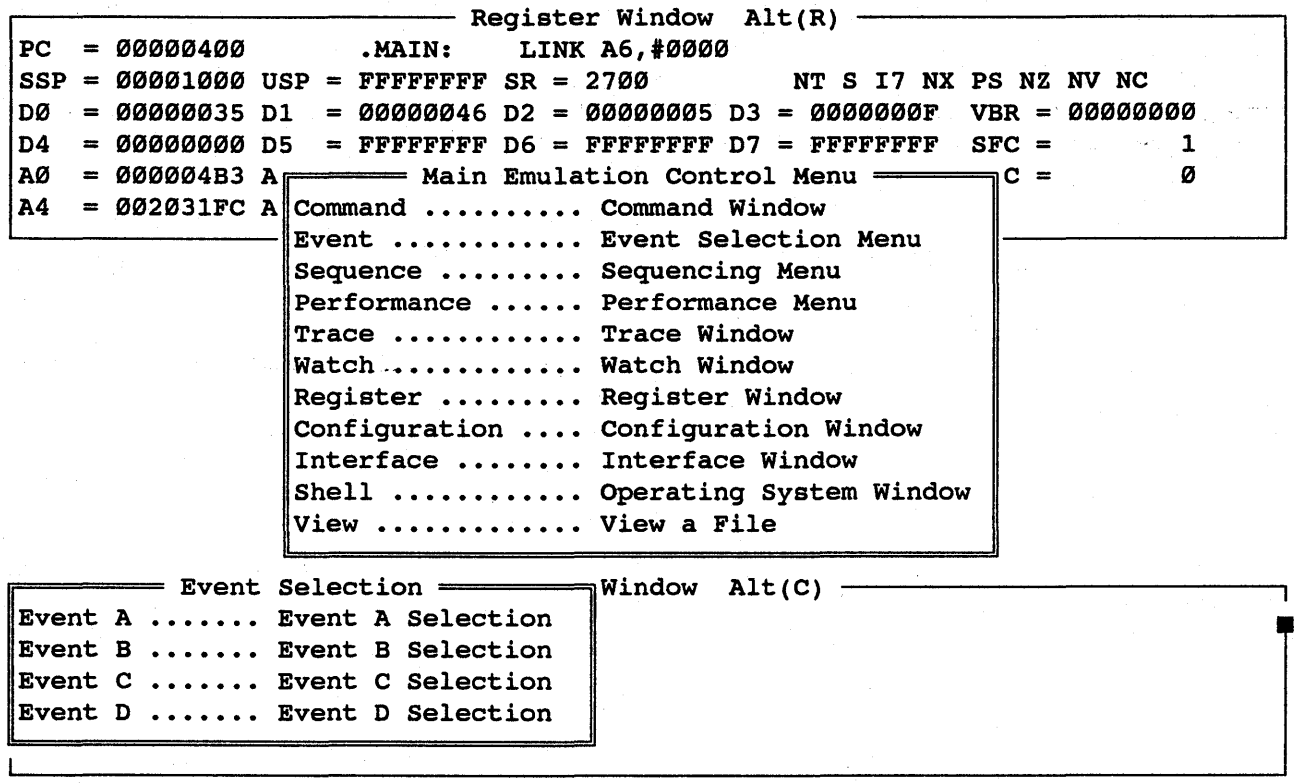


The Emulator Event Windows can be called up by pressing HOME and then TAB or <cr>. The Event windows are used to establish the parameters that define an Event. When emulation is initiated with the Go command, the system will utilize the configuration of the four events (A, B, C and D). Each Event can be defined as a combination of Address, Data, Internal status and External status. Each of these parameters can be defined in hex or binary (including don't-care conditions). There is also a pass-counter associated with each event, thus allowing each event to occur multiple times before triggering any functions as defined in the Sequence menu. An address range can also be defined for an event to break or trigger when the address is within or outside of a certain range of values (available if the Performance Analysis option is installed).

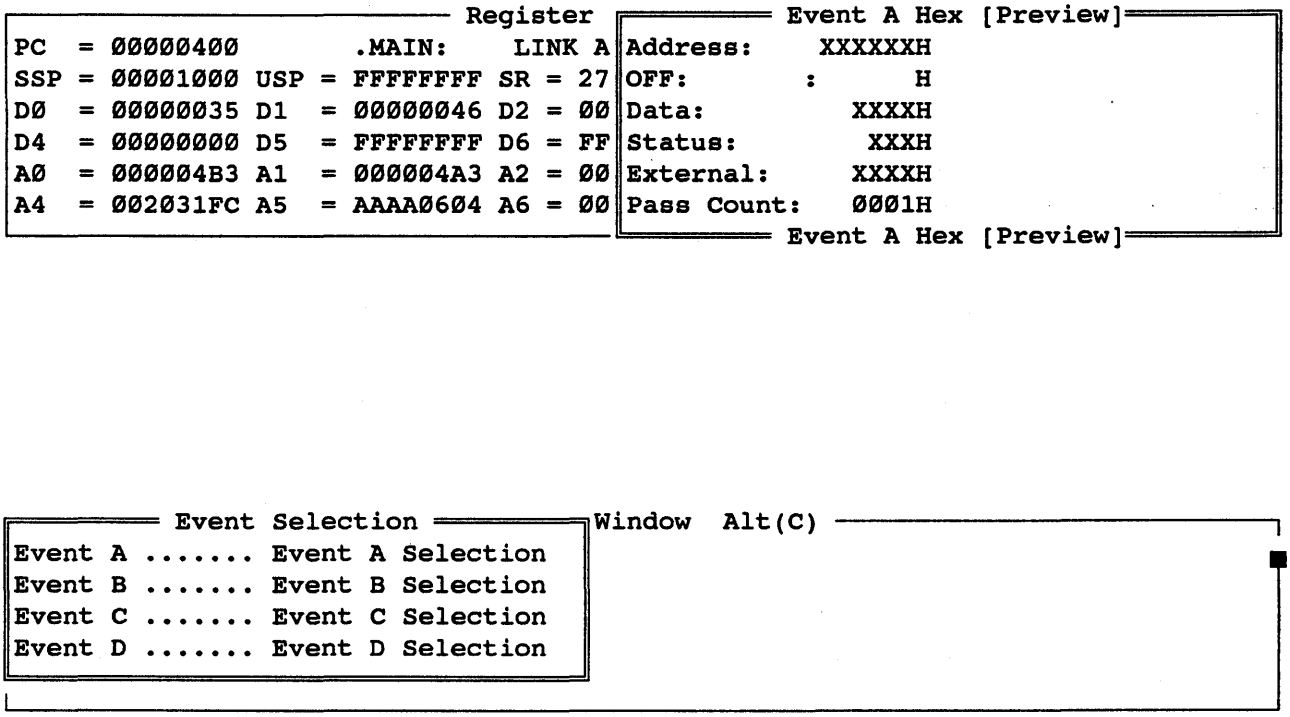
Typing the Home key at any point of the Event selection process will return control to the Main Emulation Control Window. Typing a Shift-Tab will return control to the last selection window.

3.5.1 Event Selection

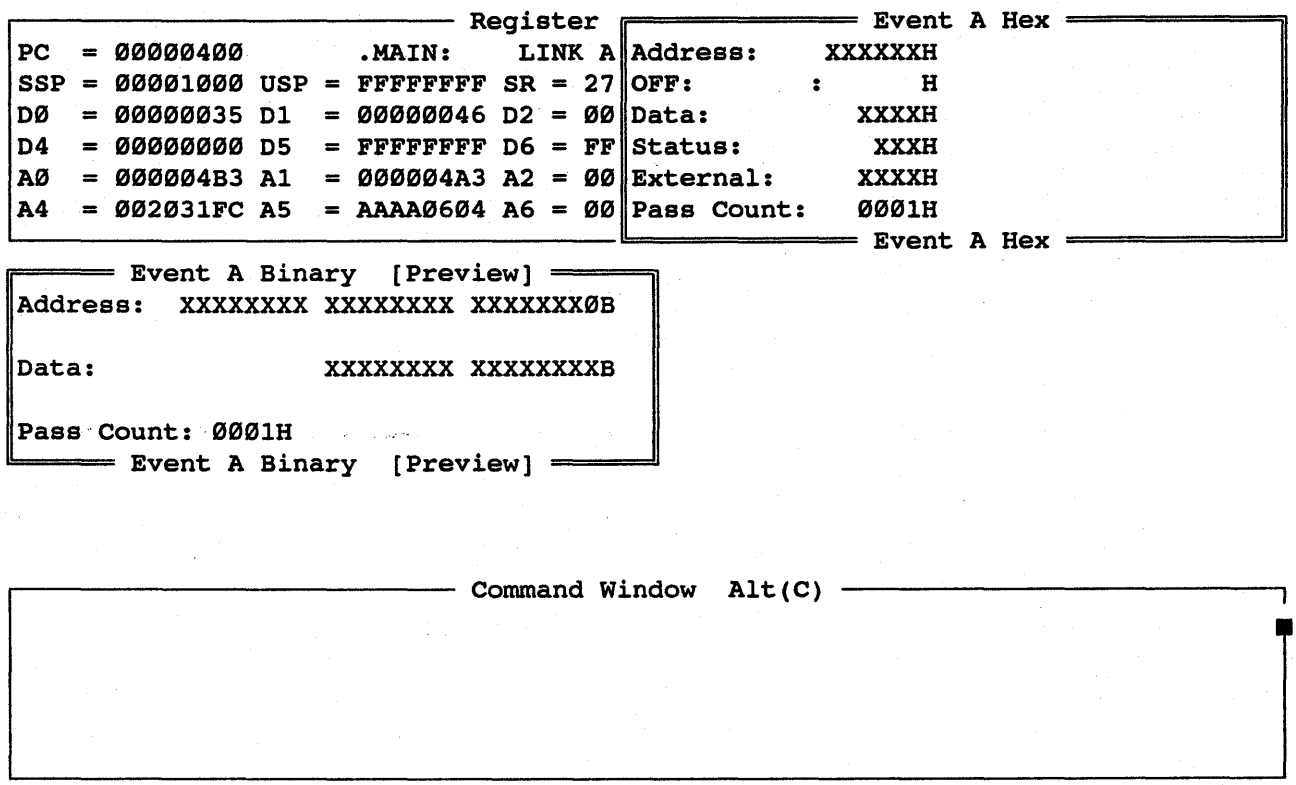
The emulator's four events have been broken down to smaller, compact windows with specific functions. Placing the cursor on the Event field in the Main Emulation Control Window will automatically show a "Selection" window for events A, B, C and D:



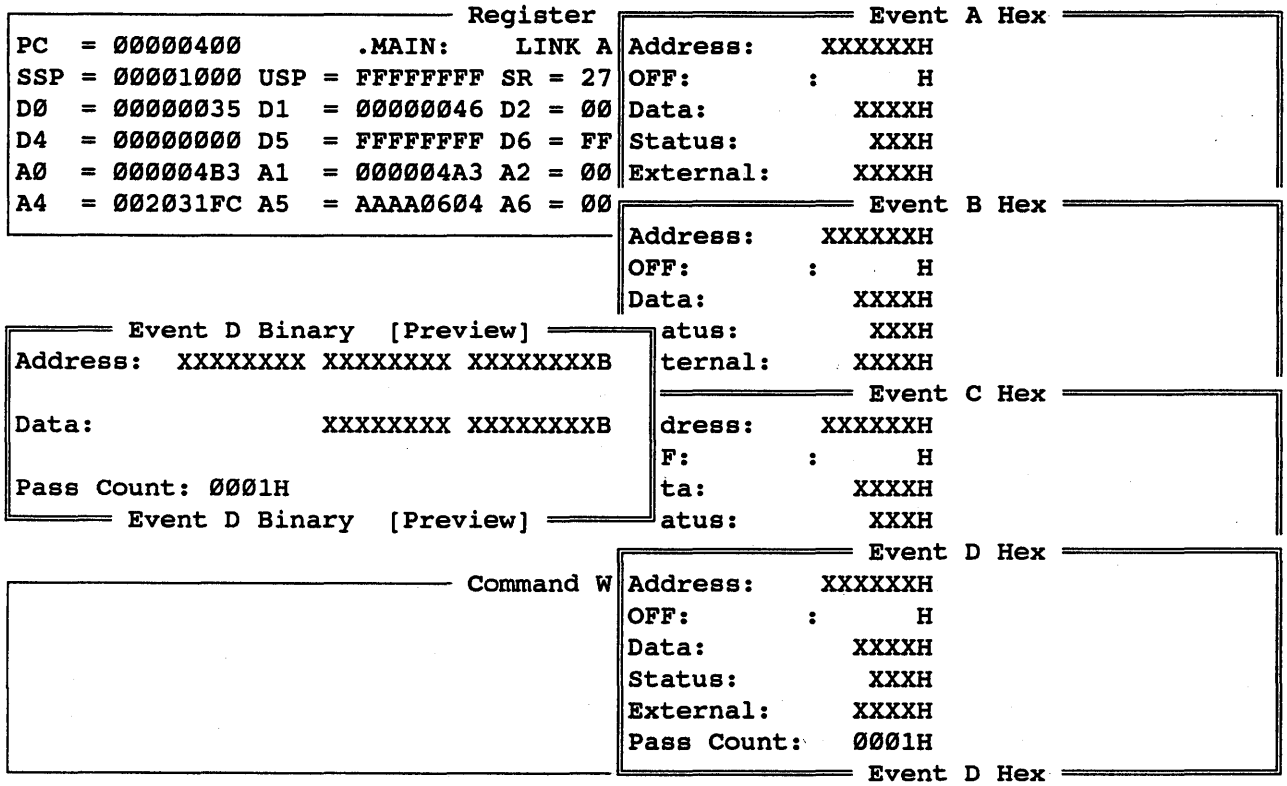
Typing a <cr> enters the Event Selection window with the corresponding Event Preview window appearing on the screen:



An additional <cr> will allow you to enter the selected Event window and the Event Selection window will disappear:



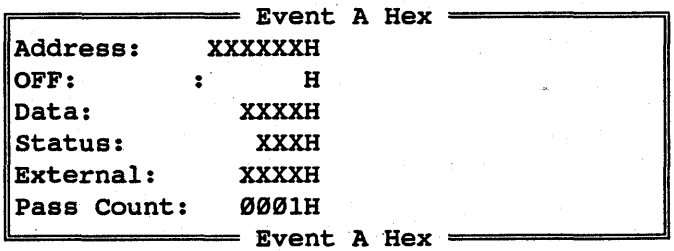
All four Event windows can be pulled up by pressing the TAB key while the cursor is on the Event field in the "Main Emulation Control" Menu. The current active event window will also have a Binary preview window displayed:

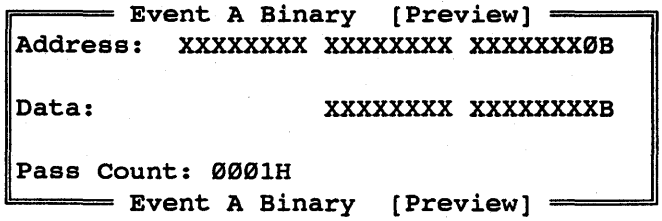


3.5.2 Event Window Fields

The Event windows are used to define a combination of Address, Data, Internal and External status, Bank information, Pass Count, and Port value parameters. Each of these parameters can be defined in hex or binary (including don't care conditions). Each field in the Event window is defined as follows:

3.5.2.1 Hex Address Field





The Hex Address Field allows hexadecimal addresses or symbols to be entered. While in the Hex Address field, a Binary Preview window will be displayed and will reflect the binary value of the hexadecimal address entered. To enter a Symbol instead of the absolute value, type a colon ':' to enter the symbol entry field. The symbol entry field is the highlighted field next to the H in the address field. For example, if symbol "Test1" has been defined to be equal to 428H, then typing :Test1 while the cursor is in the symbol field would automatically enter 428H in the address field.

Any hexadecimal digit (0-F) and (X) are allowable characters for the hexadecimal address field, where (X) represents the don't care.

3.5.2.2 Binary Address Field

```
Event A Hex
Address:    XXXXXXH
OFF:       :      H
Data:          XXXXH
Status:         XXXH
External:      XXXXH
Pass Count:    0001H
Event A Hex
```

```
Event A Binary  Alt(B)
Address:  XXXXXXXX XXXXXXXX XXXXXXX0B

Data:               XXXXXXXX XXXXXXXXB

Pass Count: 0001H
Event A Binary  Alt(B)
```

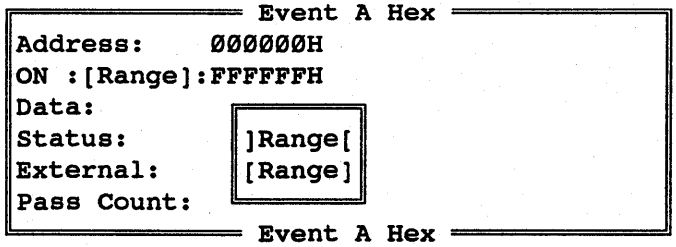
Pressing return while on the address field will display the binary address field. Any of the binary digits (0, 1 or X) are allowable characters for the binary address field, where (X) represents the don't care.

3.5.2.3 Address Range Field

The address range field can be enabled or disabled by selecting "ON" or "OFF" in the selection window:

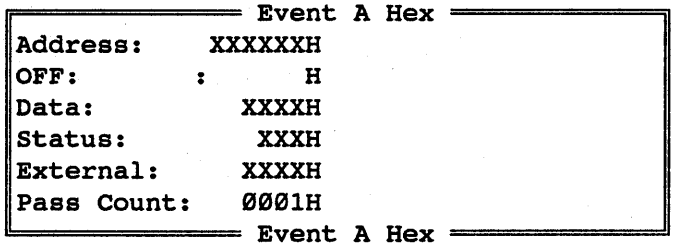
```
Event A Hex
Address:    XXXXXXH
OFF:       :      H          OFF
Data:          XXXXH          ON
Status:         XXXH
External:      XXXXH
Pass Count:    0001H
Event A Hex
```

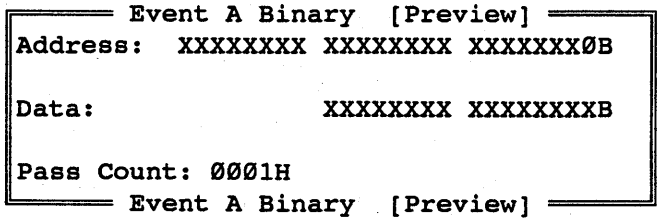
When the range field is turned "ON", then the Event can be set to break on addresses within a range ([Range]) or addresses outside the range (]Range[):



The beginning address should be placed in the Address field and the ending address should be placed in the Range field.

3.5.2.4 Hex Data Field





The Hex Data Field allows hexadecimal data or symbols to be entered. While in the Hex Data field, a Binary Preview window will be displayed and will reflect the binary value of the hexadecimal data value entered. To enter a Symbol instead of the absolute value, type a colon ':' to enter the symbol entry field. The symbol entry field is the highlighted field next to the H in the data field. For example, if symbol "Test2" has been defined to be equal to 55H, then typing :Test2 while the cursor is in the symbol field would automatically enter 55H in the data field.

Any hexadecimal digit (0-F) and (X) are allowable characters for the hexadecimal data field, where (X) represents the don't care.

3.5.2.5 Binary Data Field

```
                                    ═══ Event A Hex ═══
                                   Address:    XXXXXXH
                                   OFF:       :      H
                                   Data:         XXXXH
                                   Status:        XXXH
                                   External:     XXXXH
                                   Pass Count:   0001H
                                    ═══ Event A Hex ═══

═══ Event A Binary  Alt(B) ═══
Address:  XXXXXXXX XXXXXXXX XXXXXXX0B

Data:               XXXXXXXX XXXXXXXXB

Pass Count: 0001H
═══ Event A Binary  Alt(B) ═══
```

Pressing return while on the data field will display the binary data field. Any of the binary digits (0, 1 or X) are allowable characters for the binary data field, where (X) represents the don't care.

3.5.2.6 External Trace Bits Field

```
                                    ═══ Event A Hex ═══
                                   Address:    XXXXXXH
                                   OFF:       :      H
                                   Data:         XXXXH
                                   Status:        XXXH
                                   External:     XXXXH
                                   Pass Count:   0001H
                                    ═══ Event A Hex ═══

═══ Event A External  Alt(E) ═══
 15: X White/Purple  7: X Blue/Purple
 14: X White/Blue    6: X Blue/Blue
 13: X White/Green   5: X Blue/Green
 12: X White/Yellow  4: X Blue/Yellow
 11: X White/Orange  3: X Blue/Orange
 10: X White/Red     2: X Blue/Red
  9: X White/Brown   1: X Blue/Brown
  8: X White/Black   0: X Blue/Black
        Ground: Black/Black
(First color is wire/2nd color is tip)
═══ Event A External  Alt(E) ═══

═══ Event A Hex ═══
0: Logic Level Zero
1: Logic Level One
X: Don't Care
```

The External Trace Bits can be connected to any TTL level signal to provide up to 16 additional conditions on which to define an

event. Any hexadecimal digit (Ø-F) and (X) are allowable characters for the hexadecimal external-bits field, where (X) represents the don't care.

3.5.2.7 Pass Count Field

```
Event A Hex
Address:    XXXXXXH
OFF:       :      H
Data:         XXXXH
Status:        XXXH
External:     XXXXH
Pass Count:   ØØØ1H
Event A Hex
```

The pass counter defines the number of times that an event must occur before that event is taken to be valid. Valid pass counter values range from 1 to FFFFH.

3.5.2.8 Status Field

```
Event A Hex
Address:    XXXXXXH
OFF:       :      H
Data:         XXXXH
Status:        XXXH
External:     XXXXH
Pass Count:   ØØØ1H
Event A Hex
```

```
Event A Status  Alt(D)
11: X OF'/X   7: X VMA'*    3: X FC2
1Ø: X BERR'   6: X IPL2'*   2: X FC1
 9: X AERR'   5: X IPL1'    1: X FCØ
 8: X VPA'    4: X IPLØ'    Ø: X R/W'
* Applies to 68ØØØ, 681Ø Only
Event A Status  Alt(D)
```

```
[OF'/X]
Ø: Opcode Fetch
1: Opcode Execution
X: Don't Care
```

Certain status conditions can also make up an event. Active low signals are indicated by a "'" after their name. This means that

a zero should be entered to make them active and a one should be entered to make them not active. An "X" is the don't care condition which is valid for all fields. An explanation of each status bit follows:

R/W':

The R/W' status bit allows you to define a condition to be when a read occurs (R/W'= 1) or when a write occurs (R/W'= Ø).

FCØ, FC1, FC2:

The FCØ, FC1, and FC2 status bits allow you to define a condition to be when the function codes take on a certain value.

IPLØ', IPL1', and IPL2':

The IPLØ', IPL1', and IPL2' status bits allow you to define a condition to be when each of these signals is active (equal to Ø) or when they are not active (equal to 1).

VMA':

The VMA' status bit allows you to define a condition to be when this signal is active (VMA'= Ø) or when it is not active (VMA'= 1).

VPA':

The VPA' status bit allows you to define a condition to be when this signal is active (VPA'= Ø) or when it is not active (VPA'= 1).

AERR':

The AERR' status bit allows you to define a condition to be when the processor attempts to access a word or a long word operand or instruction at an odd address (AERR'= Ø). The inactive state of this signal is (AERR'= 1).

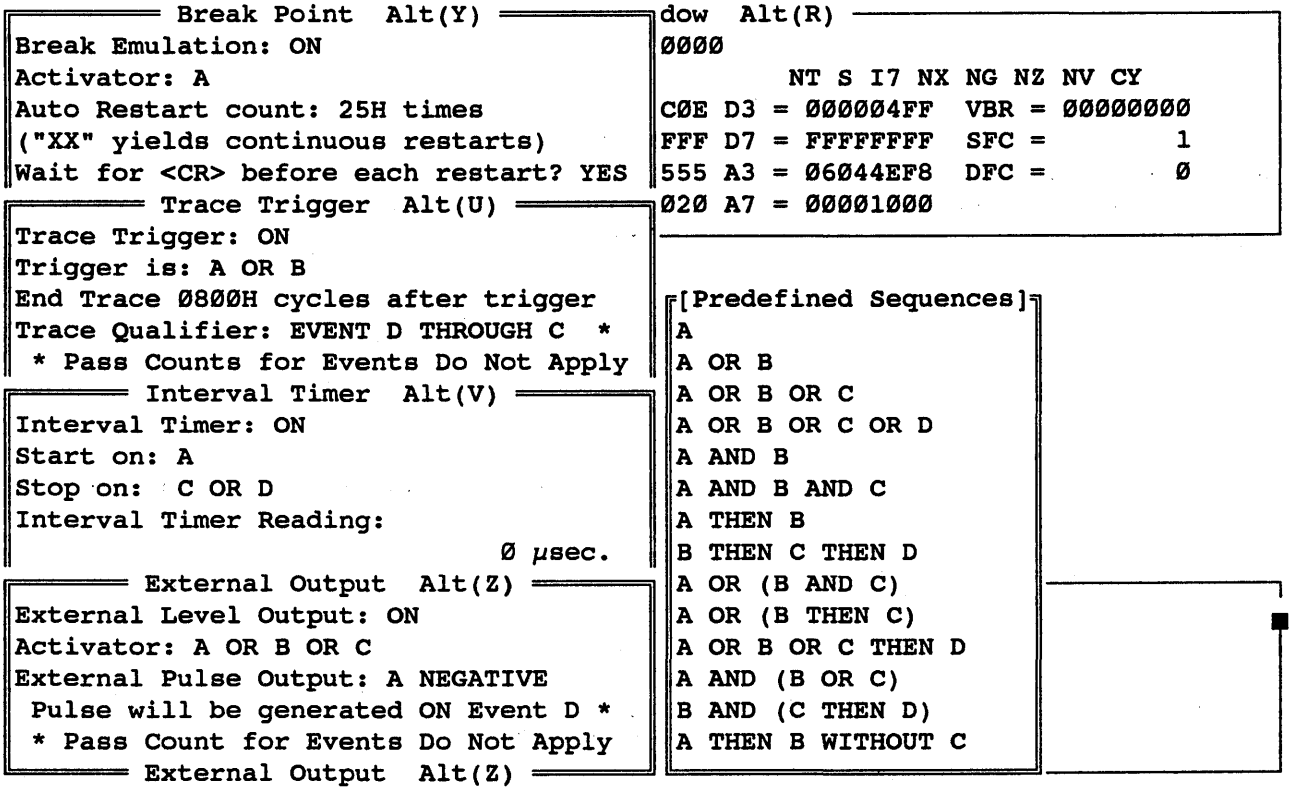
BERR':

The BERR' status bit allows you to define a condition to be when an invalid bus operation is being attempted, or, when used with the HALT signal, that the processor should retry the current cycle (BERR'= Ø). The inactive state of this signal is (BERR'= 1).

OF'/X:

The OF'/X status bit allows you to define a condition to be when an opcode is fetched (OF'/X = Ø), or when an opcode is executed (OF'/X = 1).

3.6 The Sequence Windows

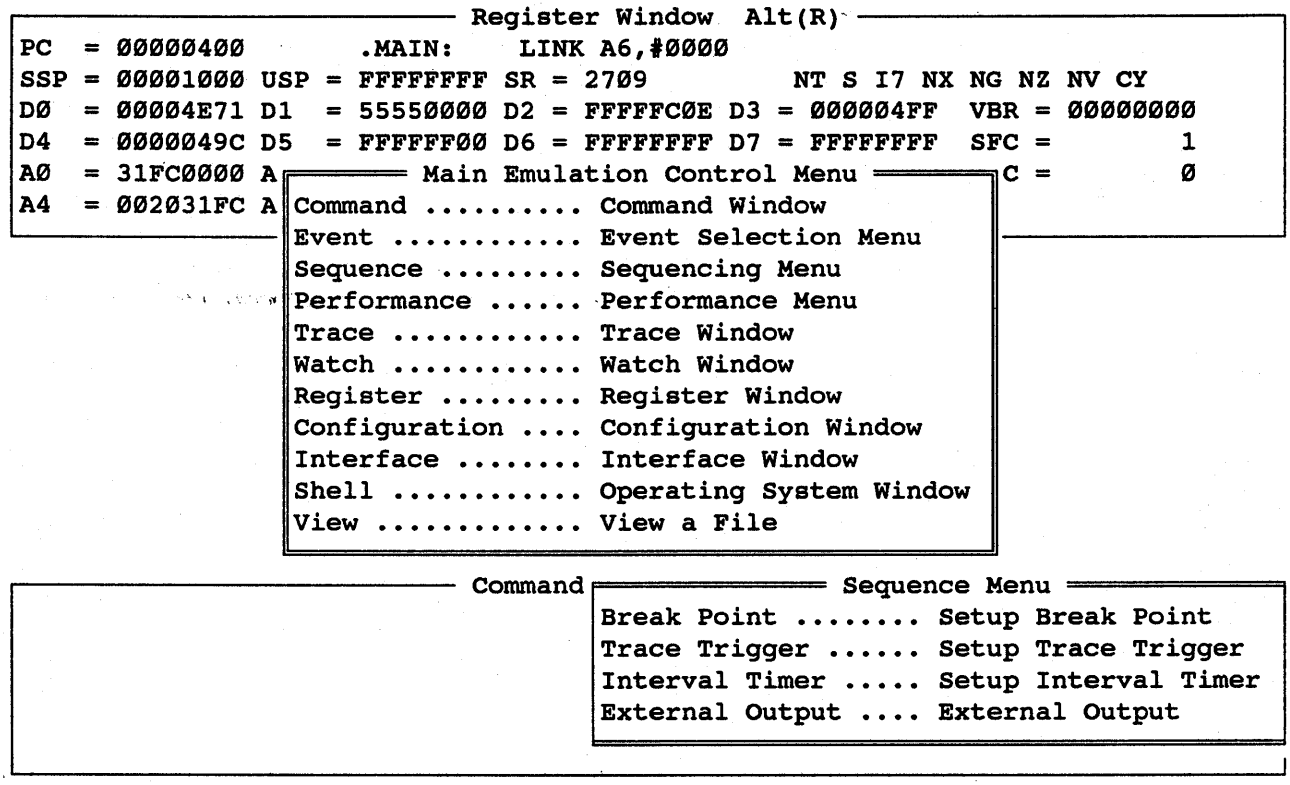


The Emulator Sequence Windows can be called up by pressing HOME and then TAB or <cr>. The Sequence windows are used to sequence the four events together to activate the selected function. Events can be sequenced together to cause emulation to terminate through Breakpoints, trigger the Trigger-trace, start and stop a Timer, and trigger signals out of the selected BNC connector. Each of these functions has a field to turn it on or off, and another field to enter the selected sequence.

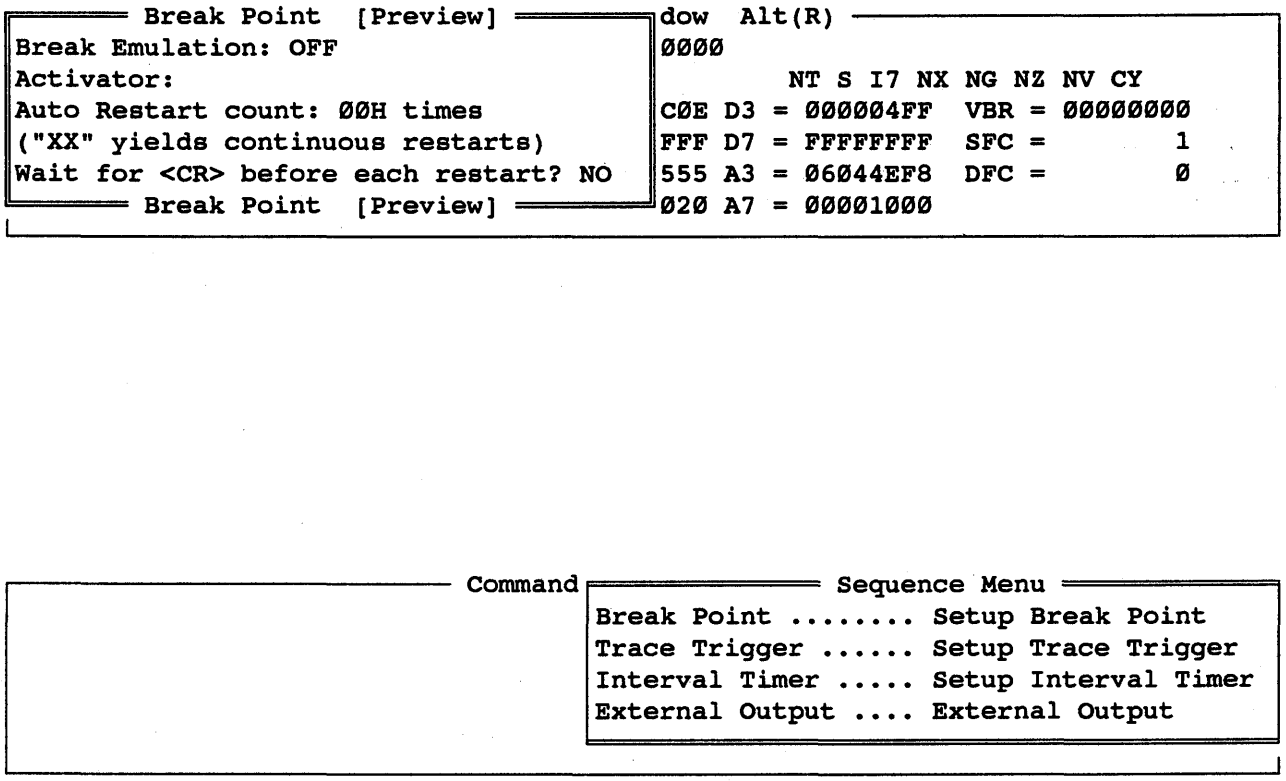
Typing the Home key at any point of the Sequence selection process will return control to the Main Emulation Control window.

3.6.1 Sequence Selection

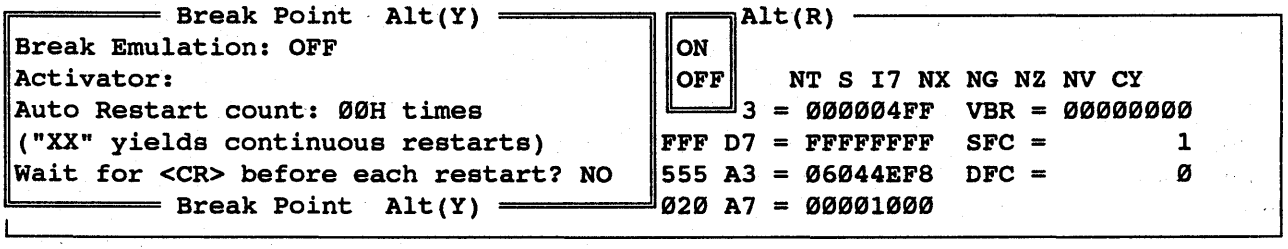
The emulator's four sequence selections have been broken down to smaller, compact windows with specific functions. Placing the cursor on the Sequence field in the window selection list will automatically show a Sequence Menu window for the Break Point, Trace Trigger, Interval Timer, and External Outputs:

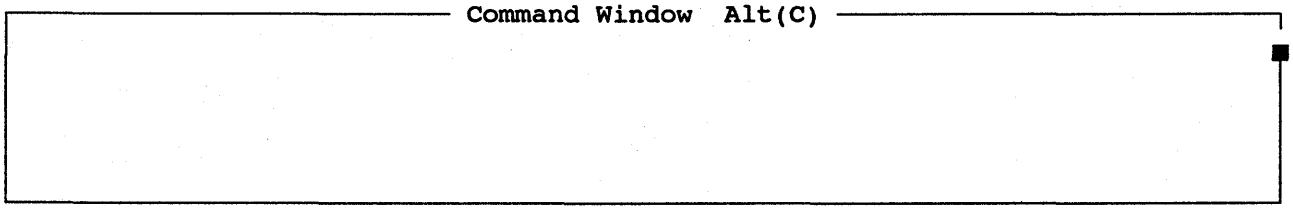


Typing a <cr> enters the Sequence Menu window with the corresponding Sequence Preview window appearing on the screen:

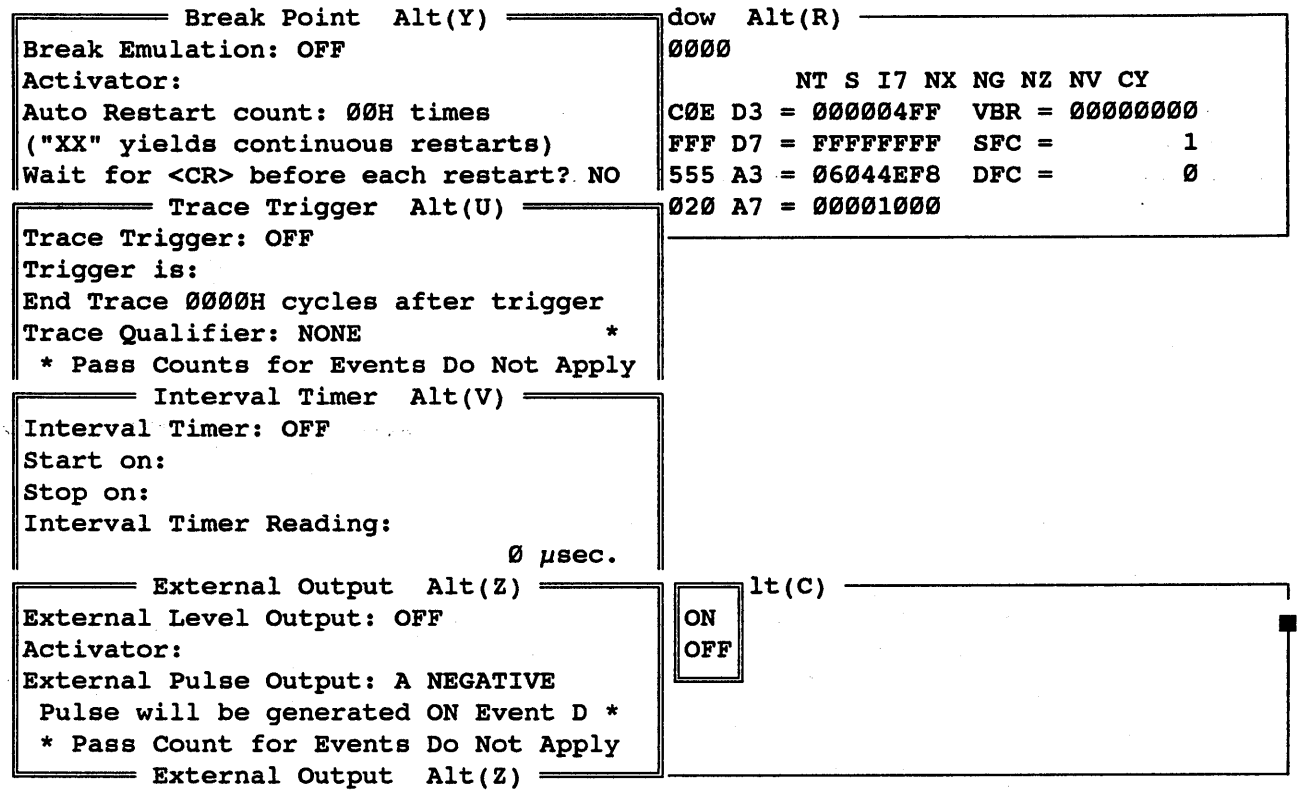


An additional <cr> will allow you to enter the selected Sequence Window and the Sequence Menu window will disappear:





All four Sequence windows can be pulled up by pressing the TAB key while the cursor is on the Sequence field in the "Main Emulation Control" Menu:



3.6.2 The Break Point Window

The Break Point window is used to establish a sequence of events that the emulator will break on. There are several fields that must be defined before breakpoints will function. Each of these fields is defined as follows:

3.6.2.1 Break Emulation ON/OFF Field

```
Break Point  Alt(Y)
Break Emulation: OFF
Activator:
Auto Restart count: ØØH times
("XX" yields continuous restarts)
Wait for <CR> before each restart? NO
Break Point  Alt(Y)
```

```
ON
OFF
```

Break-points can be enabled or disabled by selecting "ON" or "OFF" in the selection window. If break-points are to be turned "ON", then an event (or sequence of events) must be entered in the activator field.

3.6.2.2 Break Point Activator Field

```
Break Point  Alt(Y)
Break Emulation: OFF
Activator:
Auto Restart count: ØØH times
("XX" yields continuous restarts)
Wait for <CR> before each restart? NO
Break Point  Alt(Y)
```

```
[Predefined Sequences]
A
A OR B
A OR B OR C
A OR B OR C OR D
A AND B
A AND B AND C
A THEN B
B THEN C THEN D
A OR (B AND C)
A OR (B THEN C)
A OR B OR C THEN D
A AND (B OR C)
B AND (C THEN D)
A THEN B WITHOUT C
```

Enter the Event or Sequence of events which are to trigger the break-point. Choose from either the predefined list or enter your own sequence.

3.6.2.3 Auto Restart Count Field

```
Break Point  Alt(Y)
Break Emulation: OFF
Activator:
Auto Restart count: ØØH times
("XX" yields continuous restarts)
Wait for <CR> before each restart? NO
Break Point  Alt(Y)
```

An automatic restart function is available for breakpoints. By entering a two digit hexadecimal number in this field to select the desired number of restarts, the breakpoint operation will automatically break, display registers, and resume emulation (as defined in the next field). If an "XX" is entered in this field, then the restart operation will repeat until a SP (Stop Program Execution) is entered.

3.6.2.4 "Wait for <CR> Before Each Restart?" Field

```
Break Point  Alt(Y)
Break Emulation: OFF
Activator:
Auto Restart count: ØØH times
("XX" yields continuous restarts)
Wait for <CR> before each restart? NO
Break Point  Alt(Y)
```

```
YES
NO
```

If breakpoint operation is configured for auto-restart from the previous field, then you have the choice of how the emulator is to resume emulation after displaying the registers. By selecting "YES" in this field, then emulation will not resume until a carriage return is entered after the breakpoint is serviced. A "NO" in this field will allow the emulator to resume emulation immediately after the breakpoint is serviced, without waiting for user intervention.

3.6.3 The Trace Trigger Window

The Trace Trigger window is used to establish a sequence of events that the emulator will use to trigger the trigger-trace. There are several fields that must be defined before the trigger-trace feature will function. Each of these fields is defined as follows:

3.6.3.1 Trace Trigger ON/OFF Field

```
═══════ Trace Trigger  Alt(U) ═══════
Trace Trigger: OFF
Trigger is:
End Trace 0000H cycles after trigger
Trace Qualifier: NONE                *
 * Pass Counts for Events Do Not Apply
═══════ Trace Trigger  Alt(U) ═══════
```

```
ON
OFF
```

A trigger-point can be enabled or disabled by selecting "ON" or "OFF" in the selection window. If the trigger trace is turned "ON", then an event (or sequence of events) must be entered in the "Trigger is:" field.

3.6.3.2 "Trigger is:" Field

```
═══════ Trace Trigger  Alt(U) ═══════
Trace Trigger: OFF
Trigger is:
End Trace 0000H cycles after trigger
Trace Qualifier: NONE                *
 * Pass Counts for Events Do Not Apply
═══════ Trace Trigger  Alt(U) ═══════
```

```
[Predefined Sequences]
A
A OR B
A OR B OR C
A OR B OR C OR D
A AND B
A AND B AND C
A THEN B
B THEN C THEN D
A OR (B AND C)
A OR (B THEN C)
A OR B OR C THEN D
A AND (B OR C)
B AND (C THEN D)
A THEN B WITHOUT C
```

Enter the Event or Sequence of events which are to trigger the trigger-trace. Choose from either the predefined list or enter your own sequence.

3.6.3.3 "End Trace XXXXH Cycles After Trigger" Field

```
═══════ Trace Trigger  Alt(U) ═══════
Trace Trigger: OFF
Trigger is:
End Trace 0000H cycles after trigger
Trace Qualifier: NONE                *
 * Pass Counts for Events Do Not Apply
═══════ Trace Trigger  Alt(U) ═══════
```

This field is used to set the delay counter for the trigger trace. The delay counter is a hexadecimal number from 0000 to FFFF. After the trigger sequence occurs, the emulator will continue recording in the trigger trace buffer for the number of trace cycles specified in this field. After the counter has decremented to zero, the trigger trace buffer will be preserved while the break-trace buffer continues to record the on-going emulation.

3.6.3.4 Trace Qualifier Field

```
═══════ Trace Trigger  Alt(U) ═══════
Trace Trigger: OFF
Trigger is:
End Trace 0000H cycles after trigger
Trace Qualifier: NONE                *
 * Pass Counts for Events Do Not Apply
═══════ Trace Trigger  Alt(U) ═══════
```

```
NONE
DURING EVENT D
EVENT D THROUGH C
```

The trace qualifier is used to discriminate on the information being recorded in both the Trigger and Break trace buffers. Note that Pass Counts for these events do not apply. The options are as follows:

NONE: record all cycles.

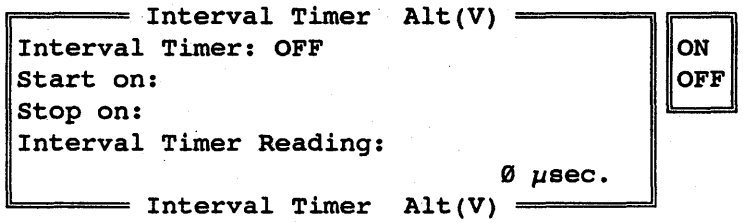
DURING EVENT D: configure event D to define the cycles to be recorded. Only those cycles which meet the conditions defined in event D will be recorded.

EVENT D THROUGH EVENT C: when the conditions of event D are met, tracing will begin. When the conditions of event C are met, tracing will stop. This process will repeat indefinitely.

3.6.4 The Interval Timer Window

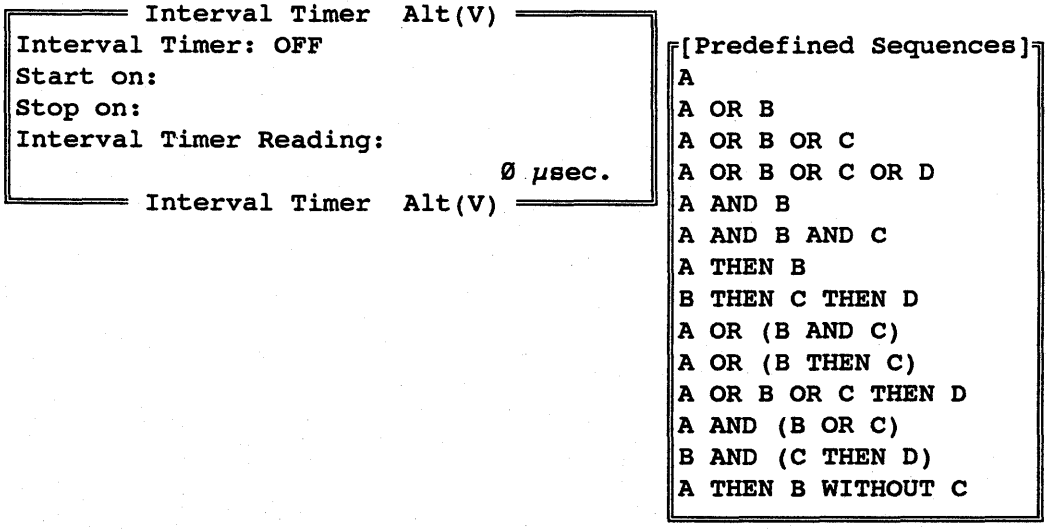
The Interval Timer window is used to establish a sequence of events that the timer will start and stop on. The timer has a resolution of 1μs and a maximum record time of 71 minutes. There are several fields that must be defined before the Interval Timer feature will function. Each of these fields is defined as follows:

3.6.4.1 Interval Timer ON/OFF Field



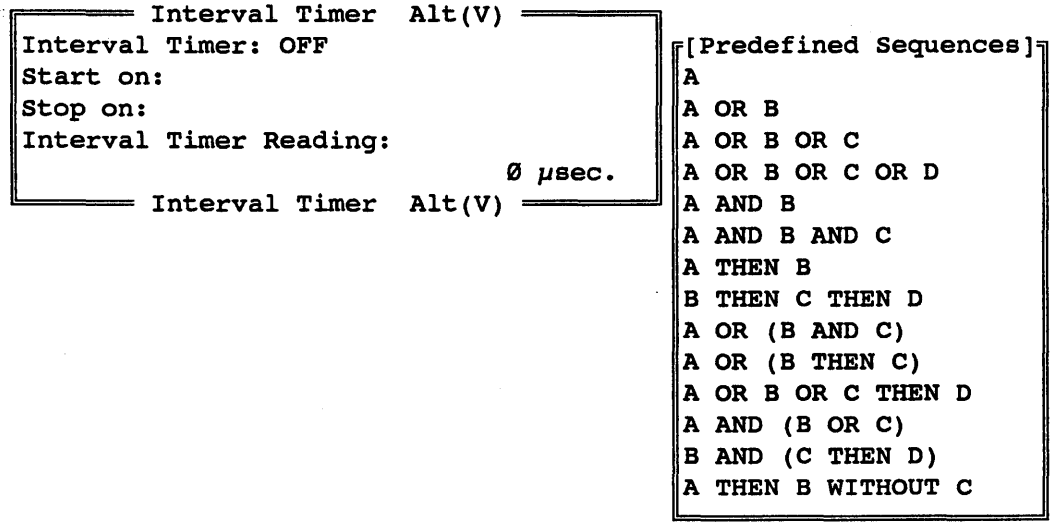
The timer can be enabled or disabled by selecting "ON" or "OFF" in the selection window. If the timer is turned "ON", then an event (or sequence of events) must be entered in both the "Start on:" and "Stop on:" fields.

3.6.4.2 "Start On:" Field



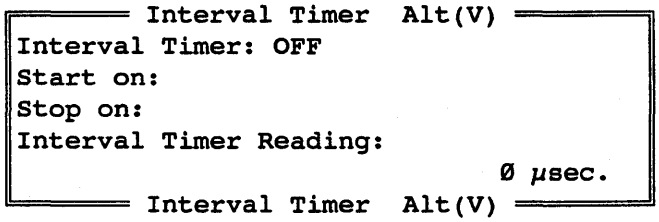
Enter the Event or Sequence of events which are to trigger the timer to start running. Choose from either the predefined list or enter your own sequence.

3.6.4.3 "Stop On:" Field



Enter the Event or Sequence of events which are to trigger the timer to stop running. Choose from either the predefined list or enter your own sequence.

3.6.4.4 "Interval Timer Reading:" Field

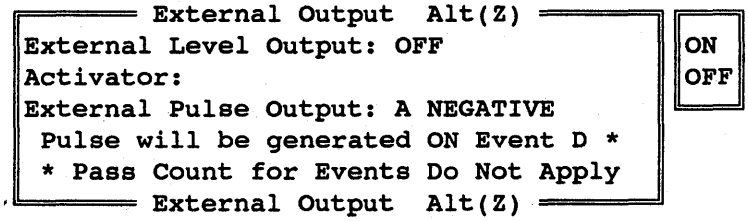


Once the timer has run, it will display the elapsed time in this field. It can be restarted by stopping emulation and then resuming or by entering a "Freeze Trace" and then a "Resume Trace". One thing to keep in mind is that the timer will not function after the Stop Sequence has occurred. Therefore, it is important to make sure that the Stop Sequence does not occur before the Start Sequence.

3.6.5 The External Output Window

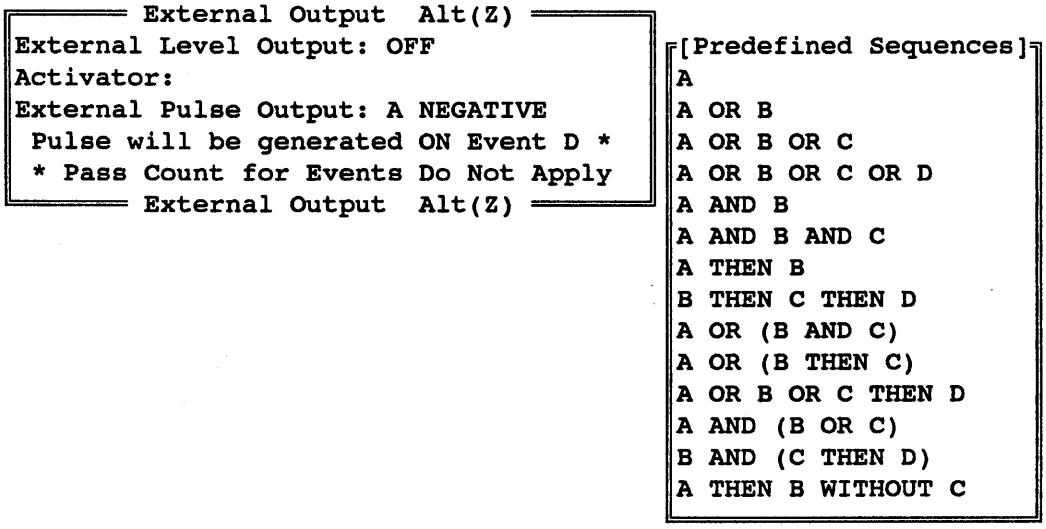
The External Output window is used to establish a sequence of events that will trigger signals out of the selected BNC connector. There are several fields that must be defined before the External Output feature will function. Each of these fields is defined as follows:

3.6.5.1 External Level Output ON/OFF Field



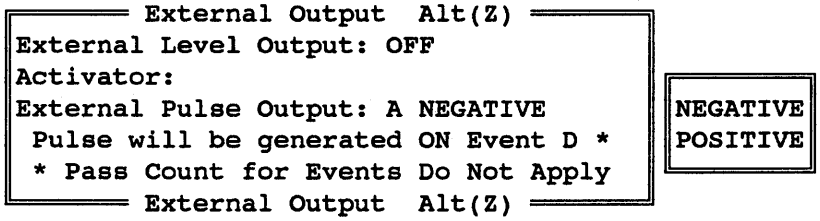
The level output can be enabled or disabled by selecting "ON" or "OFF" in the selection window. If the level-out is turned "ON", then an event (or sequence of events) must be entered in the sequence field.

3.6.5.2 External Level Output Activator Field



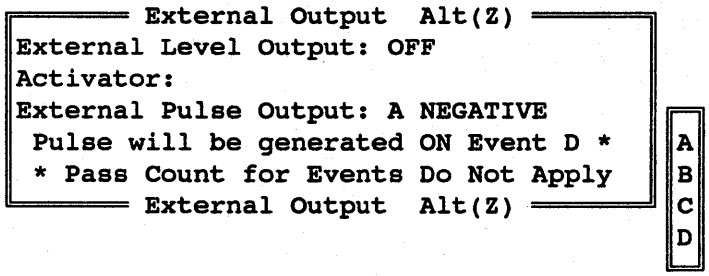
Enter the Event or Sequence of events which are to trigger the level output signal on the BNC connector on the front of the emulator. When the conditions of the sequence are met, the signal on the level output will toggle from a logic low to a logic high level. Choose from either the predefined list or enter your own sequence.

3.6.5.3 External Pulse Output Polarity Field



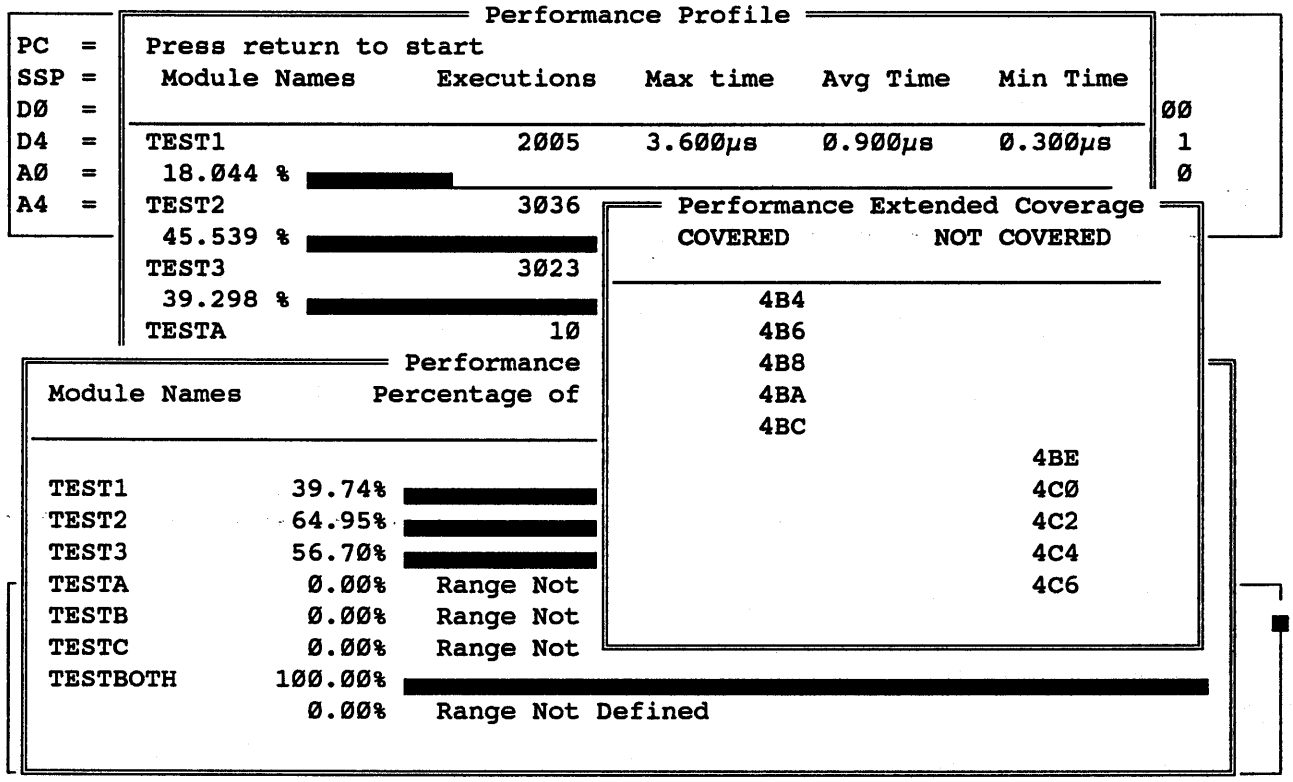
The pulse output signal on the BNC connector on the front of the emulator can be programmed to generate a "negative" or "positive" pulse on every occurrence of the selected event.

3.6.5.4 "Pulse Will Be Generated on Event X" Field



When the conditions of the selected event (A,B,C,D) occur (excluding pass counts), the emulator will generate a five volt pulse through the "Pulse Out" BNC connector located on the front of the emulator. The polarity of this pulse is determined from the previous field.

3.7 The Performance Windows

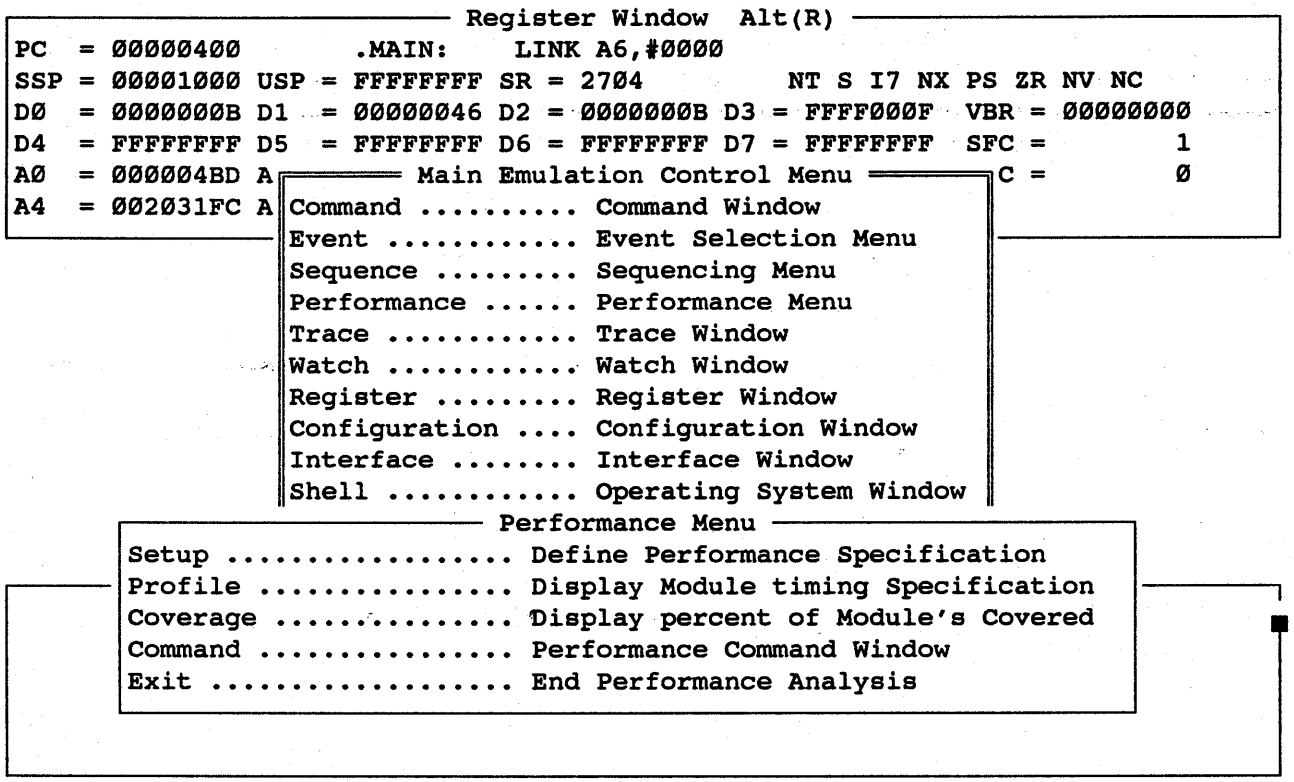


The Performance Analysis function is designed to offer "Real Time Analysis" of hardware and software systems. The result of an analysis session will be improved code optimization and a better understanding of timing and code performance.

The HMI Performance Analyzer performs its analysis in hardware using an additional optional circuit board that is installed inside the emulator. This approach is different than the "statistical" software analyzers used by most emulator manufacturers. This "statistical" approach cannot match the accuracy and speed of true "hardware" performance analyzers.

3.7.1 Performance Selection

The Performance Windows can be entered from the Main Emulation Control window. When the cursor is moved to "Performance" in the Main Emulation Control window, a Performance Selection Menu will be displayed:



Entering a <cr> on Performance will allow access to the Performance Menu:

```
                              Register Window  Alt(R)
PC  = 00000400        .MAIN:    LINK A6,#0000
SSP = 00001000 USP = FFFFFFFF SR = 2704          NT S I7 NX PS ZR NV NC
D0  = 0000000B D1  = 00000046 D2 = 0000000B D3 = FFFF000F  VBR = 00000000
D4  = FFFFFFFF D5  = FFFFFFFF D6 = FFFFFFFF D7 = FFFFFFFF  SFC =        1
A0  = 000004BD A1  = 000004AD A2 = 000004AE A3 = 06044EF8  DFC =        0
A4  = 002031FC A5  = AAAA0604 A6 = 00000FFC A7 = 00001000
```

```
                          Performance Menu
Setup ................... Define Performance Specification
Profile ................. Display Module timing Specification
Coverage ................ Display percent of Module's Covered
Command ................. Performance Command Window
Exit .................... End Performance Analysis
```

3.7.2 The Performance Analysis Setup Window

```
Performance Analysis Setup  Alt(P)
Module Name     Address  Range    Entry     Exit

TEST1           000480 - 00051A   000000    000000
TEST2           000440 - 000500   000000    000000
TEST3           000450 - 000510   000000    000000
TESTA           000000 - 000000   000408    000450
TESTB           000000 - 000000   00041E    000460
TESTC           000000 - 000000   000450    00048A
TESTBOTH        000450 - 00048A   000450    000468
                000000 - 000000   000000    000000

Recorder Mode     = Rerun set number of times
Number of Reruns  = 005
Record Time       = 00002 x 1ms
Coverage mode     = Establish new coverage
Trigger           = Automatic
Cumulative        = YES
Symbol Name       =
```

The Setup window is used to define the parameters used in each Performance session. There are several fields that must be defined before the Performance Analyzer will function. Each of these fields is defined as follows:

3.7.2.1 Module Name Field

The Module Name field is used to define a module name for the address-range or entry and exit points on which Performance analysis will be executed. The histogram and coverage analysis will be performed on each of the defined modules. Up to 8 modules can be defined.

3.7.2.2 Address-Range Fields

The address-range defines a range whereby if the processor accesses an address in the defined range it will be recorded for processing in the profile and coverage windows.

The first address field in the range is always smaller than the second address field. The address-range can be any length for timing analysis in the profile window but must be less than 64K for code coverage analysis.

Any hexadecimal digits (Ø-F) are allowable characters for the hexadecimal address-range fields.

3.7.2.3 Entry-Exit Fields

The entry-exit defines the beginning address to time-stamp and the ending address to time-stamp. The code must execute the beginning address and ending address to count as an execution. The difference between the entry and exit point can be any length for timing analysis in the profile window.

Any hexadecimal digits (Ø-F) are allowable characters for the entry-exit field.

3.7.2.4 Combined Address-Range and Entry-Exit Fields

Address-Ranges and Entry-Exit points can also be combined. The entry-exit defines an address in the range where the code must enter (only if it enters the range at the defined entry point) and exit (only if it exits the range at the defined exit point) in order for executions to be counted. The timer continues for addresses in the defined range but the time is not counted for addresses outside the range.

3.7.2.5 Recorder Mode Field

The recorder mode determines how Performance analysis will be run. The available options are the following:

Rerun Set Number of Times: The rerun option runs, then updates the screen, then runs again. This continues as per the number of reruns defined.

Free run (continuous): The free run option runs, then updates the screen, then runs again continuously until told to stop.

3.7.2.6 Number of Reruns Field

If the Rerun Set Number of Times option is chosen, then this field will tell the emulator the number of times to run the Performance analysis and then stop.

3.7.2.7 Record Time Field

The record time determines how long the analysis will be run for each pass. It is defined in milliseconds.

3.7.2.8 Coverage Mode Field

The coverage mode determines how Performance analysis will be displayed. Select from one of the following options:

Establish New Coverage: All exiting analysis will be written over by the new coverage that is being run.

Append Current Coverage: The exiting analysis is saved and the new coverage is added to the coverage that was previously run.

3.7.2.9 Trigger Field

The Trigger field can be set to be automatic or manual mode. When automatic is selected, the Performance Analysis will run from where you currently are in the code.

When manual is selected, the Performance Analysis will run from the address of the module that is defined in the setup.

3.7.2.10 Cumulative Field

The Cumulative field allows you to either run each analysis independent of the next without accumulation or to accumulate the values to reflect the total of all the runs.

3.7.2.11 Symbol Name Field

The Symbol Name field allows you to enter symbol names instead of hex values for the Address-Range and Entry-Exit fields. While in these fields, entering a ':' will take you to the Symbol Name Field.

3.7.3 The Performance Analysis Profile Window

Performance Profile

Press return to start

Module Names	Executions	Max time	Avg Time	Min Time
TEST1	2005	3.600µs	0.900µs	0.300µs
18.044 %				
TEST2	3036	7.200µs	1.500µs	0.300µs
45.539 %				
TEST3	3023	5.200µs	1.300µs	0.300µs
39.298 %				
TESTA	10	426.600µs	426.500µs	426.500µs
42.649 %				
TESTB	11	120.200µs	120.200µs	120.200µs
13.222 %				
TESTC	197	4.400µs	4.400µs	4.400µs
8.668 %				
TESTBOTH	183	6.800µs	6.800µs	6.800µs
12.444 %				
	0	0.000µs	0.000µs	0.000µs
0.000 %				

Reruns Requested = 005 Record Time = 2.000ms
Reruns Executed = 5 Total Record Time = 10.000ms
Timer Resolution = 001 x 100 ns

The Performance Profile displays the module name, percent of time that was spent in each module, number of executions for each module, along with max, min, and avg time spent in each module. There are several fields that are displayed when the Performance Profile window is displayed. Each of these fields is defined as follows:

3.7.3.1 Reruns Requested and Record Time Fields

These values were defined in the Performance Analysis Setup Section.

3.7.3.2 Reruns Executed Field

This is the actual number of reruns that have been executed.

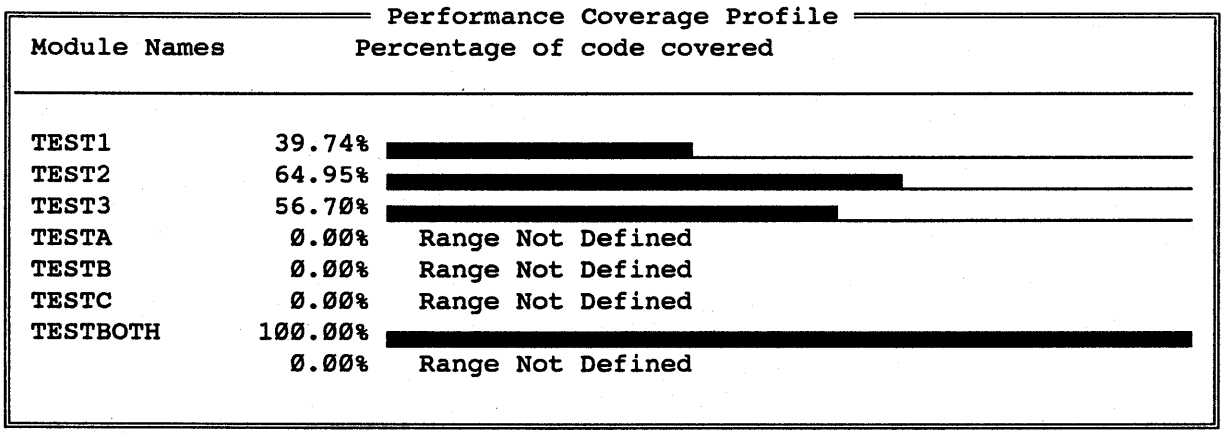
3.7.3.3 Total Record Time Field

The total record time is the total time it took for all of the reruns to be executed.

3.7.3.4 Timer Resolution Field

The timer resolution is the defined resolution for the time tagging in the trace buffer. The timer resolution is defined in nanoseconds.

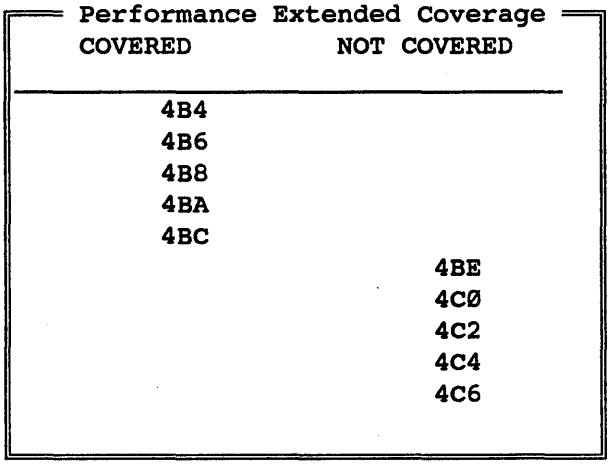
3.7.4 The Performance Analysis Coverage Window



The Performance Coverage window displays the module names and histograms, showing percentage of code that was covered in each module that has an Address-Range defined. Modules with no Address-Range defined will display the message "Range Not Defined".

Note: Coverage Analysis is performed only for those modules that have an Address-Range defined.

3.7.5 The Performance Analysis Extended Coverage Window

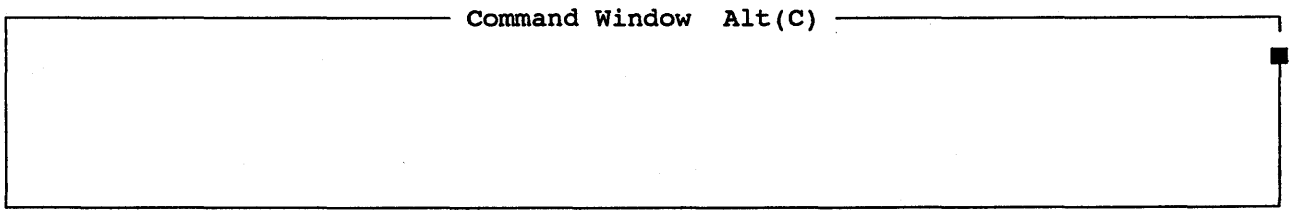


Extended coverage can be displayed on each of the modules by pressing <cr> on the desired module, while in the Coverage window.

The Extended Coverage displays the covered and not-covered addresses for the module selected in the Performance Analysis Coverage window.

Available Commands
N - To find not-covered addresses
C - To find covered addresses
FT - Top of screen
FZ - Bottom of screen
F <addr> - Find a specified address

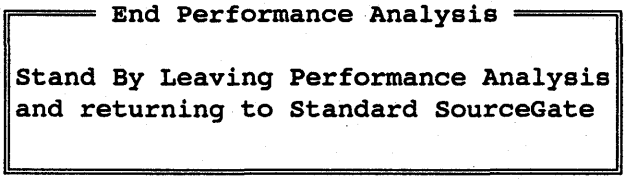
3.7.6 The Performance Analysis Command Window



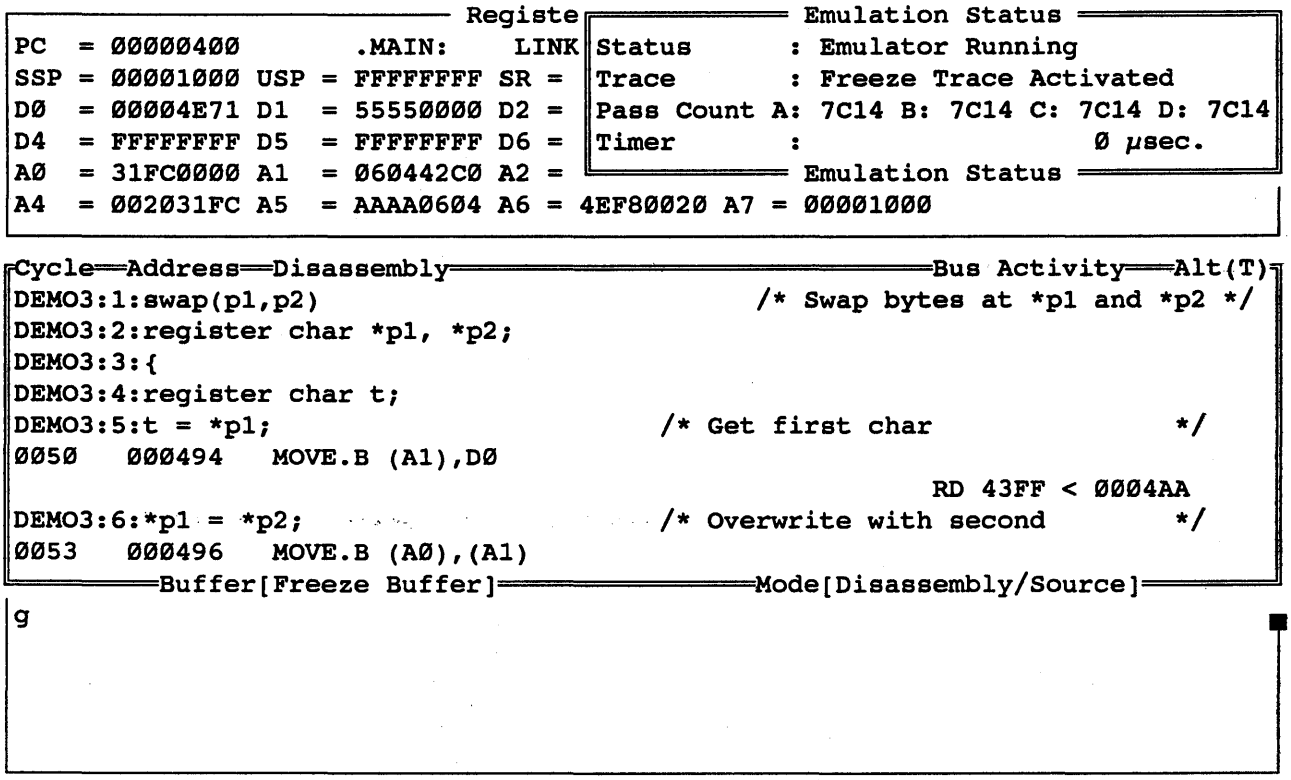
Selecting "Command" while in the Performance Selection Menu will take you back to the emulator's Command window. Commands such as go, stop, reset, etc. can then be executed.

3.7.7 Exiting The Performance Analyzer

Selecting "Exit" while in the Performance Selection Menu will end the Performance Analysis session. The following message will be flashed on the screen:



3.8 The Trace Windows



The Trace menu allows the user to view and search through the two 4k by 72 bit trace buffers (Trigger Trace and Break Trace) and to initiate and view the Freeze Trace. Note that the trigger trace will exist if the user has setup a trigger point condition and during emulation has met that condition. Scrolling through the trace can be accomplished by using the page and arrow keys on the keyboard. The format of the trace display can be Disassembly, State, Source, Disassembly/State, Disassembly/Source, or Disassembly/State/Source. Other functions of the Trace window include a Trace Search and Trace Print.

The Trace window is updated each time emulation is terminated through a break point, stop command or reset command and also on a freeze trace command.

3.8.1 Trace Window Commands

The following is a list of commands that are available in the trace window:

<CR>	Select Display Format
(UpArrow)	Line Up
(DnArrow)	Line Down
(PgUp)	Page Up
(PgDn)	Page Down
B	Beginning of buffer
E	End of buffer
F	Freeze trace
R	Resume trace
CNTRL-S	Setup parameters for search
S	Search trace buffer
N	Search for Next occurrence
CNTRL-P	Setup parameters for print
P	Print trace
T	Go to Trigger point

<CR>:

By pressing <CR> while in the Trace window, the Trace Configuration Window will be displayed. The Trace Configuration window is used to define the buffer, format, time-tag, etc.

UpArrow:

Enter the 'Up Arrow' key to scroll the trace back one line. Going backwards will display a cycle which occurred earlier than the cycle currently at the top of the window.

DnArrow:

Enter the 'Down Arrow' key to scroll the trace forward one line. Going forward will display a cycle which occurred later than the cycle currently at the bottom of the window.

PgUp:

Enter the 'Page Up' key to scroll the trace backwards one page. Going backwards will display a page of trace which occurred earlier than the cycles currently displayed in the window.

PgDn:

Enter the 'Page Down' key to scroll the trace forward one page. Going forward will display a page of trace which occurred later than the cycles currently displayed in the window.

B:

Entering the 'B' command will move the trace pointer to the Beginning of the current trace buffer. The beginning of the trace is the earliest information in the trace buffer.

E:

Entering the 'E' command will move the trace pointer to the End of the current trace buffer. The trace data at the end of the trace is the latest information in the trace buffer.

T:

If the information in the current trace window is that of the trigger trace buffer, and the trigger point lies between the beginning and ending of the recorded information, then the 'T' command will search for and display the Trigger point.

Note:

The "F" and "R" commands are covered in section 3.8.4; "CNTRL-S", "S", and "N" commands are covered in section 3.8.5; "CNTRL-P" and "P" commands are covered in section 3.8.6.

3.8.2 Trace Window Fields

The Trace window has several fields that are listed in the top border of the window. The fields that are shown depend on the format chosen for the Trace window display.

If the Trace window is in one of the following formats:

Disassembly
Source
Disassembly/Source

then the trace will be displayed showing the Cycle number, Bank, Address, Disassembly (and/or Source code), and any Bus Activity:

```
Cycle==Address==Disassembly=================================Bus Activity===Alt(T)
DEMO3:1:swap(p1,p2)                              /* Swap bytes at *p1 and *p2 */
DEMO3:2:register char *p1, *p2;
DEMO3:3:{
DEMO3:4:register char t;
DEMO3:5:t = *p1;                          /* Get first char                */
0050   000494   MOVE.B (A1),D0
                                                          RD 43FF < 0004AA
DEMO3:6:*p1 = *p2;                        /* Overwrite with second         */
0053   000496   MOVE.B (A0),(A1)
                                                          RD 00FF < 0004BA
                                                          WR 0000 > 0004AA
DEMO3:7:*p2 = t;                          /* Overwrite second with first   */
0055   000498   MOVE.B D0,(A0)
                                                          WR 4343 > 0004BA
DEMO3:8:}
0058   00049A   UNLK A6
                                                          RD 0000 < 000FE0
                                                          RD 0FFC < 000FE2
005A   00049C   RTS
                                                          RD 0000 < 000FE4
                                                          RD 045E < 000FE6
0060   00045E   ADDQ.L #8,A7
0061   000460   ADDQ.L #1,D2
         ==Buffer[Break buffer]=================Mode[Disassembly/Source]=======
```

If the Trace window is in one of the following formats:

State
Disassembly/State
Disassembly/State/Source

then, in addition to the previously mentioned fields, the Data, External Trace bits, and Status bits will be displayed:

```
Cycle==Address=Data==External=====Status: BA PM IPL FC  R==Bus Activity=Alt(T)
DEMO3:1:swap(p1,p2)                          /* Swap bytes at *p1 and *p2 */
DEMO3:2:register char *p1, *p2;
DEMO3:3:{
DEMO3:4:register char t;
DEMO3:5:t = *p1;                           /* Get first char              */
0050   000494  1011  FFFF                    11 11 111 110 1
   MOVE.B (A1),D0
                                                          RD 43FF < 0004AA
0051   000FEC  0000  FFFF                    11 11 111 101 1
0052   000FEE  04BA  FFFF                    11 11 111 101 1
DEMO3:6:*p1 = *p2;                         /* Overwrite with second       */
0053   000496  1290  FFFF                    11 11 111 110 1
   MOVE.B (A0),(A1)
                                                          RD 00FF < 0004BA
                                                          WR 0000 > 0004AA
0054   0004AA  43FF  FFFF                    11 11 111 101 1
DEMO3:7:*p2 = t;                           /* Overwrite second with first */
0055   000498  1080  FFFF                    11 11 111 110 1
   MOVE.B D0,(A0)
                                                          WR 4343 > 0004BA
0056   0004BA  00FF  FFFF                    11 11 111 101 1
0057   0004AA  0000  FFFF                    11 11 111 101 0
DEMO3:8:}
========Buffer[Break buffer]==============Mode[Disassembly/State/Source]=====
```

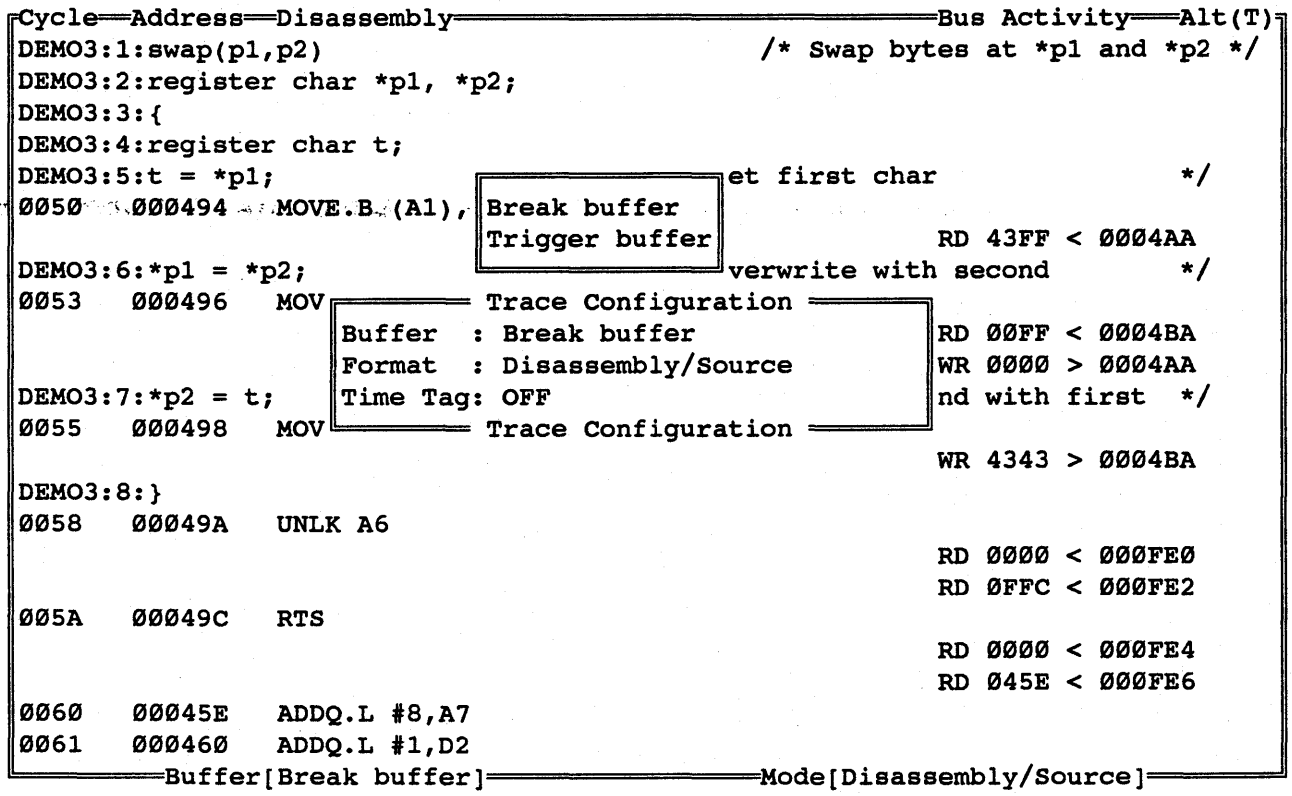
The External Trace bits are the 16 lines that plug into the front of the emulator (as described in section 3.5.2.6) while the Status bits are described in section 3.5.2.8 and as follows:

BA - BERR',AERR'
PM - VPA',VMA'
IPL - IPL2',IPL1',IPL0'
FC - FC2,FC1,FC0
R - R/W'

3.8.3 The Trace Configuration Window

The Trace Configuration window can be viewed by entering a carriage return while in the Trace Window. To exit the Trace Configuration window, enter the 'ESC' key. The Trace Configuration window will disappear and the new configuration of the Trace window will become active. There are several fields that can be defined in the Trace Configuration window. These fields are defined as follows:

3.8.3.1 Trace Buffer Selection Field



By entering a <cr> while in the Buffer field, the trace can be set to view the Break or Trigger trace buffers. Notice that the buffer being viewed (either Trigger or Break) is shown in the bottom border of the Trace window. Buffer definitions follow:

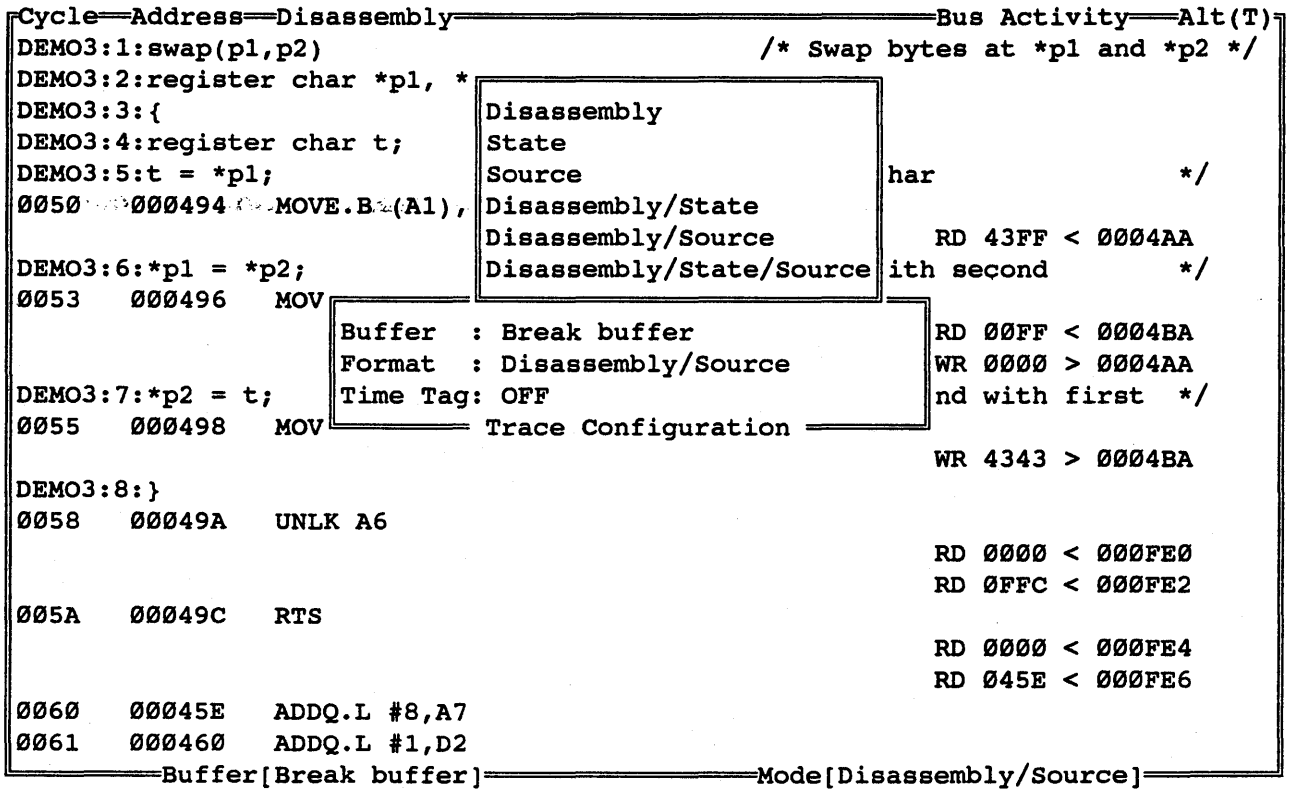
Break Buffer:

The Break buffer is the traced information (up to 4K deep) which was recorded until either a Stop command, Reset command, Freeze command, or Break Point occurred.

Trigger Buffer:

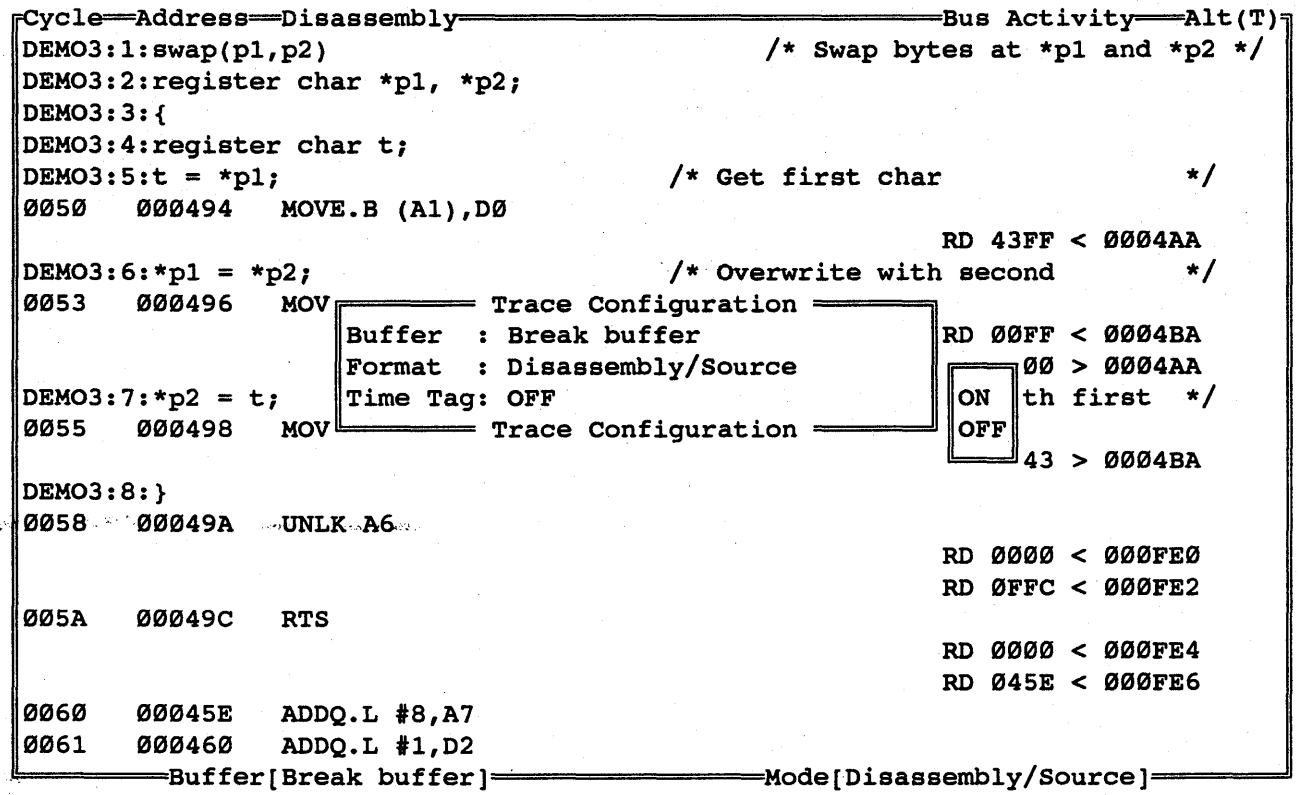
The Trigger buffer is the traced information which was recorded before the Event definition of the trigger point occurred and the information recorded for the number of cycles specified in the delay counter after the trigger point. The parameters defining the trigger trace are found in the Trigger Trace section of the Sequence windows.

3.8.3.2 Trace Display Format Field



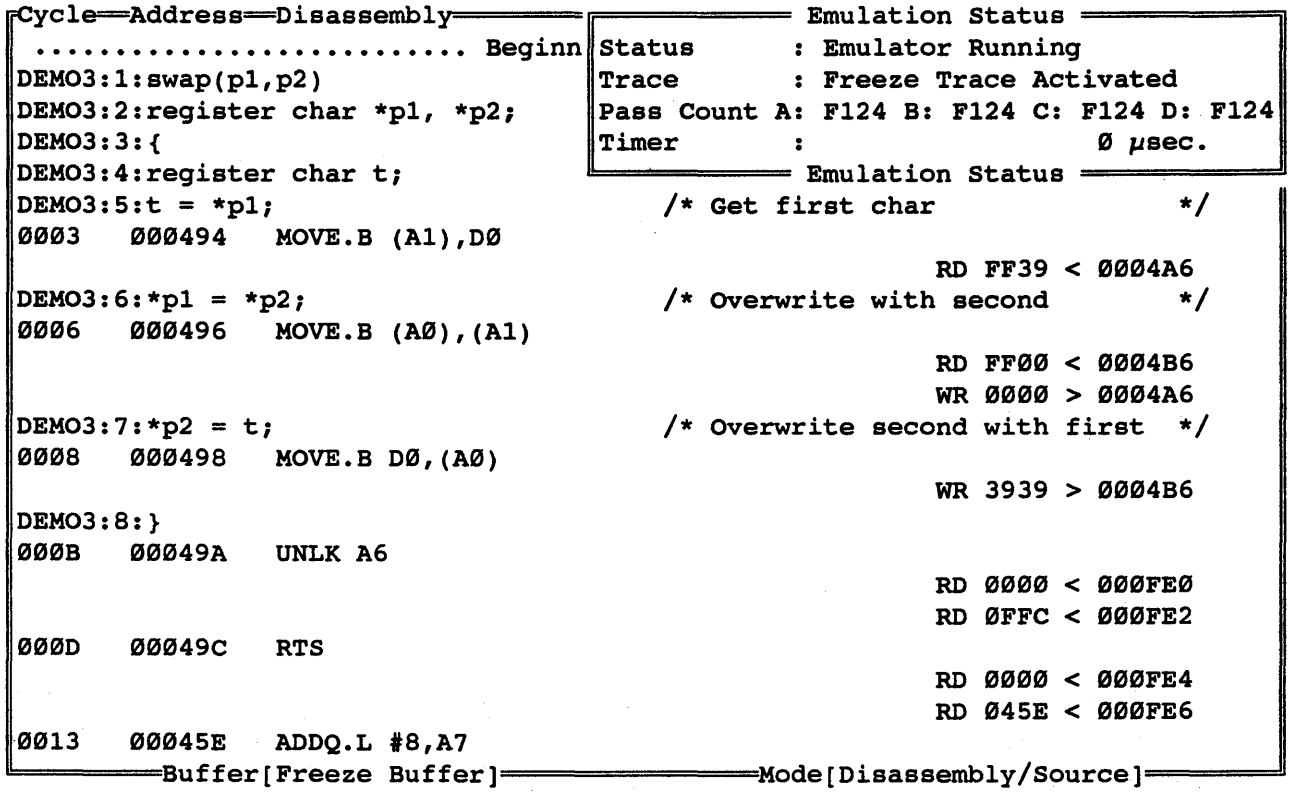
By entering a <cr> while in the Format field, the desired trace format can be entered. Notice that the current Format is shown in the bottom border of the Trace window.

3.8.3.3 Time Tag ON/OFF Field



The Time Tag field can be selected to be "ON" or "OFF" by a <cr> while in the Time Tag field and then arrowing to the desired selection. If ON is selected, each cycle in the trace buffer is displayed with a time tag from the Performance Analysis card (if the Performance Option is installed).

3.8.4 The Freeze Trace Window



The Freeze function may be executed by entering the 'F' command while in the trace window. This function allows the user to view the trace buffer and to modify the Event and Sequence definitions while the processor continues to run. While the "freeze" is active, functions such as tracing, breaking, and triggering are disabled. To reactivate these functions, enter the Resume command 'R'.

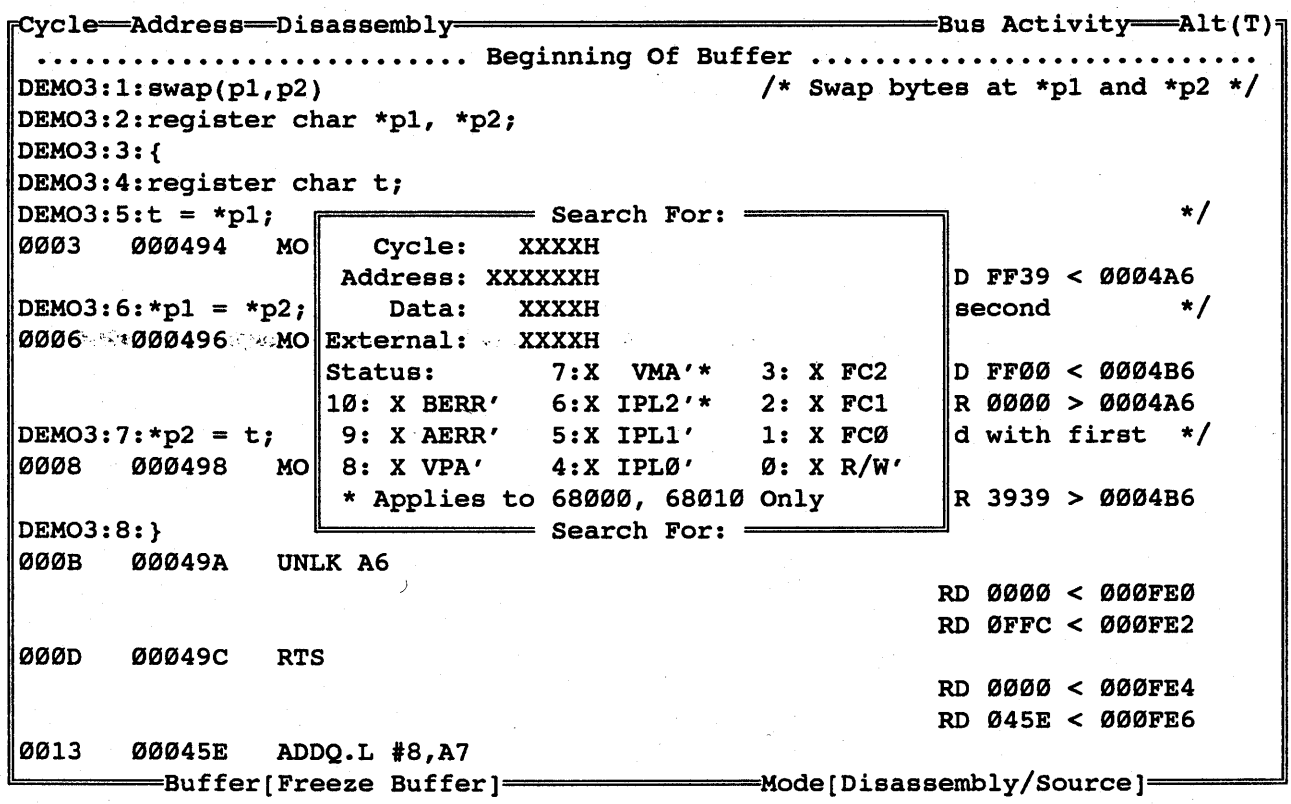
After the "freeze" command is entered, the contents of the trace buffer will be determined by the following:

a. If the trigger condition plus delay has been met, then the freeze trace will display the trigger trace buffer.

b. If the trigger condition plus delay has not been met, then the freeze trace will display a "snapshot" of the last 4K cycles of code executed before the freeze trace command was given.

Neither the "freeze" nor the "resume" disturb the real time emulation of the processor. When the "resume" is executed, the current Event and Sequence parameters are reprogrammed, thus providing the user a mechanism to dynamically modify these parameters.

3.8.5 The Trace "Search For:" Window



An automatic searching function can be initiated by entering the "CNTRL-S" command. After entering this command, a Trace Search Pattern window will appear. To search for a cycle number, address, data, etc., simply fill in the desired pattern. Press ESC to delete the Trace "Search For:" window, then type in "S" to begin the search. To search for next occurrence of a previously defined pattern, type in "N".

There are several fields that can be defined before a search of the trace buffer is initiated. Each of these fields is defined as follows:

3.8.5.1 Search For: Cycle Field

The cycle number field corresponds with the cycle pattern that is to be searched for in the trace buffer.

3.8.5.2 Search For: Address Field

The address field corresponds to the address pattern that is to be searched for in the trace buffer. Any hexadecimal digit (Ø-F) and (X) are allowable characters for the hexadecimal address field, where (X) represents the don't care.

To enter a Symbol instead of the absolute value, type a colon ':' to enter the symbol entry field.

3.8.5.3 Search For: Data Field

The data field corresponds to the data pattern that is to be searched for in the trace buffer. Any hexadecimal digit (Ø-F) and (X) are allowable characters for the hexadecimal data field, where (X) represents the don't care.

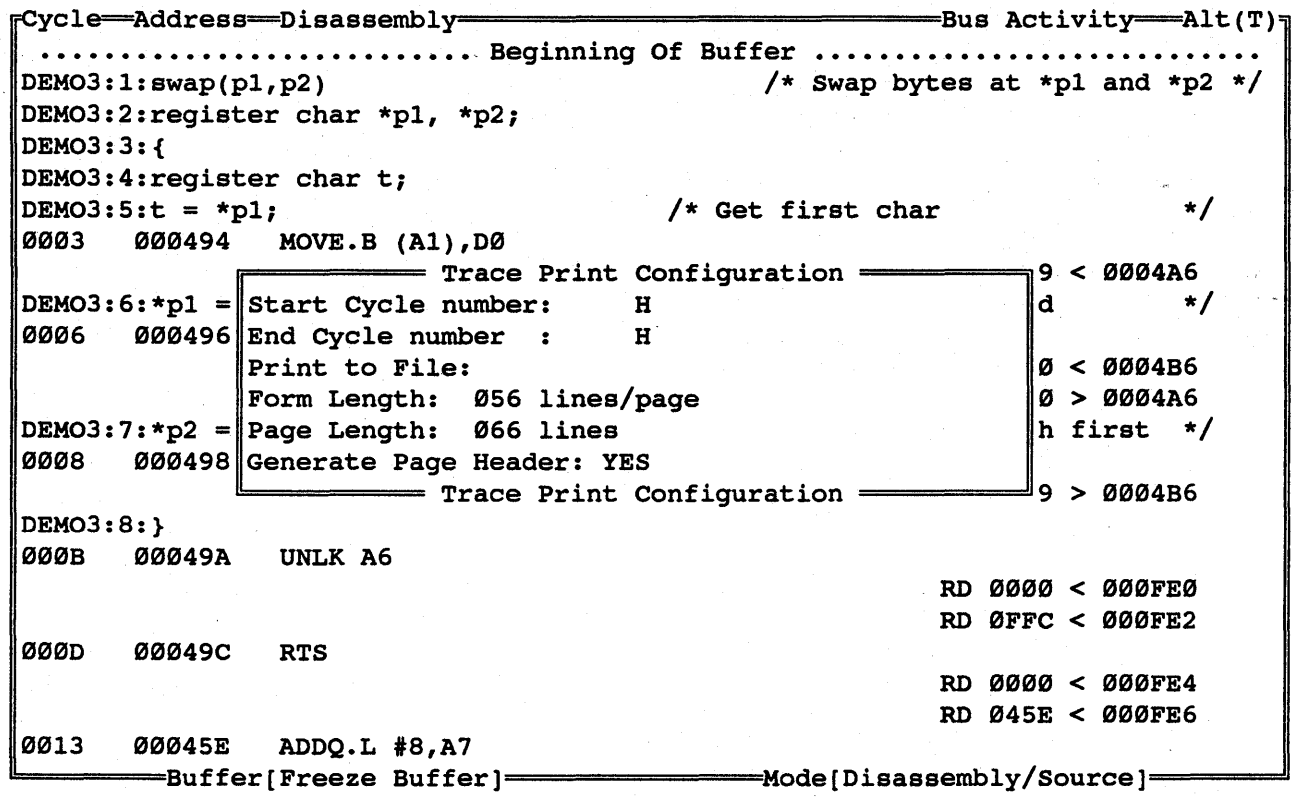
3.8.5.4 Search For: External Field

The external field corresponds to the external-bit pattern generated by the external trace lines that is to be searched for in the trace buffer. Any hexadecimal digit (Ø-F) and (X) are allowable characters for the hexadecimal external-bit field, where (X) represents the don't care.

3.8.5.5 Search For: Status Fields

The individual status conditions to be searched for are the same conditions that were defined in section 3.5.2.8.

3.8.6 The Trace Print Configuration Window



Using the "CNTRL-P" command will display the "Trace Print Configuration" Window which will prompt the user for the necessary information required to open a file and record the desired trace data in a formatted ASCII file, or to direct the trace output to a printer. The format of the file produced will be determined by the current mode of the display. If the trace is being displayed in a Source Only mode, then that is the format of the data which will be written to the specified file. A hard copy of the trace data may be obtained directly by entering PRN (or other valid device name such as LPT1, a preceeding colon is not necessary) for the file name.

While printing, this window will display the cycle number of the line which is currently being written to the output file. If you wish to terminate the printing function before the ending cycle number is reached, type ESC and the file will be closed automatically, ending with the last cycle which was written before the ESC was executed.

Once the parameters of the "Trace Print Configuration" Window are defined, press ESC to delete the window. A "P" begins the print transfer. To abort the transfer, press ESC.

There are several fields that should be defined prior to printing the Trace display. Each of these fields is defined as follows:

3.8.6.1 Trace Print "Start Cycle Number" Field

The start cycle number corresponds with the cycle number of the first line in the trace buffer which is to be written to the specified file.

3.8.6.2 Trace Print "End Cycle Number" Field

The End cycle number corresponds with the cycle number of the last line in the trace buffer which is to be written to the specified file.

3.8.6.3 Trace Print "Print to File" Field

This filename field can consist of any path directives or a device name (such as LPT1 or PRN for a printer). Any file which already exists with the same filename will automatically be overwritten.

3.8.6.4 Trace Print "Form Length" Field

The Form Length is used to define the number of lines of trace data which are to be written to each page. The difference between the "Pagelength" and the "Formlength" will be displayed as blank lines after the last line on the page.

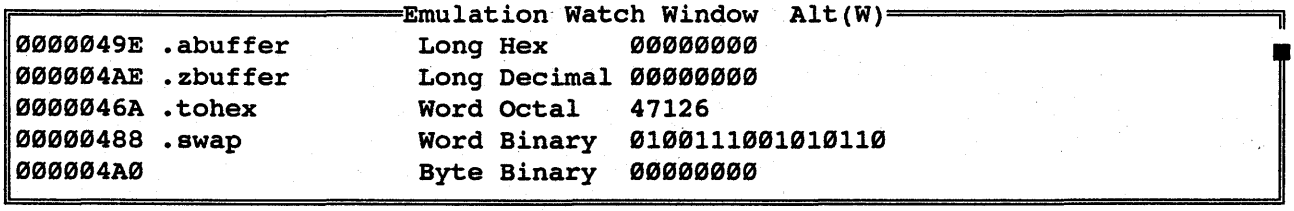
3.8.6.5 Trace Print "Page Length" Field

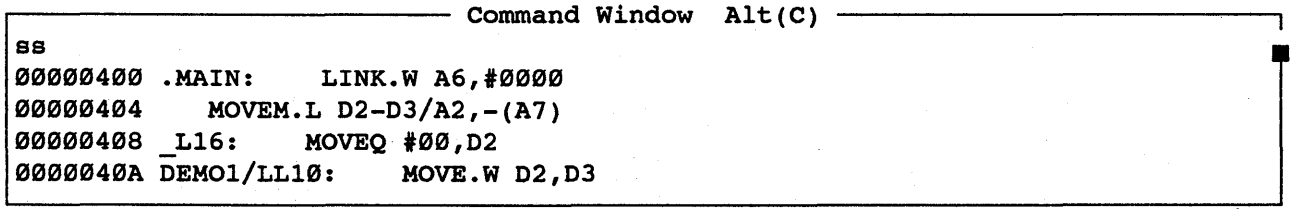
The Page Length is used to define the number of lines needed to make up a page. The difference between the "Pagelength" and the "Formlength" will be displayed as blank lines after the last line on the page.

3.8.6.6 Trace Print "Generate Page Header" Field

The Page Header is the column title for each column of data (e.g. CYCLE, ADDRESS, DATA, etc.). This option is available for each page as a guide for reading the printed pages.

3.9 The Watch Window

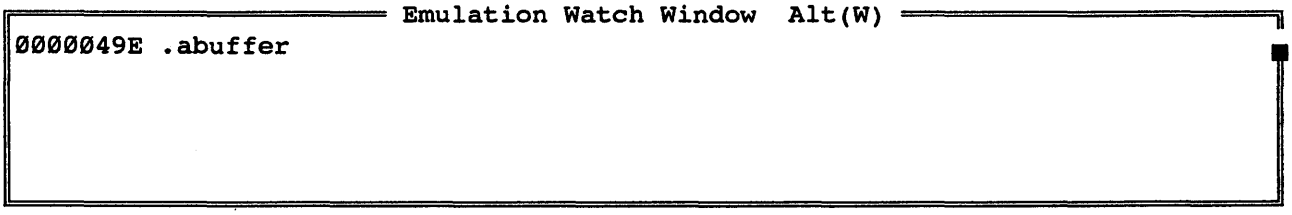




The watch window is used to select certain memory locations which are to be monitored continuously. Each time an emulator memory command is executed, or an emulator function is performed (such as single stepping or servicing breakpoints), the specified locations in the watch window are updated.

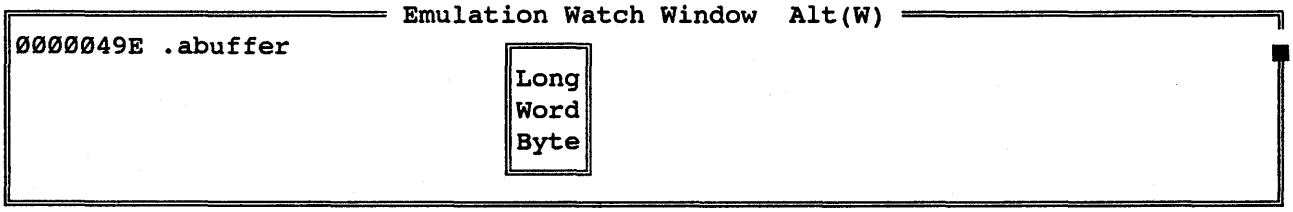
There are several fields in the Watch window that must be defined before certain locations can be "watched". Each of these fields are defined as follows:

3.9.1 Watch Window Address Field



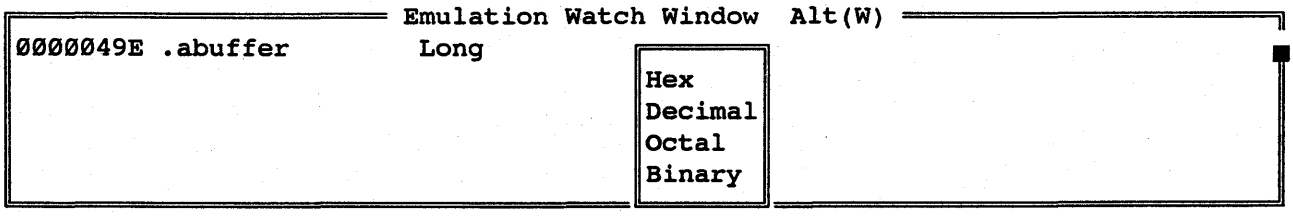
Type in the address to be monitored. To enter a symbol, type ":" <cr>. This will take you to the highlighted field next to the address field. Then type in the name of the symbol that is to be monitored. If you wish to exit the symbol field without entering a symbol, use ESC.

3.9.2 Watch Window Length of Display Field



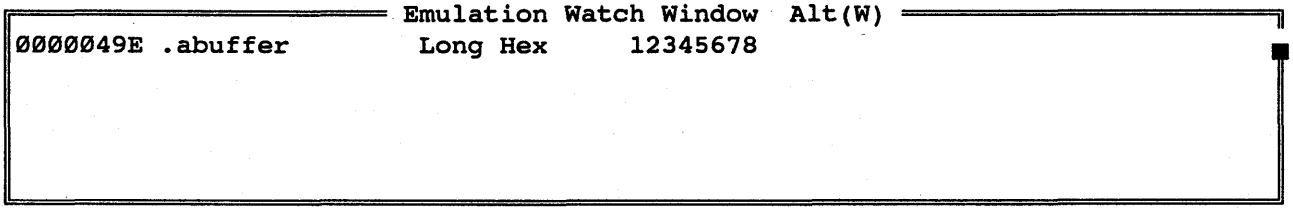
Enter the length of the value to be displayed. Use the arrow up and down keys to highlight the length desired and then enter a <cr> to select that type.

3.9.3 Watch Window Display Type Field



Select the format in which the results are to be displayed. Use the arrow up and down keys to highlight the type of display desired and then enter a <cr> to select that type.

The current value will now be displayed at the end of the line:



Note: The selection process can be terminated during any of the above steps by entering the ESC key.

Each time a memory value changes, the results will be highlighted.

To enter or exit the selection field, hit the '<cr>' key. Cursor movement is performed in the selection window by moving the cursor up or down using the "Up" and "Down" arrow keys. To select the highlighted function, enter a carriage return. Any function can alternately be selected by typing the function followed by a carriage return instead of using the selection window. However, this option is case sensitive and all functions must be entered exactly as they appear in the selection window. If the function was typed incorrectly, then the last valid function will re-appear in the field.

3.10 The Configuration Window

```
.Idling.....................................................................

  Memory Map -      Unallocated Memory = 0000K  |       Processor: 68010
  Block    Combine      Base      Ending   Block |  Clock     Emulator
  Number   Type         Address   Address  Size  |  Target System (BR,IPL0-2,
    1      UP UD SP SD 000000H   0FFFFFH  1024K |  BGACK,BERR):  Enabled
    2      -- -- -- -- 000000H   000000H  0000K |  Allow bus arbitration:
    3      -- -- -- -- 000000H   000000H  0000K |   Only During Emulation
    4      -- -- -- -- 000000H   000000H  0000K |  DTACK source is
 BLK 1   S/U = SUPV/USER  P/D = PROG/DATA       |      From the Emulator

       E = Emulator, T = Target System, RO = Read Only, RW = Read/Write
000000 E RW  02C000 E RW  058000 E RW  084000 E RW  0B0000 E RW  0DC000 E RW
004000 E RW  030000 E RW  05C000 E RW  088000 E RW  0B4000 E RW  0E0000 E RW
008000 E RW  034000 E RW  060000 E RW  08C000 E RW  0B8000 E RW  0E4000 E RW
00C000 E RW  038000 E RW  064000 E RW  090000 E RW  0BC000 E RW  0E8000 E RW
010000 E RW  03C000 E RW  068000 E RW  094000 E RW  0C0000 E RW  0EC000 E RW
014000 E RW  040000 E RW  06C000 E RW  098000 E RW  0C4000 E RW  0F0000 E RW
018000 E RW  044000 E RW  070000 E RW  09C000 E RW  0C8000 E RW  0F4000 E RW
01C000 E RW  048000 E RW  074000 E RW  0A0000 E RW  0CC000 E RW  0F8000 E RW
020000 E RW  04C000 E RW  078000 E RW  0A4000 E RW  0D0000 E RW  0FC000 E RW
024000 E RW  050000 E RW  07C000 E RW  0A8000 E RW  0D4000 E RW
028000 E RW  054000 E RW  080000 E RW  0AC000 E RW  0D8000 E RW
```

The Configuration window is used to establish the memory map and select hardware specific parameters. Many of these fields must be configured differently to work with specific target systems. Any field that is highlighted can be changed by moving the cursor to that field and hitting the space bar. Each of these fields is defined as follows:

3.10.1 Configuration "Processor" Field

This field shows the current processor configuration for the emulator. This field will toggle between 68000, 68010, and 68008 (when the 68008 adapter board is installed).

3.10.2 Configuration "Clock" Field

The source of the clock can be the "Emulator" on-board clock or the "Target System" clock.

3.10.3 Configuration "Target System (BR,IPL0-2,BGACK,BERR)" Field

These target system control signals can be Enabled or Disabled. "Enabled" means that these signals are active during emulation. "Disabled" means that these signals are gated off to the processor always. These signals should be disabled when the emulator is not connected to a target system.

3.10.4 Configuration "Allow Bus Arbitration" Field

Bus Arbitration can be selected to be enabled "Only During Emulation" or "Always". If it is enabled "Only During Emulation", then bus arbitration can only be activated when real-time emulation is in progress. This also requires that the target system control lines be enabled. If bus arbitration is selected to be active "Always", then it can occur at any point during emulator operation, even if the target system control signals have been disabled. This is particularly valuable for systems which use bus arbitration to perform refresh to dynamic memory. However, if bus arbitration is always enabled, a very active bus arbitration circuit may cause some noticable speed difference in the operation of the emulator.

3.10.5 Configuration "DTACK Source Is:" Field

The processor requires a DTACK signal to acknowledge all memory operations. This signal can come "From the Emulator", "From the Target System", or "According to Memory Map".

When running the emulator stand-alone without a target system, DTACK should be selected to come "From the Emulator". The "Target System Port Size" for this emulator generated DTACK signal should be entered at this time.

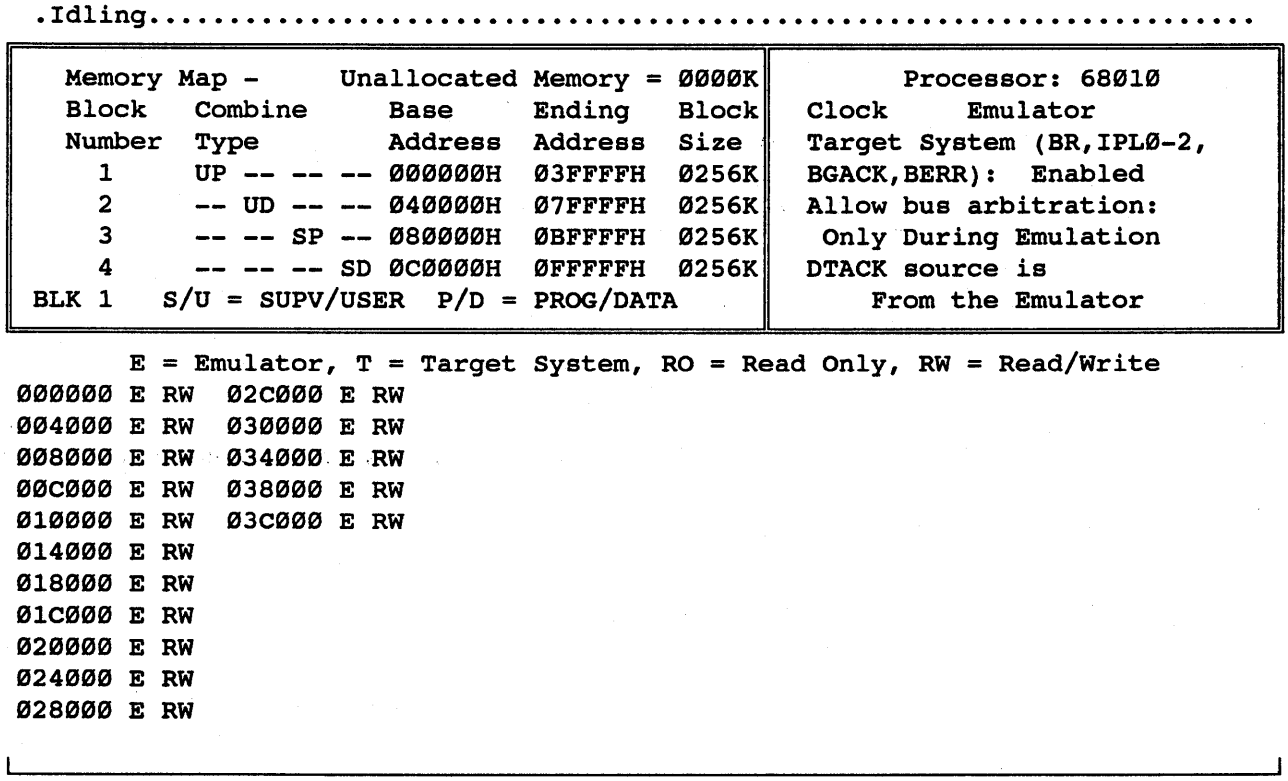
When plugged into a target system, DTACK can be selected to come "From the Target System" or "According to Memory Map". If "According to Memory Map" is chosen, then the DTACK signal will come from the emulator when the 68020 makes a memory access to emulator memory, or from the target system when the 68020 makes a memory access to target system memory.

3.10.6 Configuration Memory Mapping Fields

Many users will simply configure all of the emulation memory into one block or partition containing all four of the following combine types:

SD - Supervisor Data
SP - Supervisor Program
UD - User Data
UP - User Program

In the upper-left section of the Configuration window is a one column field for "Block Number", a four column field for "Combine Type" (one column each for UP,UD,SP,and SD respectively), fields for "Base Address", "Ending Address" and "Block Size", along with fields showing the memory map and page of the currently displayed block:



3.10.6.1 "Block Number" Field

Up to four blocks can be defined with various combinations of combine types, beginning and ending addresses, and block sizes.

3.10.6.2 "Combine Type" Fields

The system will not allow any of the parameters to be entered for a block until at least one combine type is selected. To select a combine type for a given block, move the cursor to the highlighted field where the block number's row meets the selected combine type column and press the space bar. The two letter abbreviation of the combine type should move to the selected block.

If, for example, block number one and combine type UP were selected, the left-most field of the combine type would contain the abbreviation "UP".

3.10.6.3 "Base Address" and "Ending Address" Fields

A contiguous block of memory can be defined by specifying a Base Address, an Ending Address, or a Block Size. Once defined, global operations can be performed on all 8K word blocks within the specified contiguous block by placing the cursor on the 'E', 'T', 'RW', 'RO', or 'Bit Port Size' fields and hitting the space bar.

3.10.6.4 "Unallocated Memory" Field

The total memory allocated in defined blocks may not exceed the total emulation memory. Unallocated memory is displayed at the top of the menu for quick reference. If the addresses or block size specified exceed the amount of unallocated memory or does not adhere to the boundary requirement, the system will change the entry to a value that will meet both criteria.

3.10.6.5 "Block Size" Field

A block's address boundaries may also be expressed by the base address and a block size. This block size is a multiple of 8K words. Any time the block size is entered along with a beginning address, the system will compute and display an ending address. Notice that the "Unallocated Memory" field will display the memory left (or unallocated) in the emulator.

3.10.6.6 "Block" Field

The "Block" field shows which block of memory is currently being displayed.

3.10.6.7 "E" and "T" Fields

To the right of each address in the memory table is a toggle field which will specifiy 'E' or 'T'. If the 'E' is selected, then the 8K word block of memory represented by this address will be referenced in the emulator. If the 'T' is selected, then the 8K word block of memory represented by this address will be referenced in the Target System.

3.10.6.8 "RW" and "RO" Fields

There is a second toggle field next to the 'E' or 'T' field which can toggle between 'RW' and 'RO'. This field is used for write protecting Memory in the emulator. Selecting an 'RW' will allow its associated 8K word block to be written to during emulation. Selecting an 'RO' will prevent its associated 8K word block from being written to during emulation or single stepping.

3.11 The Interface Window

.Idling.....

Interface Menu		Trace Print Format	
MAIN	AUXILIARY		
Autobaud ? YES DCE	DCE	Include Headers	YES
Baud rate 9600	Baud rate 9600	Page Length	066
Data bits 8	Data bits 8	Form Length	056
Parity NONE	Parity NONE		
Stop bits 2	Stop bits 2		
Protocol XON/XOFF	Protocol CTS/RTS		

The Interface menu shows the current configuration of the Main and Auxiliary RS232 ports.

3.11.1 Main Port Fields

The attributes of the main port cannot be changed from this menu (with the exception of the handshaking protocol), but are shown to reflect the switch setting for switch S1 or, for autobaud mode, the software switches entered when starting the program.

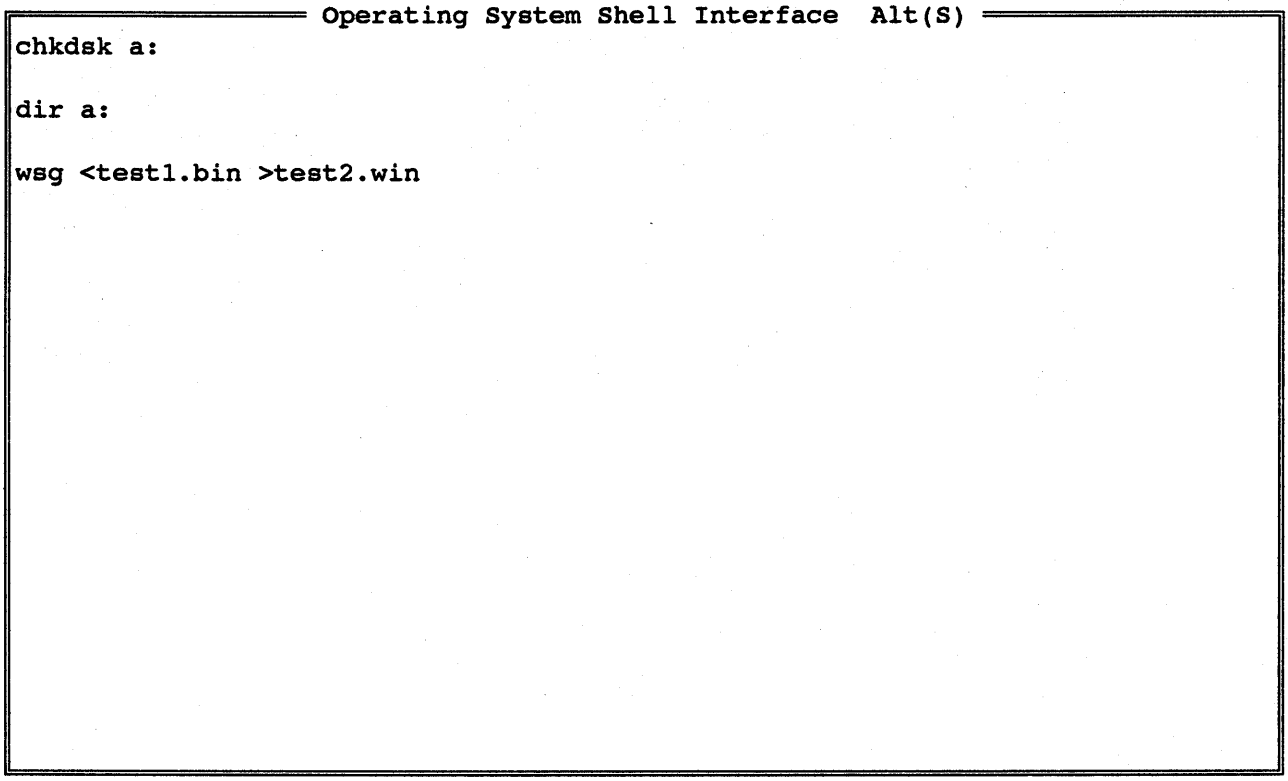
3.11.2 Auxiliary Port Fields

The configuration for the auxiliary port is configurable from this menu. The handshaking protocol (CTS/RTS or XON/XOFF) is selectable.

3.11.3 Trace Print Format Fields

The page format can be defined from this menu for the Trace Print command in the Trace Window. These are the same fields that can be defined in the "Trace Print Configuration" window.

3.12 The Operating System Shell Window



The Shell window can be entered by highlighting "Shell" in the "Main Emulation Command" Window and entering a <cr>.

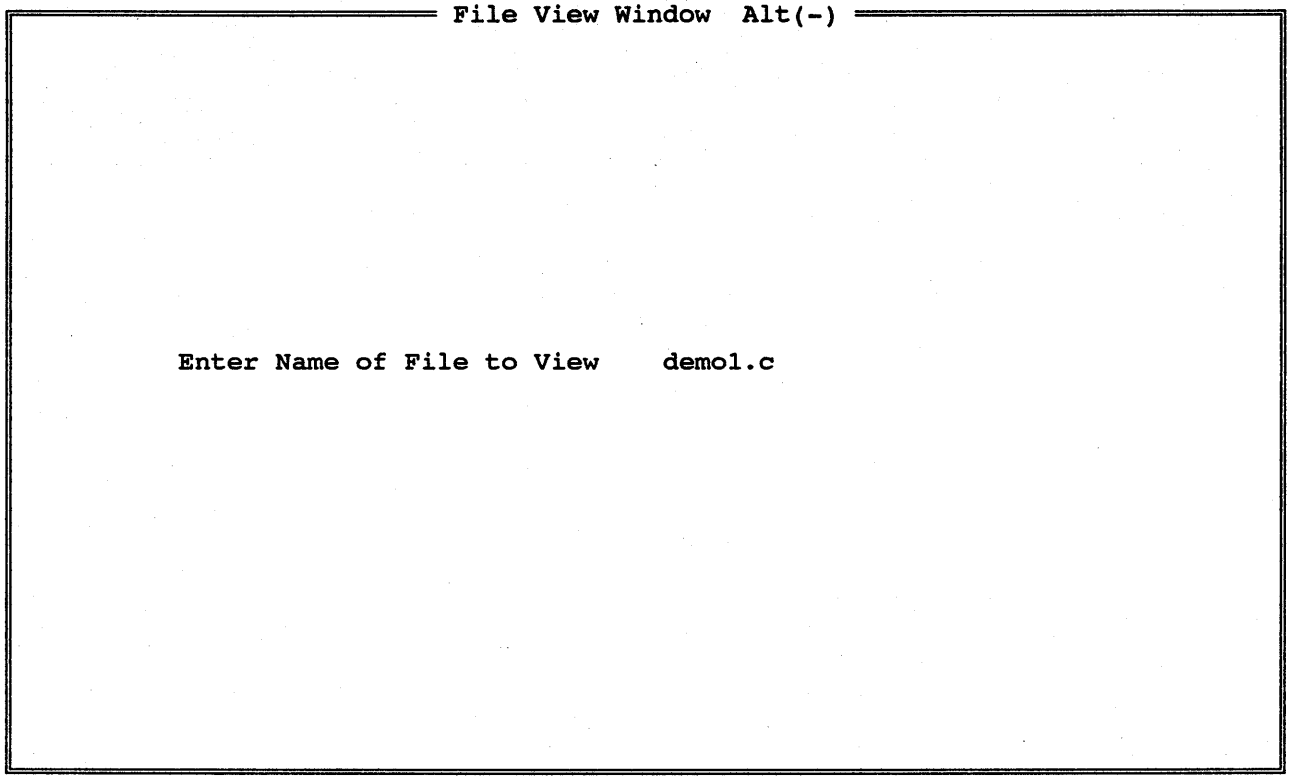
The Shell window allows operating system commands to be executed without exiting SourceGate. The Shell window will maintain a list of commands which have been executed. Any command can be re-executed by placing the command bar over it and hitting a return. ESC will exit the Operating System Shell window.

3.13 The View Window

View - demo1.c 17 lines 0 pages

```
extern  int     tohex();                /* Binary to HEX converter      */
extern          swap();                 /* Byte swap routine            */
char    abuffer[16];                    /* ASCII hex buffer             */
char    zbuffer[16];                    /* Zero buffer                  */
main()
{
        register int i;
        for (;;) {                                          /* "Forever"    */
                for (i = 0; i < sizeof abuffer; i++)
                        abuffer[i] = tohex(i);      /* Fill ASCII buffer */
                for (i = 0; i < sizeof zbuffer; i++)
                        zbuffer[i] = 0;             /* Fill Zero buffer  */
                for (i = 0; i < sizeof abuffer; i++)
                        swap(&abuffer[i],&zbuffer[i]);  /* Swap buffers */
        }
}
```

The View window is used to view any text file in your system. View will ask for the name of the file you wish to view and will display the file upon entering a <cr>:



This will display the source code for file Demo1.c as shown on the previous page.

ESC will exit the View window.

Chapter 4 Command Reference

4.0 Introduction

Emulation commands can be executed from the Command Window. Each command can be selected by typing its one or two character abbreviation.

4.1 Command Line Format

In the Command Window, the command abbreviation and the first parameter are not required to be separated, but may be separated by one or more spaces. If multiple parameter fields are used, they must be separated by a recognized delimiter. Allowed delimiters include a comma, a single space, or multiple spaces. If a comma is used as the delimiter, it may be preceded and followed by one or more spaces. This allows the user some flexibility in entering the commands in a form that is comfortable and also allows for minor typing errors.

All commands are entered in the Command Bar located in the Command Window. Once a command is entered, a carriage return will initiate execution of the command. The Command Bar can be moved forward or backwards in the Command window allowing previous commands and data to be reviewed. Any command can be re-executed by placing the Command Bar on a previously executed command and hitting a return.

4.1.1 Command Line Overrides

An override is an optional command parameter that allows the user to temporarily override the established global (default) parameter. Overrides are entered on the command line in brackets or parenthesis. More than one override may be specified in the brackets and several may be entered immediately adjacent to each other without separating spaces or delimiters.

Command line overrides do not apply to all commands. They are represented in the command format as a square brackets enclosing the override codes that apply to the command (the override codes and a brief description of each are given in the table below):

S (Supervisor Mode)

Memory addresses given in the command are assumed to be references into either the Supervisor Data or Supervisor Program memory partition. Which one is used will be determined by other overrides or global parameters.

U (User Mode)

Memory addresses given in the command are assumed to be references into either the User Program or User Data memory partition. Which one is used will be determined by other overrides or global parameters.

P (Program Mode)

Memory addresses given in the command are assumed to be references into either the User Program or Supervisor Program memory partition. Which one is used will be determined by other overrides or global parameters.

D (Data Mode)

Memory addresses given in the command are assumed to be references into either the User Data or Supervisor Data memory partition. Which one is used will be determined by other overrides or global parameters.

W (Word Mode)

The optional data size of input parameter or output (display) values is selected to be "Word".

B (Byte Mode)

The optional data size of input parameter or output (display) values is selected to be "Byte".

T (Target System)

Memory will be accessed in the Target System's memory space overriding the memory map as defined in the configuration menu.

R (Repeat)

Rd
Rx

This override will re-execute the current command in three optional formats.

a. If an R is entered alone, then the command will re-execute each time the carriage is entered until a period followed by a carriage return is entered.

b. If an R is entered followed by a number, then the command will re-execute 'd' times.

c. If an R is entered followed by an X, then the command will be re-executed continually until a carriage return is entered.

In the command descriptions that follow, the available override options for each command will be listed. The options available will be either the entire set of overrides listed above, a subset of the entire set, or none. The entire set of override options will be listed as (SUPDWBTR) with subsets consisting of other combinations.

4.2 Available Commands

By pressing "F1" in the Command Window, a listing of all the available commands will be displayed as follows:

FORMAT	SELECT FOR COMMAND LINE FORMAT
OVERRIDES	COMMAND LINE OVERRIDES
WINDOWS	SELECT FOR WINDOW OPERATION
'HOME'	LIST WINDOW SELECTION
'ESC'	EXIT/DELETE WINDOW

The following is a list of commands which can be executed from the Command Window:

A	ASSEMBLE
BM	BYTE DISPLAY MODE
C	COMPARE MEMORY BLOCK
CH	HALT ALWAYS ENABLED
CHDIR	PERFORM DOS CHDIR FUNCTION
CM	SHOW STATUS OF CONTROL BOARD MEMORY
CONFIG	RECALL FACTORY CONFIGURATION
CR	RESET ALWAYS ENABLED
D	DUMP MEMORY BLOCK
DIR	PERFORM DOS DIR FUNCTION
DL	DISABLE ADDRESS LATCHING TO TARGET SYSTEM
DM	DATA MEMORY MODE
d:	CHANGE DEFAULT DISK DRIVE
E	ENTER/CHANGE MEMORY
EH	ENABLE HALT DURING EMULATION ONLY
EL	ENABLE ADDRESS LATCHING TO TARGET SYSTEM
ER	RESET ENABLED DURING EMULATION ONLY
EXIT	RETURN TO THE OPERATING SYSTEM
F	FILL MEMORY BLOCK
G	BEGIN REAL-TIME EMULATION
I	INPUT COMMAND
L	LIST DISASSEMBLED CODE
M	MOVE MEMORY BLOCK
MT	TEST BLOCK OF MEMORY
N	ESTABLISH DEFAULT FILENAME FOR RD AND W
NH	DISABLE HALT LINE
NR	DISABLE RESET LINE
O	OUTPUT COMMAND
PD	PROGRAM WAIT STATES FOR DTACK COMMAND
PM	PROGRAM MEMORY MODE
QUIT	EXIT SOURCEGATE WITHOUT SAVING WINDOWS
RD	READ HEX FILE
RESET	RE-INITIALIZE EMULATOR
RS	RESET TARGET SYSTEM

SM	SUPERVISOR MEMORY MODE
SP	STOP PROGRAM EXECUTION
SR	SEARCH BLOCK OF MEMORY
SS	SINGLE STEP
ST	SHOW PROCESSOR STATUS
STF	TURN EMULATION STATUS WINDOW OFF
STI	SHOW EMULATION STATUS WINDOW DURING EMULATION
STO	TURN EMULATION STATUS WINDOW ON ALWAYS
TA	DEFINE TARGET ADDRESS
TR	SHOW STATUS FOR TRD/TRE
TRD	ADDRESS BUS TRI-STATE DISABLE
TRE	ADDRESS BUS TRI-STATE ENABLE
TYPE	PERFORM DOS TYPE FUNCTION
UM	USER MEMORY MODE
VER	DISPLAY VERSION NUMBER
W	WRITE MEMORY BLOCK
X	EXAMINE/CHANGE REGISTERS
:	SHOW/DELETE/EDIT SYMBOL VALUE
:=<ACDMS>	PERFORM SYMBOL MODULE FUNCTIONS (assembly, clear, display, mixed, source)
<	OPEN A BATCH FILE
!	TERMINATE AND STAY RESIDENT

4.3 Description of Commands

The commands listed in section 4.2 are described in detail with examples on how to use them. By pressing F1 in the command window and then pressing F1 again on the desired command, additional help information will be displayed in SourceGate.

4.3.1 A (Assemble) Command

```
Command Window  Alt(C)
a5000
005000  NOP     MOVEA.L (A6),A0
005002  NOP     MOVE.B (A1),D0
005004  NOP     MOVE.B D0,(A0)
005006  NOP     CMP.L D1,D0
005008  NOP     CMP.L A6 -- Invalid Entry --
005008  NOP     .

L5000
005000    MOVEA.L (A6),A0
005002    MOVE.B (A1),D0
005004    MOVE.B D0,(A0)
005006    CMP.L D1,D0
005008    NOP
00500A    NOP
00500C    NOP
00500E    NOP
005010    NOP
005012    NOP
005014    NOP
005016    NOP
005018    NOP
00501A    NOP
```

This command allows assembly language mnemonics to be entered into memory starting at the specified address (In-Line Assembly). Once the Assembly command is entered, the address will be displayed with the cursor to the right. At this point, 68020 assembly language instructions can be entered. After each entry, the emulator displays the next address and prompts for the next instruction. An error in the entered instruction will be flagged by displaying "--Invalid Entry--" and redisplaying the current address. A period (.) followed by a carriage return terminates the entry sequence and the command.

The format for the Assemble command is:

A([SUPDT]) Addr

where: Addr - is the hexadecimal starting address

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.2 BM (Byte Display Mode) Command

This command sets the default for input or output values to "Byte Mode".

ALSO SEE SECTION 4.1.1

4.3.3 C (Compare Memory Block) Command

```
Command Window  Alt(C)

C 400 40C 500
000400 4E56  000500 0F0F
000402 0000  000502 0F0F
000404 48E7  000504 0F0F
000406 3020  000506 0F0F
000408 7400  000508 0F0F
00040A 3602  00050A 0F0F
00040C 247C  00050C 0F0F

C :.MAIN  :.MAIN + C  500
000400 4E56  000500 0F0F
000402 0000  000502 0F0F
000404 48E7  000504 0F0F
000406 3020  000506 0F0F
000408 7400  000508 0F0F
00040A 3602  00050A 0F0F
00040C 247C  00050C 0F0F
```

The Compare Command will compare one block of memory to another. The first word in the first block will be compared to the first word in the second block, and so on. Any non-comparisons will be displayed on the screen showing the address in the first block with its data value and the address in the second block with its data value. If the two blocks match, then nothing will be displayed.

The format for the Compare command is:

C([SUPDTR]) Baddr1 Eaddr1 ([SUPDT])Baddr2

where:

Baddr1 - is the hexadecimal beginning address of the first compare block

Eaddr1 - is the hexadecimal ending address of the first compare block

Baddr2 - is the hexadecimal beginning address of the second compare block

The size of the first block sets the number of words to be compared.

For each word in block one that does not match the corresponding word in block two, a line is displayed on the screen.

This display line has the format:

Addr1 Value1 Addr2 Value2

where: Addr1 - is the address of the word in block one

Value1 - is the contents of addr1

Addr2 - is the address of the word in block two

Value2 - is the contents of addr2

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.4 CH (Constant HALT) Command

```
Command Window  Alt(C)
CH
 HALT line is enabled.  (CH)
 RESET line is enabled.  (CR)
```

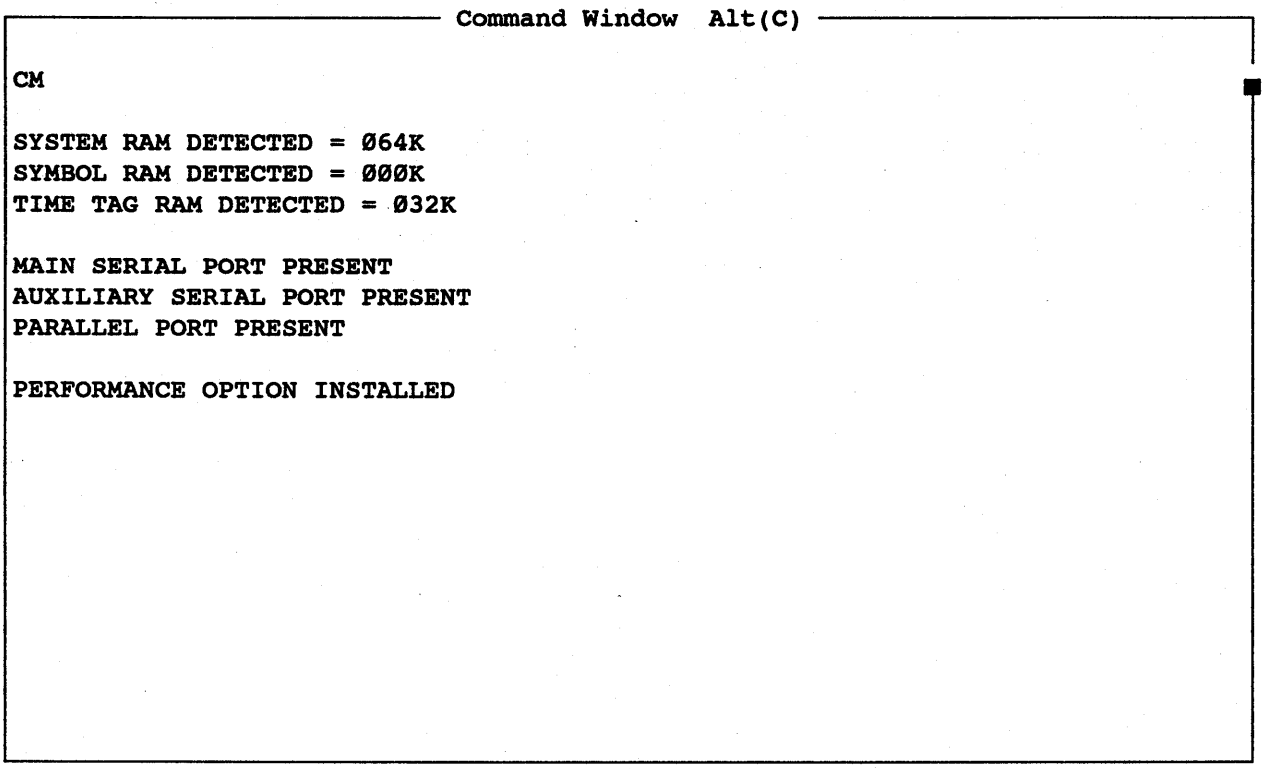
The execution of this command enables the HALT signal to the processor at all times. The HALT signal can control the emulator during emulation and while idling. Use the EH (Emulation HALT) command to enable the HALT signal only during emulation.

4.3.5 CHDIR [dir] (Change Directory) Command

Perform the DOS CHDIR function.

where [dir] is a DOS directory

4.3.6 CM (Show Status Of Control Board Memory) Command



Executing this command will show the emulator options that are installed. It should be noted that "System RAM" in this case refers to the system RAM inside the emulator (not the PC) and "Symbol RAM" refers to the optional 256K of RAM which allows symbolic debug when using the emulator with a dumb terminal. The SourceGate program does not use this RAM for symbolic or source-level debugging.

4.3.7 CONFIG (Recall Factory Configuration) Command

This command will return the emulator menus, ports, memory map, predefined sequences and other battery backed-up configuration values to the factory configuration.

4.3.8 CR (Constant RESET) Command

```
Command Window  Alt(C)
CR
 HALT line is enabled.  (CH)
 RESET line is enabled.  (CR)
```

The execution of this command enables the RESET signal to the processor at all times. The RESET signal can control the emulator during emulation and while idling. Use the ER (Emulation RESET) command to enable the RESET signal only during emulation.

4.3.9 D (Dump Memory Block) Command

```
Command Window  Alt(C)

D[B]400 46F
000400 4E 56 00 00 48 E7 30 20 74 00 36 02 24 7C 00 00  NV..H.0 t.6.$|..
000410 04 9E 2F 02 4E B9 00 00 04 6A 15 80 30 00 58 8F  ../.N....j..0.X.
000420 52 82 70 10 B0 82 6E E2 74 00 30 02 24 7C 00 00  R.p...n.t.0.$|..
000430 04 AE 42 32 00 00 52 82 70 10 B0 82 6E EC 74 00  ..B2..R.p...n.t.
000440 30 02 24 7C 00 00 04 AE 48 72 00 00 30 02 20 7C  0.$|....Hr..0. |
000450 00 00 04 9E 48 70 00 00 4E B9 00 00 04 88 50 8F  ....Hp..N.....P.
000460 52 82 70 10 B0 82 6E D8 60 9E 4E 56 00 00 22 2E  R.p...n.`.NV.."."

D[W]400 46F
000400 4E56 0000 48E7 3020 7400 3602 247C 0000  NV..H.0 t.6.$|..
000410 049E 2F02 4EB9 0000 046A 1580 3000 588F  ../.N....j..0.X.
000420 5282 7010 B082 6EE2 7400 3002 247C 0000  R.p...n.t.0.$|..
000430 04AE 4232 0000 5282 7010 B082 6EEC 7400  ..B2..R.p...n.t.
000440 3002 247C 0000 04AE 4872 0000 3002 207C  0.$|....Hr..0. |
000450 0000 049E 4870 0000 4EB9 0000 0488 508F  ....Hp..N.....P.
000460 5282 7010 B082 6ED8 609E 4E56 0000 222E  R.p...n.`.NV.."."
```

The Dump Command will display the contents of memory in the specified address range showing the hexadecimal data with its ASCII equivalent on the screen.

If no ending address is given, then one screen of data will be displayed and the command will terminate. If no starting address is specified then the last ending address (whether user entered or default) will be used as the starting address. Using this method of defaulting allows the user to dump a display of memory from one starting address and get the next and subsequent pages of data dumped by simply typing the "D" and a carriage return.

The Format for the Dump Command is:

D([SUPDWBTR]) (Baddr) (Eaddr)

where: Baddr - is the hexadecimal beginning address of the block of memory to be displayed

Eaddr - is the hexadecimal ending address of the block of memory to be displayed

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.10 DIR (Directory) Command

Under PC-DOS or MS-DOS, typing "DIR" on the emulator command line will run the DOS directory function.

4.3.11 DL (Disable Address Latching To Target System) Command

```
Command Window  Alt(C)
DL
 Address latching is disabled.
```

This command is used to restore the emulator's address latches to a mode in which all addresses which are generated by the processor are placed on the target system address bus. In this mode, the emulator will be reading from various addresses in the target system. If this is unacceptable, use the EL (Enable Latch) command.

4.3.12 DM (Data Memory Mode) Command

The execution of this command sets the default memory category to be Data Memory mode. Using the SM (Supervisor Mode) or UM (User Mode) commands further define the default memory mode.

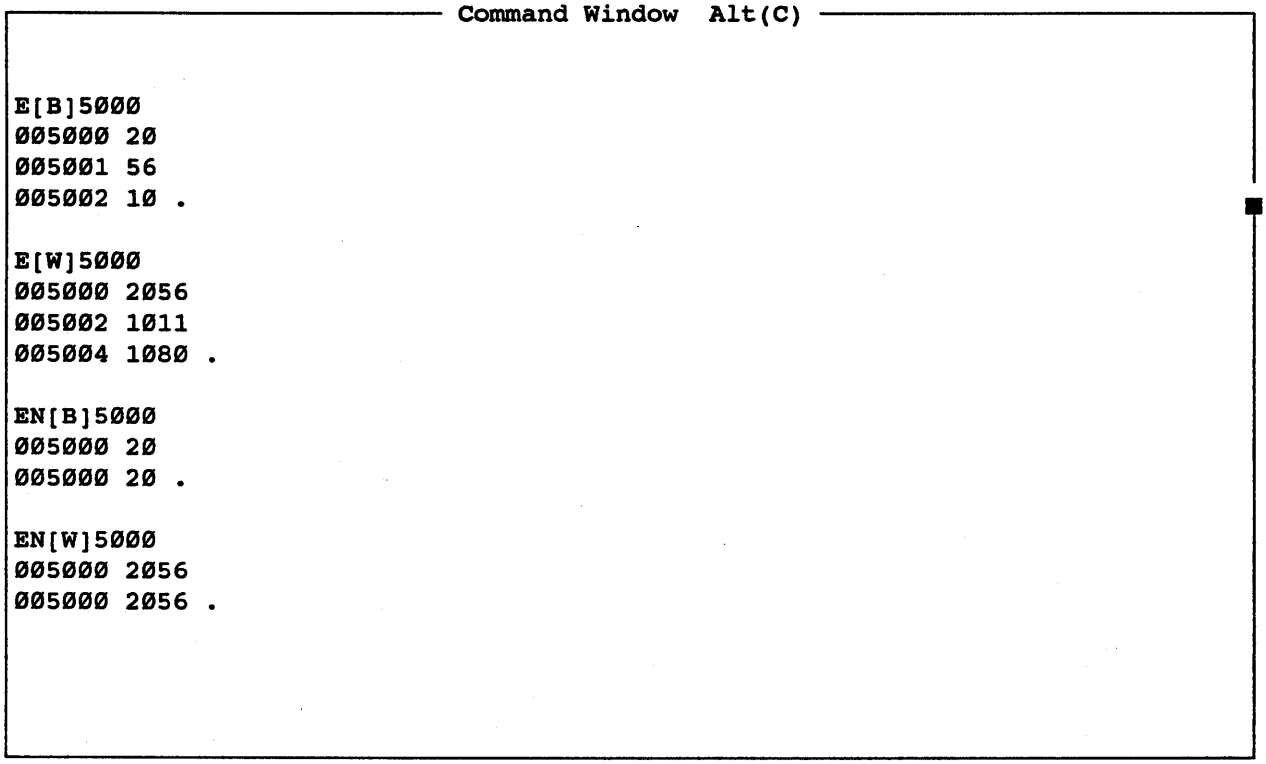
ALSO SEE SECTION 4.1.1

4.3.13 d: (Change Default Disk Drive) Command

Perform the DOS function to change the default (logged) drive.

where:
d: is a DOS disk drive letter.

4.3.14 E/EN (Enter) Command



The Enter Command is used to examine memory at the specified location with the option to change its contents. After executing the Enter Command, the selected address will appear on the screen with the current data. The data can be altered by entering new data and pressing a carriage return, or it can remain unchanged by just typing a carriage return. In either case the next address will be displayed with its data. This command can be terminated by typing a period (.) in the data field followed by a carriage return.

There is a variation of the Enter command (EN) that does not advance to the next address. Whether you enter a change and a carriage return or just a carriage return, it will display the value at the specified address only. This feature is very useful in monitoring a memory mapped I/O location.

The Format for the Enter Command is:

E([SUPDWBT]) Addr

or

EN([SUPDWBT]) Addr

where: Addr - is the hexadecimal address of the data value to be displayed for possible substitution

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.15 EH (Emulation HALT) Command

```
Command Window  Alt(C)
EH
 HALT line is disabled.  (EH)
 RESET line is enabled.  (CR)
```

The execution of this command enables the HALT signal to the processor only during emulation. The HALT signal can control the emulator during emulation but not while idling. Use the CH (Constant HALT) command to enable the HALT signal at all times.

4.3.16 EL (Enable Address Latching To Target System)

```
Command Window  Alt(C)
EL
 Address latching is enabled.
```

The emulator is designed to be as flexible as possible in its interface to a wide variety of target system designs. Generally, when the emulator is plugged into the target system, the address lines are changing, reflecting the emulator's on-board monitor which the processor is executing. For the most part, this does not interfere with the operation of the target system.

There are a few cases, however, where a target system will not function properly when random addresses (reads) are placed on its bus. An example of this type of situation would occur when the user has configured the emulator to use target system DTACK and the target system's memory has parity check. When the emulator and target system are powered up, the emulator processor begins to run internal code while the target system interprets the emulator's operation as memory reads. When memory reads are performed on uninitialized memory with parity, then parity errors are bound to occur. The parity error may cause the target system to fail to return a DTACK signal, thus causing the emulator to time-out.

The Enable Latch (EL) command is available to remedy this situation. The EL command will force the emulator to latch out the last valid address used by the target system. For example, after a Reset command is issued, the address 00000006 will remain on the target system address bus. Another example would be if memory were dumped from 100 to 150, then the address 00000150 would be latched on the target system address bus.

If the EL command has been activated, and the emulator tries to read memory where no DTACK signal will be returned, then the emulator will time-out. Instead of latching out the address which caused the DTACK time-out, the emulator will force a dummy read to a known good address, initially 0 (see TA command), thus latching out a valid address which will always return a DTACK on each memory read which the emulator performs.

4.3.17 ER (Emulation RESET) Command

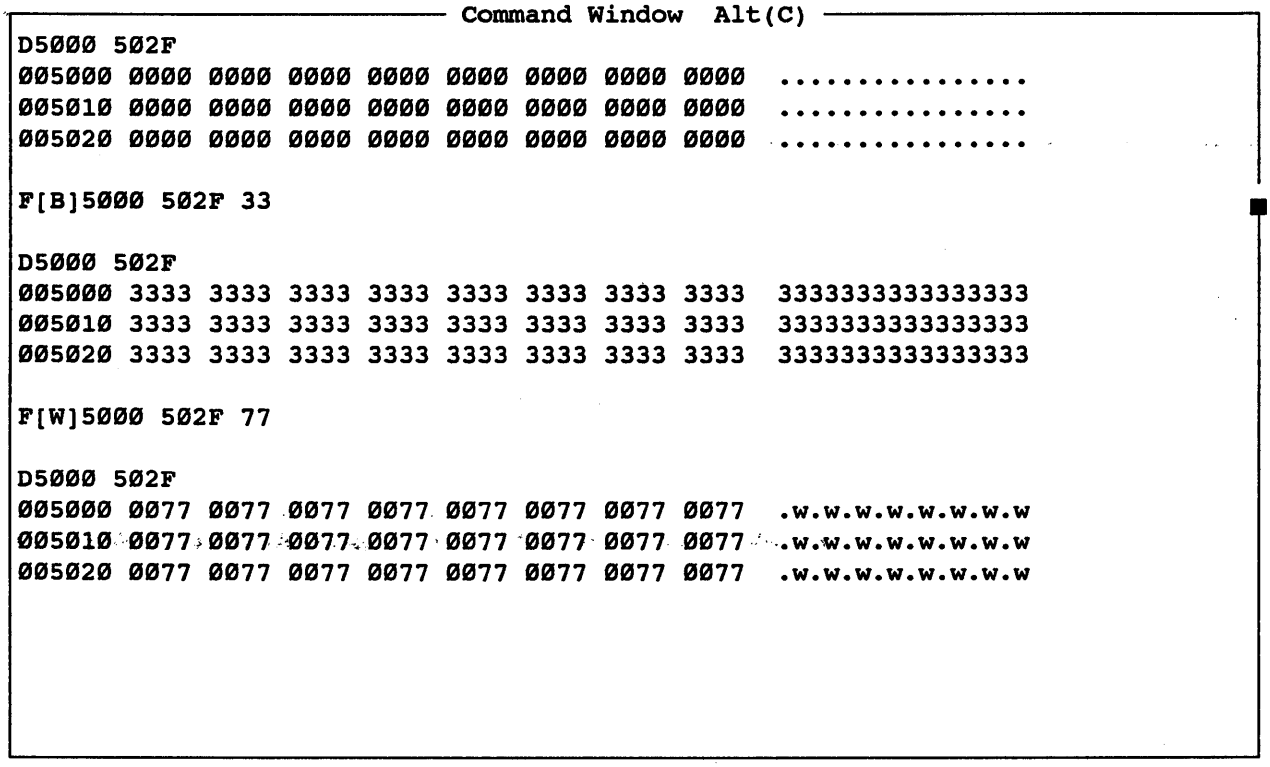
```
Command Window  Alt(C)
ER
 HALT line is disabled.  (EH)
 RESET line is disabled.  (ER)
```

The execution of this command enables the RESET signal to the processor only during emulation. The RESET signal can control the emulator during emulation but not while idling. Use the CR (Constant RESET) command to enable the RESET signal at all times.

4.3.18 EXIT Command

Typing "EXIT" at the emulator command bar will cause SourceGate to exit and return to the operating system. All Window positions and sizes will be saved in the Inx file so that re-positioning and sizing will not be necessary the next time SourceGate is called up.

4.3.19 F (Fill) Command



The Fill Command is used to fill memory with a constant. The given data value will be written to each memory location within the specified range. The value may be a byte or a word in length, determined by the global parameters and overrides used.

The format for the Fill Command is:

F([SUPDWBTR]) Baddr Eaddr Val

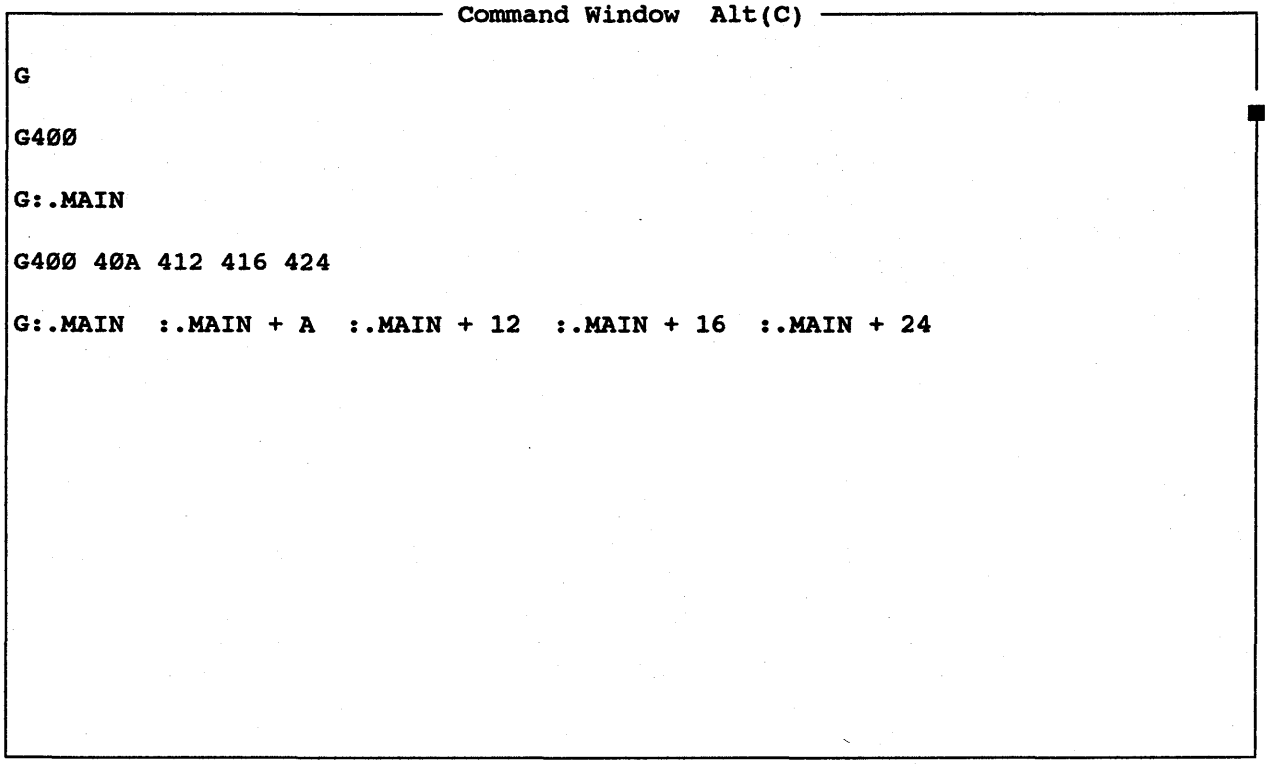
Where: Baddr - is the hexadecimal beginning address of the memory block to be filled

Eaddr - is the hexadecimal ending address of the memory block to be filled

Val - Constant value to be used to fill the memory block

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.20 G (Go) Command



The Go Command is used to start real-time emulation. The options for this command are to run with temporary breakpoints as set in this command, or without a breakpoint, in which case any break conditions which occur are due to the configuration of the events from the Event windows and the breakpoint sequence as defined in the Sequence Break Point window.

If the option for running with breakpoints from this command is selected, then it is important to note that the configuration of the Event and the Sequence windows become inactive during execution of this command.

In either case emulation will begin at the start address given or default to the current program counter. If a starting address is to be used it must be immediately adjacent to the G (otherwise it will be taken as a breakpoint address).

The format for utilizing the Event and Sequence menu configurations is:

G(Saddr)

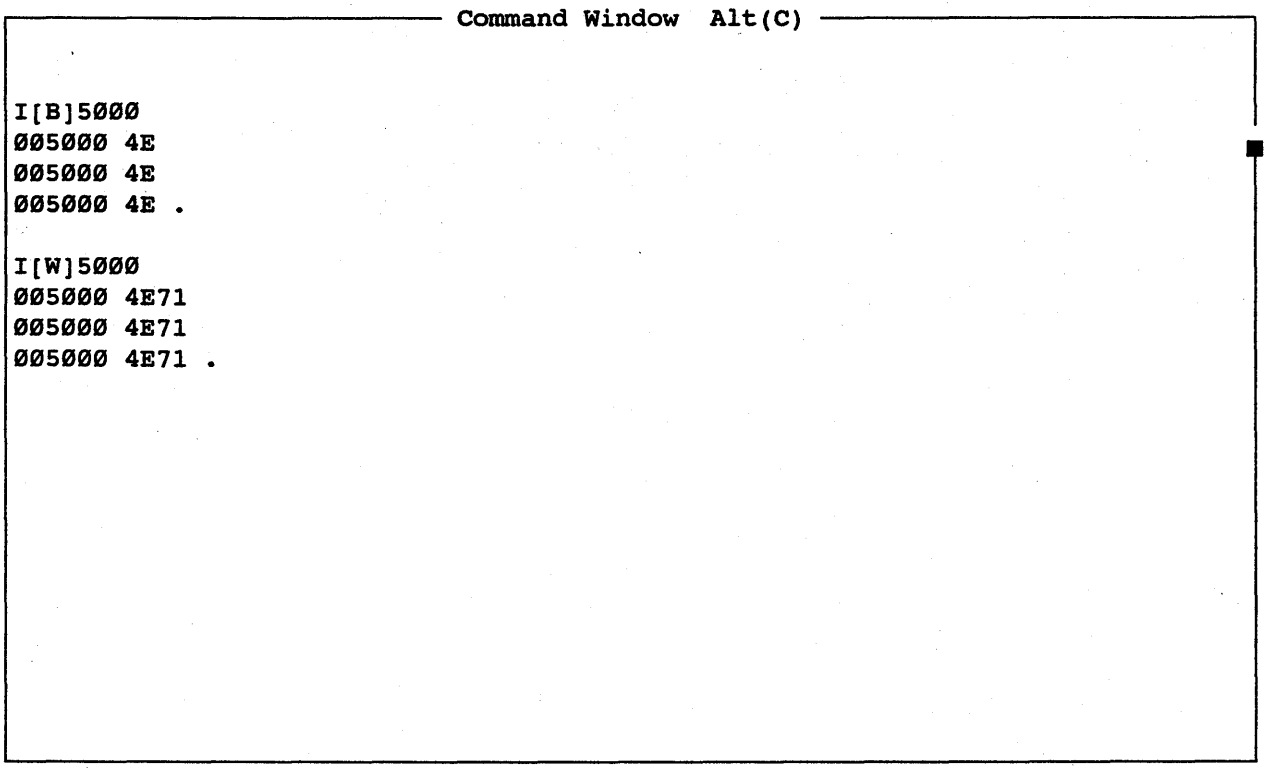
The format for setting temporary breakpoints is:

G(Saddr) Braddr1 (Braddr2 Braddr3 Braddr4)

Where: Saddr - is the hexadecimal address where execution (real-time emulation) will begin

Braddr1,
Braddr2,
Braddr3,
Braddr4 - are hexadecimal address where execution (real-time emulation) will stop (or break) on op-code execution

4.3.21 I (Input) Command



The I command is used to read and display a certain memory address. This function is very useful when monitoring memory mapped I/O locations. Each time a <cr> is entered, the memory location is read and shown.

The format for the Input command is:

I([SUPDWBT]) Addr

Where: Addr - is the hexadecimal address of the data value to be read

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.22 L (List Code) Command

```
Command Window  Alt(C)
L40A 424
DEMO1:10:                abuffer[i] = tohex(i);     /* Fill ASCII buffer */
00040A DEMO1/LL10:     MOVE.W D2,D3
00040C     MOVEA.L #0000049E:.ABUFFER,A2
000412     MOVE.L D2,-(A7)
000414     JSR.L [00]00046A:.TOHEX
00041A     MOVE.B D0,00(A2,D3.W)
00041E     ADDQ.L #4,A7
000420     ADDQ.L #1,D2
000422     MOVEQ #10,D0
000424

L:.MAIN + A  :.MAIN + 24
DEMO1:10:                abuffer[i] = tohex(i);     /* Fill ASCII buffer */
00040A DEMO1/LL10:     MOVE.W D2,D3
00040C     MOVEA.L #0000049E:.ABUFFER,A2
000412     MOVE.L D2,-(A7)
000414     JSR.L [00]00046A:.TOHEX
00041A     MOVE.B D0,00(A2,D3.W)
00041E     ADDQ.L #4,A7
000420     ADDQ.L #1,D2
000422     MOVEQ #10,D0
000424
```

This command is used to disassemble the program memory data into assembly language mnemonics. All data and addresses are displayed in hexadecimal.

If no ending address is specified, then one screen of disassembled mnemonics will be displayed and the command will terminate. If no beginning address is specified then the last ending address (whether entered or default) will be used as the beginning address. This allows the user to disassemble code one display screen at a time by entering the command and an initial starting address for the first display and for the next and each subsequent page only the "L" and a carriage return need be entered.

If a range is given, disassembled mnemonics will be displayed from the starting address to the ending address, scrolling the screen if necessary. To pause a scrolling screen enter CNTRL-S and, to resume scrolling a CNTRL-Q. This can also be accomplished by using the "pause" key to stop scrolling and <cr> to resume scrolling. To terminate the command while scrolling depress the space bar or enter a carriage return.

The format for the List Command is:

L([SUPDTR]) (Baddr) (Eaddr)

Where: Baddr - is the hexadecimal beginning address of the memory data to be disassembled

Eaddr - is the hexadecimal ending address of the memory to be disassembled

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.23 M (Move) Command

```
Command Window  Alt(C)

D400 41F
000400 4E56 0000 48E7 3020 7400 3602 247C 0000  NV..H.0 t.6.$|..
000410 049E 2F02 4EB9 0000 046A 1580 3000 588F  ../.N....j..0.X.

M400 41F 5000

D5000 501F
005000 4E56 0000 48E7 3020 7400 3602 247C 0000  NV..H.0 t.6.$|..
005010 049E 2F02 4EB9 0000 046A 1580 3000 588F  ../.N....j..0.X.

M:.MAIN  :.MAIN + 1F 8000

D8000 801F
008000 4E56 0000 48E7 3020 7400 3602 247C 0000  NV..H.0 t.6.$|..
008010 049E 2F02 4EB9 0000 046A 1580 3000 588F  ../.N....j..0.X.
```

The Move Command is used to copy the contents of one block to the contents of another block (non-destructive move). Block one is defined with its starting and ending addresses being the first two address fields displayed, and block two is defined with its starting address being the third address field. The contents of the first memory location of block one is copied to the first memory location of block two, and so on. The next prompt character is not displayed until the move is complete.

The format of the Move Command is:

M([SUPDTR]) Baddr1 Eaddr1 ([SUPDT])Baddr2

where:

Baddr1 - is the hexadecimal beginning address of the data "Source" block

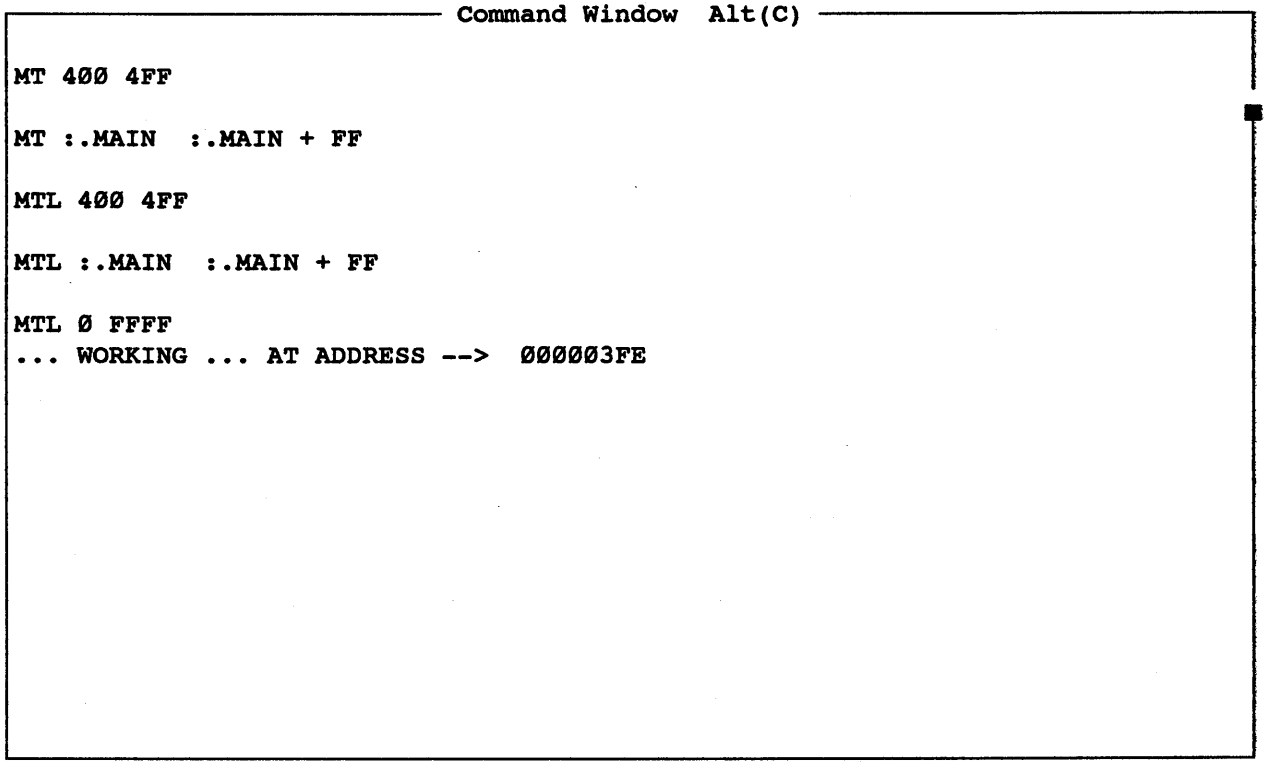
Eaddr1 - is the hexadecimal ending address of the data "Source" block

Baddr2 - is the hexadecimal beginning address of the data "Destination" block

The size of the "Source" block defines the number of bytes to be copied into the "Destination" block.

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.24 MT/MTL (Memory Test) Command



This command can be used to test memory (RAM) within the given address range.

The short memory test (MT) writes, reads and compares two separate bit patterns for each byte within the range. The bit patterns are 55H and AAH, thus cycling all bits of the byte through both one and zero.

The long memory test (MTL) is a walking bit pattern. This test writes, reads and compares the byte while having only one bit in the byte on at a time. This is done by using the Hexadecimal values 1, 2, 4, 8, 10, 20, 40, 80 for each byte within the range.

In either test, any read not comparing with what was written into a location will cause the address, data written and data read to be displayed on the next line of the screen. Both of these tests are non-descructive and can be performed without corrupting data currently in memory. A <cr> will terminate this test at any time.

The format for the Memory Test command is:

MT([SUPDTR]) Baddr Eaddr

The format for the long Memory Test command is:

MTL([SUPDTR]) Baddr Eaddr

Where: Baddr - is the hexadecimal beginning address of the memory block to be tested.

Eaddr - is the hexadecimal ending address of the memory block to be tested.

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.25 N filename,ABHMS Command

The 'N' command is used to establish a default filename for use with subsequent 'RD' and/or 'W' commands.

ABHMS are switches which control how an hex file is read/written:

- A - Read files into the emulator using ASCII hex file transfer.
- B - Read files into the emulator using Binary hex file transfer.
- H - Read only the data record portions of the hex file. Ignore symbol records.
- S - Read only the symbol record portions of the hex file. Ignore data records.
- M - Treat a file of 'S' records as an Hitachi hex file. (Otherwise a file of 'S' records is considered to be a Motorola hex file.)

4.3.26 NH (No HALT) Command

```
Command Window  Alt(C)
NH
 HALT line is disabled.  (NH)
 RESET line is disabled.  (ER)
```

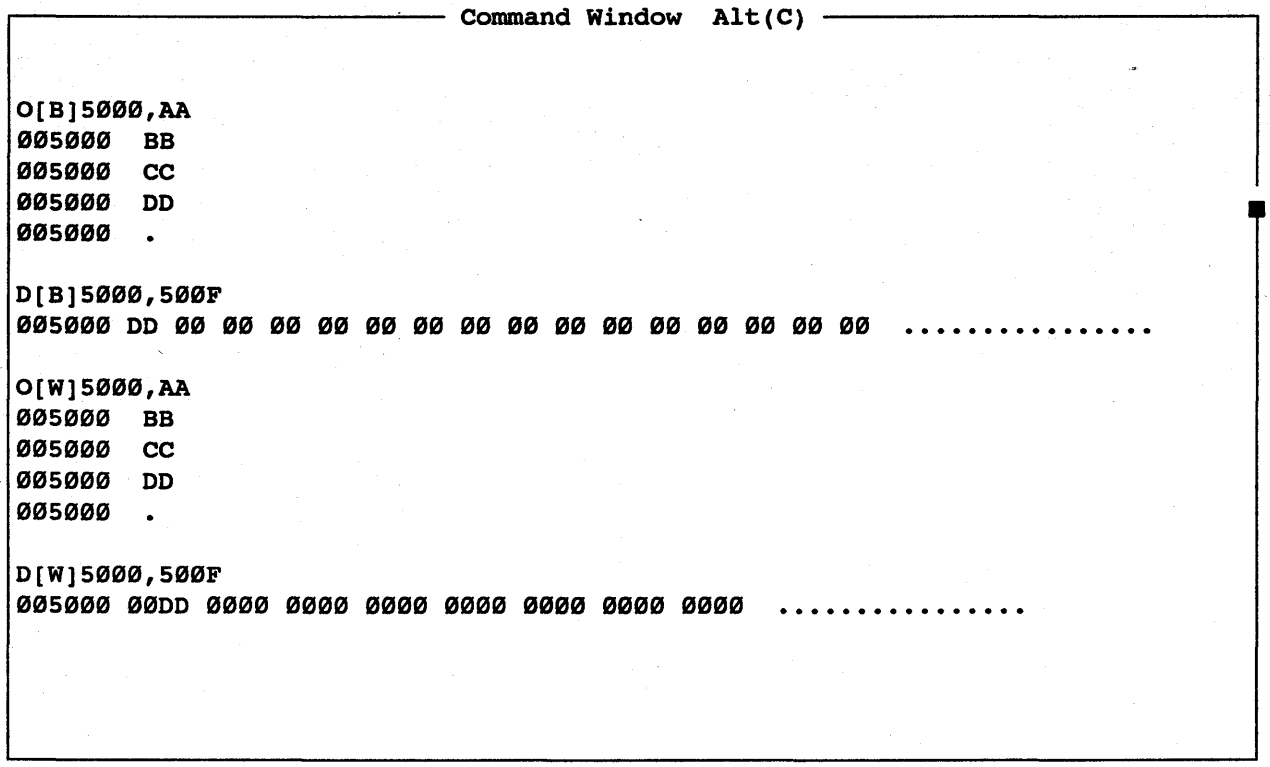
This command disables the HALT signal to the processor at all times. The HALT signal cannot be transmitted or received by the emulator during emulation or while idling.

4.3.27 NR (No RESET)

```
Command Window  Alt(C)
NR
 HALT line is disabled.  (NH)
 RESET line is disabled.  (NR)
```

This command disables the RESET signal to the processor at all times. The RESET signal cannot be transmitted or received by the emulator during emulation or while idling.

4.3.28 O (Output) Command



The O command is used to write to a certain memory address. This function is very useful when testing memory mapped I/O locations. Each time a <cr> is entered, the desired value is written to the displayed address. The O command will not increment the address to be written but will write the same address each time a <cr> is entered.

The format for the Output command is:

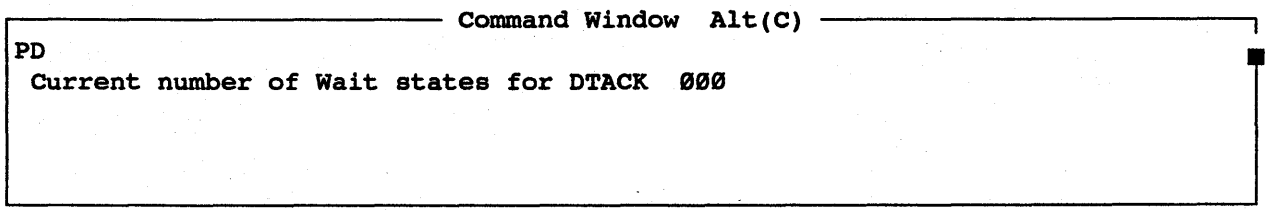
O([SUPDWBT]) Addr,Val

Where: Addr - is the hexadecimal address to be written with the defined value

Val - is the hexadecimal value to be written

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.29 PD (Program DTACK Wait States) Command



This command is used to program the emulator to insert a predefined number of wait states into each memory cycle. Each wait state is one clock cycle in length. The wait state are inserted on any memory cycle in which the emulator is supplying the DTACK signal. The wait states are programmed by typing:

PD count

Where: count - is the number of wait states to be inserted within the range of 0-254

Typing PD<cr> will show the current number of wait states selected.

4.3.30 PM (Program Memory Mode) Command

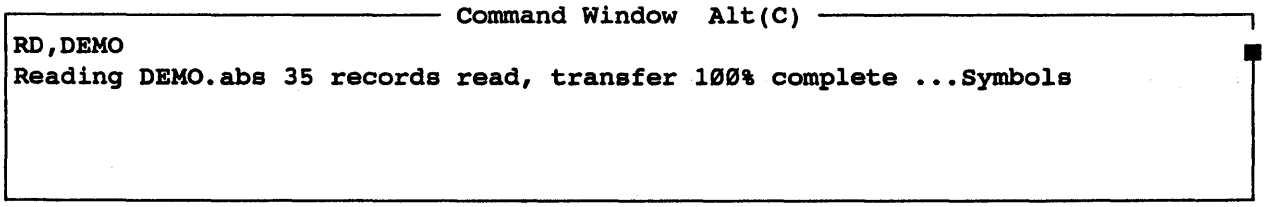
The execution of this command sets the default memory category to be Program Memory mode. Using the SM (Supervisor Mode) or UM (User Mode) commands further define the default memory mode.

ALSO SEE SECTION 4.1.1

4.3.31 QUIT Command

Entering the Quit command will exit SourceGate without updating the Inx file with the latest Window changes. To exit SourceGate with an Inx update, use the Exit Command.

4.3.32 RD (Read) Command



This command is used to transfer a hexadecimal file from a host computer to the emulation memory. The file name and RS232 port (main or auxiliary) can be specified with this command. When downloading a file using the Huntsville Microsystems, Inc. communication program (SourceGate), the ASCII file is converted to binary before being transferred to the emulator. This essentially doubles the speed of the transfer over the normal ASCII transfer. When downloading from a host which doesn't use SourceGate, a standard ASCII transfer will occur.

The file formats accepted by the SourceGate software are Motorola S-Record, Intel Hex, Texhex, HMI Binary, Extended Tek Hex, Microtec Research Absolute and Hitachi.

Once the Read Command has been executed, it can be terminated by typing a CNTRL - X.

When reading a file from the main RS232 port, the format is:

RD(Offset),Filename

When reading a file from the auxiliary RS232 port the format is:

RDX(Offset),(String)

where:

Offset - is a hexadecimal value representing the address offset

Filename - is a file name, valid to the host computer, that designates the file to be down loaded to the emulator. If no extension is specified, the default .ABS will be used. Note: A filename is not necessary when using the RDX command.

String - is any string of characters, followed by a carriage return, which the user wants to be transmitted out the auxiliary port. Typically this string would be a command to a computer to transmit a file to the emulator.

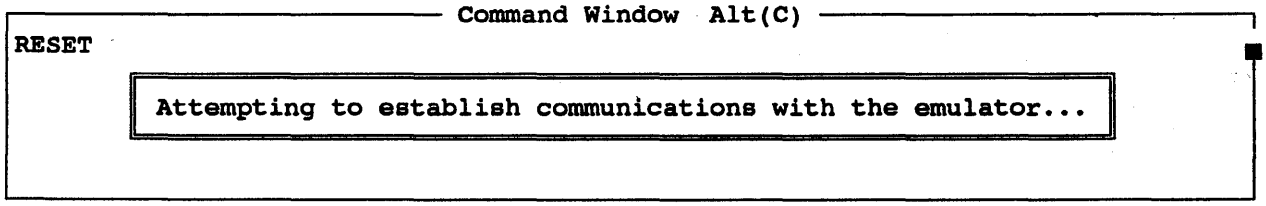
The 'RD' command opens the named format file (.ABS being the default extention) and reads its symbols into the internal SourceGate symbol table, and its hex data into the emulator's (or target system's) memory.

If the optional read bias is given, this offset value will be added to the addresses of all bytes read from the file, before these bytes are stored in memory. This offset is a hexadecimal number.

If the filename is not given, the 'RD' command will use the filename given in the last 'RD' or 'W' or 'N' command.

4.3.33 Reset (Re-initialize Emulator) Command

By typing "Reset" on the command line, this will re-initialize the emulator:



4.3.34 RS (Reset Target System) Command

This command will issue a reset to the 68Ø2Ø processor. All registers will be updated to the value which they contain upon a processor reset. The Program Counter is initialized to the contents of memory locations 4H and 6H and the Stack Pointer is initialized to the contents of memory locations ØH and 2H.

4.3.35 SM (Supervisor Memory Mode) Command

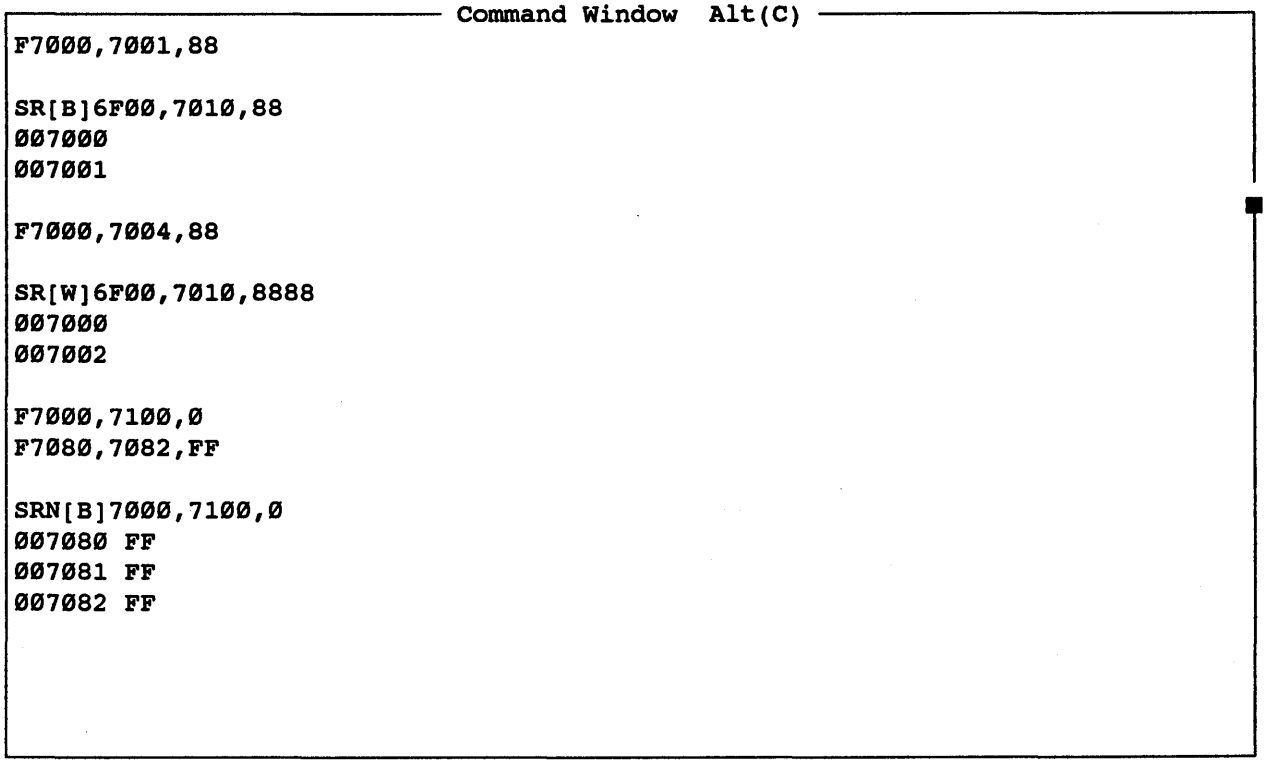
The execution of this command sets the default memory category to be Supervisor Memory mode. Using the PM (Program Mode) or DM (Data Mode) commands further define the default memory mode.

ALSO SEE SECTION 4.1.1

4.3.36 SP (Stop) Command

This command is used to stop emulation in order to regain control of the processor. Upon entering this command, emulation will break, and the message "PROCESSOR IS IDLING" will be displayed and the register window will be updated with the current register values.

4.3.37 SR/SRN (Search) Command



This command can be used to search memory between two addresses for a particular data pattern. When a match is found, the address is displayed. To search for the next occurrence, hit the Carriage Return key. To terminate this command, type a period. If no match is found, then nothing is displayed.

There is also a Search Not option available. In this mode the value is searched for, but the command displays the address of all memory locations in the given range that do not match the search value. This is useful when checking a block of memory that should contain a constant value, such as zero or the value of a previous Fill command.

The format for the Search command is:

SR([SUPDWBTR]) Baddr Eaddr Val

The format for the Search Not command is:

SRN([SUPDWBTR]) Baddr Eaddr Sval

where:

Baddr - is the hexadecimal beginning address of the block of memory to be searched

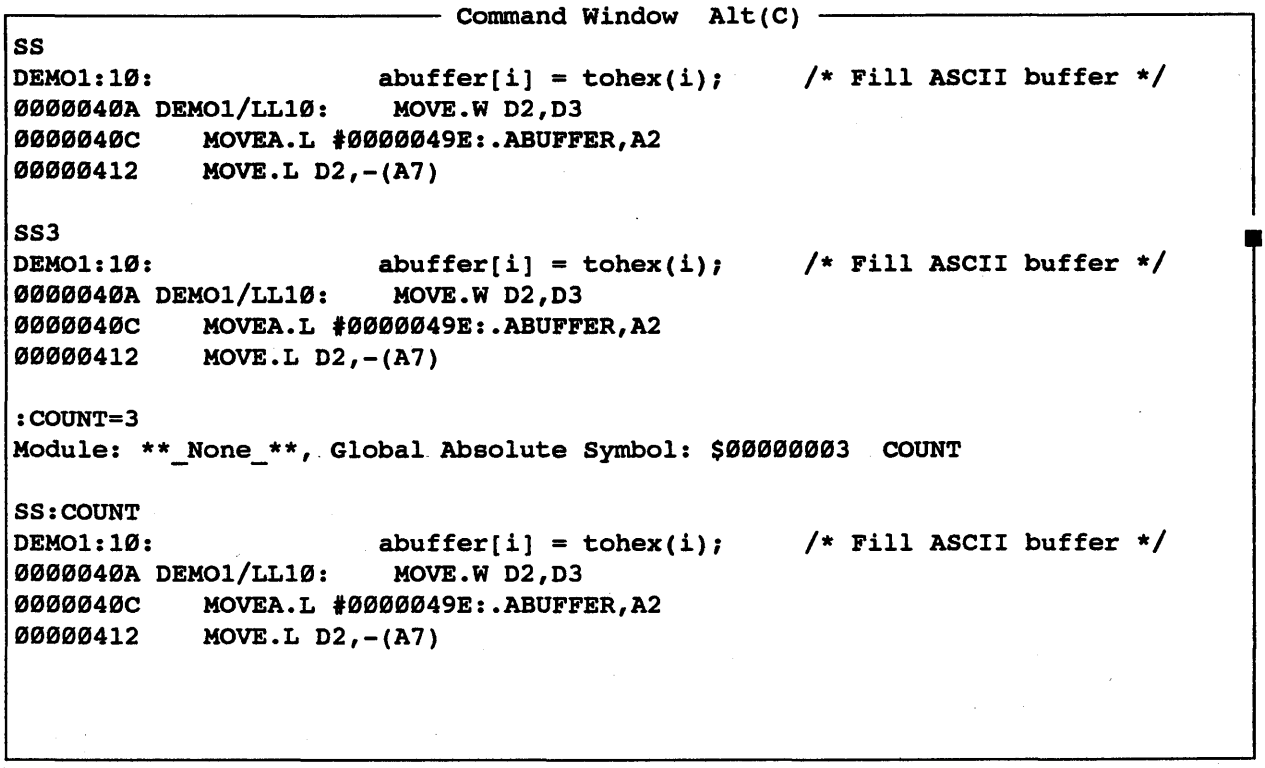
Eaddr - is the hexadecimal ending address of the block of memory to be searched

Val - is the data value to be searched for and will be 1 byte or word value or or a 1-20 character ASCII string enclosed in single quotes.

Sval - is the data value to be searched for and will be a byte value, a word value or one or two ASCII characters enclosed in quotes.

FOR COMMAND LINE OVERRIDES (SUPDWBTR), SEE SECTION 4.1.1

4.3.38 SS/SSX (Single Step) Command



This command is used to execute one instruction at a time. Each time a step is performed, the address and instruction to be executed are displayed in both the register and command windows. Therefore, multiple steps will display a history of program flow. After execution of the instruction is performed, the register window will display the register values, that were changed by execution of the last instruction, and both the register window and command window will show the updated program counter and next instruction to be executed.

There are three options for operating the Single Step Command. The first option is to single step every time a carriage return is entered. To terminate this mode, type a period. The second option is to single step a specified number of times. The third option is to single step continuously until a carriage return is entered. To terminate the second and third options, a carriage return should be entered.

The format for Single Stepping once for each carriage return entered is:

SS

The format for Single Stepping a specified number of times is:

SSCount

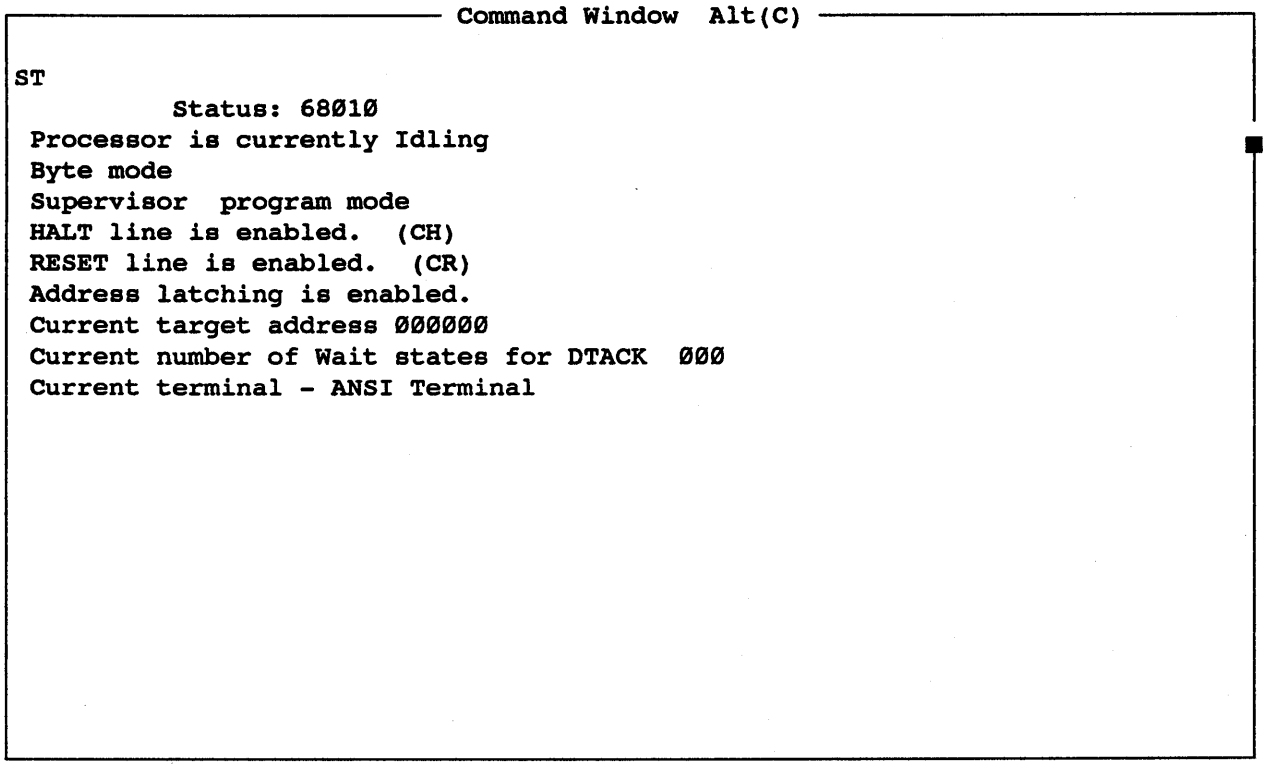
The format for Single Stepping continuously until a carriage return is entered is:

SSX

Where:

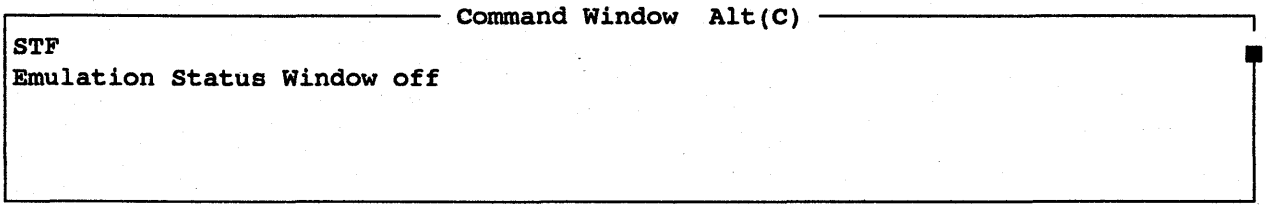
Count - is the number of iterations to single step

4.3.39 ST (Status) Command



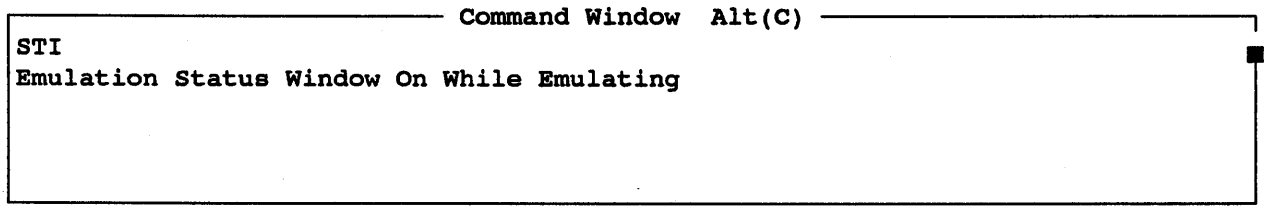
Executing the Status command will display the current status of the emulator.

4.3.40 STF (Status Window Off) Command



Executing the Status Window Off command will disable the emulation status window from appearing on the screen.

4.3.41 STI (Status Window Intermediate) Command



Executing the Status Window Intermediate command will display the emulation status window only during emulation. After emulation is terminated, the emulation status window will disappear as soon as the next key is typed.

4.3.42 STO (Status Window On) Command

```
Emulation Status
Status       : Reset Command Received
Trace        : Trigger Trace Inactive
Pass Count A: 32AD B: 32AD C: 32AD D: 32AD
Timer        :                    Ø µsec.
Emulation Status
```

```
Command Window  Alt(C)
STO
Emulation Status Window on
```

Executing the Status Window On command will display the emulation status Window at all times.

4.3.43 TA (Target Address) Command

```
Command Window  Alt(C)
TA
 Current target address 000000
```

If the EL (Enable Latch) command has been activated and the emulator tries to read memory where no DTACK signal will be returned, then the emulator will time-out. Instead of latching out the address which caused the DTACK time-out, the emulator will force a dummy read to a known good address, initially 0. If this default address is not a good address in which to perform a dummy read, then the Target Address can be changed by typing:

TAaddr

Where: addr - is the address which is to be latched onto the target system's address bus after any DTACK time-out

Typing TA<cr> will show the current default target address.

4.3.44 TR (Show Status For TRD/TRE) Command

```
Command Window  Alt(C)
TR
Target System buffers tri-state each cycle
```

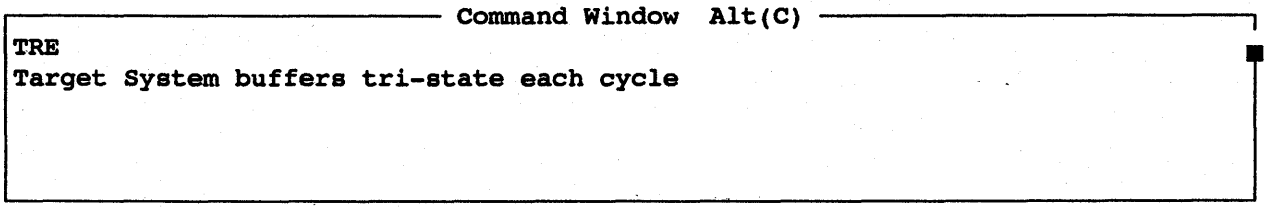
Executing this command will show the status of the tri-state buffers to the target system.

4.3.45 TRD (Tri-State Disable) Command

```
Command Window  Alt(C)
TRD
Target System buffers do not tri-state each cycle
```

Executing this command will allow the address signals to be active during the entire processor cycle. This assures that the active address signals are available on the target system's address bus as soon as possible. Use the TRE (Tri-State Enable) command to tri-state the address bus in the inactive portion of the address strobe (AS').

4.3.46 TRE (Tri-State Enable) Command



Executing this command will cause the address bus to be tri-stated during the inactive portion of the address strobe (AS'). To disable the tri-state function, execute the TRD (Tri-State Disable) command.

4.3.47 TYPE Command

```
                    Type File [DEMO1.C]
extern  int     tohex();                /* Binary to HEX converter      */
extern          swap();                 /* Byte swap routine            */
char    abuffer[16];                    /* ASCII hex buffer             */
char    zbuffer[16];                    /* Zero buffer                  */
main()
{
        register int i;
        for (;;) {                                          /* "Forever"    */
                for (i = 0; i < sizeof abuffer; i++)
                        abuffer[i] = tohex(i);      /* Fill ASCII buffer */
                for (i = 0; i < sizeof zbuffer; i++)
                        zbuffer[i] = 0;             /* Fill Zero buffer  */
                for (i = 0; i < sizeof abuffer; i++)
                        swap(&abuffer[i],&zbuffer[i]);  /* Swap buffers */
        }
}


      Hit Enter to Return from type window
```

On PC-DOS or MS-DOS, giving the 'TYPE' command at the emulator command level prompt causes SourceGate to run the DOS TYPE command, with the given filename as its argument:

```
                    Command Window  Alt(C)
TYPE DEMO1.C
```

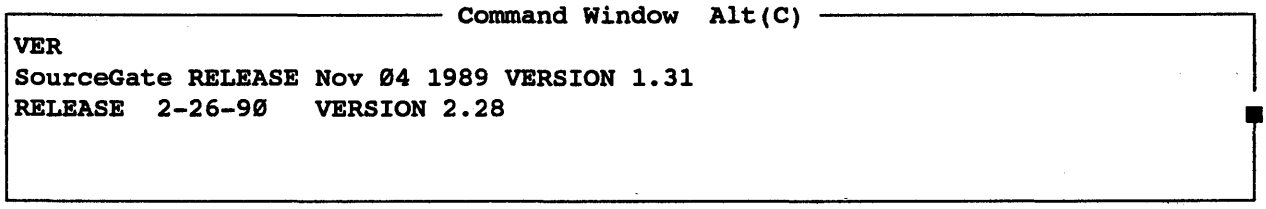
This will display the type screen shown above for file "Demo1.c".

4.3.48 UM (User Memory Mode) Command

The execution of this command sets the default memory category to be User Memory mode. Using the PM (Program Mode) or DM (Data Mode) commands further define the default memory mode.

ALSO SEE SECTION 4.1.1

4.3.49 VER (Version) Command



This command displays the current version and release date of the SourceGate and emulator firmware.

The top line of the display shows the version of SourceGate and the bottom line shows the version of emulator firmware installed in the unit.

4.3.5Ø W/WX (Write) Command

```
Command Window  Alt(C)
W7ØØØ,71ØØ TEST
Writing TEST.abs 74 records written ...Symbols
```

This command is used to transfer files from the emulation memory to a host computer. The size of the block being transferred is specified by the two address fields activated by this command. When using the Huntsville Microsystems, Inc. communication software (SourceGate), the block will be written as an Intel Hex file with symbol information added to the beginning of the file. This symbol information is in a format that is easily read by HMI software. The extent of this file defaults to .ABS. This symbol/hex file is written to disk on the host computer under the specified filename.

The format for writing to the main RS232 port is:

W Baddr Eaddr Filename

The format for writing to the auxiliary RS232 port is:

WX Baddr Eaddr

where:

Baddr - is the hexadecimal beginning address of code to be written

Eaddr - is the hexadecimal ending address of code to be written

Filename - is a filename, valid to the host computer, that designates the file to be uploaded (written) to the host computer. If no extension is specified, the default .ABS will be used.

The 'W' command opens the given file for writing, and writes the contents of the SourceGate internal symbol table into this file, along with the data bytes in memory, from Baddr through Eaddr.

'Baddr' and 'Eaddr' are hexadecimal numbers (or symbols).

4.3.51 X/XFL (Examine Registers and Flags) Command

```
Command Window  Alt(C)
X
PC=00000400 .MAIN:    LINK A6,#0000
SSP=00001000   USP=FFFFFFFF   SR=2719 - NT S I7 XT NG NZ NV CY
D0=FFFFFFFF    D1=AAAAFFFF    D2=F7FF7FFF    D3=FFFFFFFF
D4=FFFFFFFF    D5=FFFFFFFF    D6=FFFF7FFF    D7=FFFF7FFF
A0=31FC0000    A1=060442C0    A2=31FC5555    A3=06044EF8
A4=002031FC    A5=AAAA0604    A6=4EF80020    A7=00001000

XSR=2700

X
PC=00000400 .MAIN:    LINK A6,#0000
SSP=00001000   USP=FFFFFFFF   SR=2700 - NT S I7 NX PS NZ NV NC
D0=FFFFFFFF    D1=AAAAFFFF    D2=F7FF7FFF    D3=FFFFFFFF
D4=FFFFFFFF    D5=FFFFFFFF    D6=FFFF7FFF    D7=FFFF7FFF
A0=31FC0000    A1=060442C0    A2=31FC5555    A3=06044EF8
A4=002031FC    A5=AAAA0604    A6=4EF80020    A7=00001000

XFL
NT S I7 NX PS NZ NV NC

XPC
PC= 00000400
```

This command allows the user to examine and modify the registers and flags (status register). There are three options to the examine command. The first option is to display all registers and flags. The second option is to display a specific register and allow it to be modified. The last option is to allow the flags to be modified by typing in the condition code status.

The format for displaying the registers and condition codes is:

X

The format for displaying a specific register and allowing it to be modified is:

XReg

The format for displaying just the condition code values is:

XFL

Where:

Reg - is any processor register such as A1 or SR

FL - are the condition codes (flags)

An explanation of each condition code is listed below:

Bit Location in SR	Flag or Condition Code Name	Not Active Code	Active Code
15	Trace Mode	NT	TR
14	Not Defined	--	--
13	Supervisor/User State	U	S
12	Not Defined	--	--
11	Not Defined	--	--
10	Interrupt Priority Mask 2 ---\		
9	Interrupt Priority Mask 1 ----	I0 through I7	
8	Interrupt Priority Mask 0 ---/		
7	Not Defined	--	--
6	Not Defined	--	--
5	Not Defined	--	--
4	Extend	NX	XT
3	Negative	PS	NG
2	Zero	NZ	ZR
1	Overflow	NV	OV
0	Carry	NC	CY

4.3.52 : (Symbol Editing) Commands

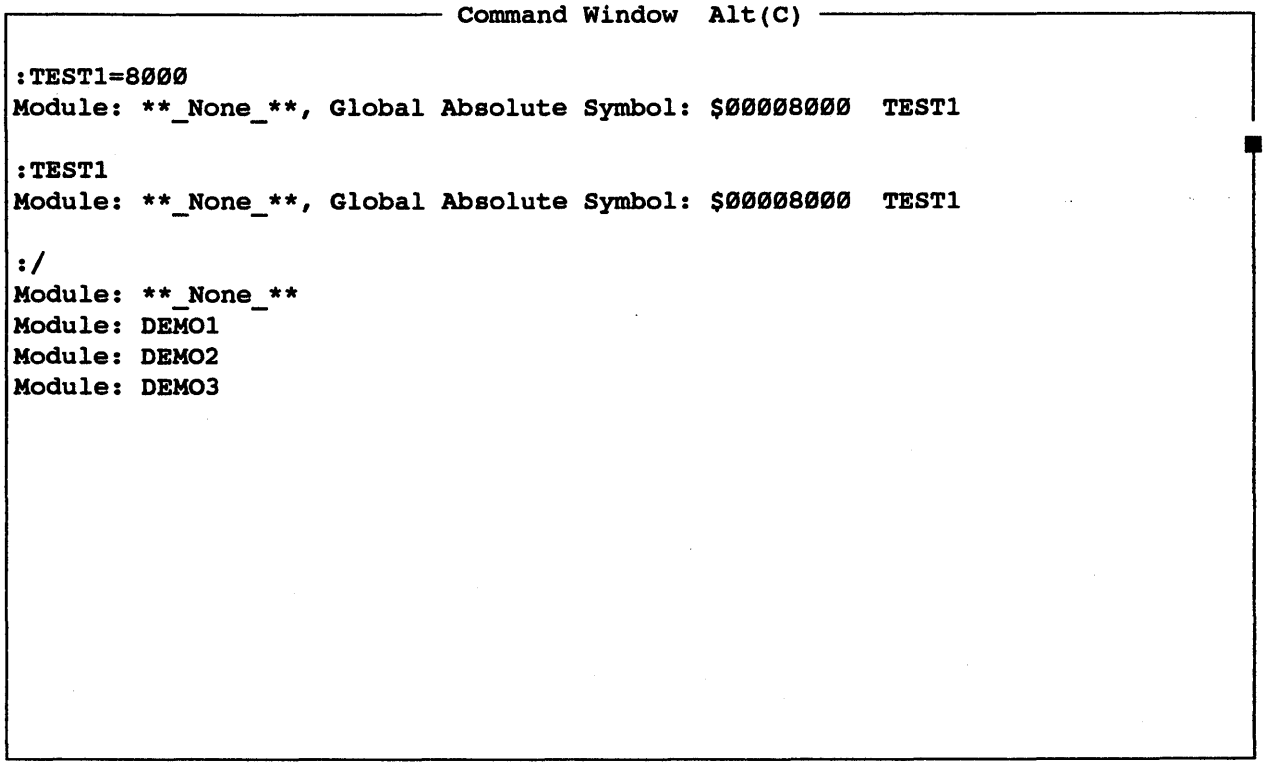
This command and its variations is used for symbol viewing and editing.

The ':' command is used for symbol table display and editing. The 'symbol', '=', and 'value' portions of this command are all optional. This command comes in many different formats:

:

':' given by itself on a line causes SourceGate to display the entire contents of its internal symbol table:

```
Command Window  Alt(C)
:
Module: **_None_**
 Absolute Symbols:
  COUNT    $00000003

Module: DEMO1
 Absolute Symbols:
  .MAIN    $00000400
  .ZBUFFER $000004AE
  .ABUFFER $0000049E
  _L4      $00000440
  _L7      $0000042A
  _L16     $00000408
  LL1      $00000408
  LL2      $00000408
  LL10     $0000040A
  LL3      $00000408
  LL11     $00000428
  LL4      $00000408
  LL12     $0000042A
  LL5      $00000408
  LS0      $00000400
  LL6      $00000408
```



:/

':' followed by a '/' character causes SourceGate to display the names of all modules defined in the current symbol table.

:symbol

':' followed by a symbol will cause SourceGate to search for and display the first occurrence of the named symbol in the internal symbol table.

:module/symbol

':' followed by a module name, followed by a '/', followed by a symbol name will cause SourceGate to search for the named symbol in or following the named module in the symbol table. If the symbol is found, it will be displayed.

:symbol=
:module/symbol=

These ':' commands perform symbol searches in the same fashion as outlined above for symbol displays, but when the named symbol is found by SourceGate, it will be deleted from the internal symbol table.

:symbol=value
:module/symbol=value

These ':' commands perform symbol lookup as outlined above, but when the named symbol is found, the old value of the symbol is discarded, and the given value substituted for it.

'value' may be given as a hexadecimal number, or may be given as the name of another symbol, preceded by a colon.

If the named symbol does not exist it will be created. If 'module' is given, the symbol will be created in the named module, otherwise a dummy module name of "**_NONE_**" is assumed.

:symbol/symbol=value
:module/symbol/symbol=value

These special forms of the ':' command can be used for creating a symbol at a special location in the internal SourceGate symbol table. Giving two symbol named in the 'symbol=value' form of the ':' command causes SourceGate to insert the second named symbol into the symbol table immediately after the first named symbol.

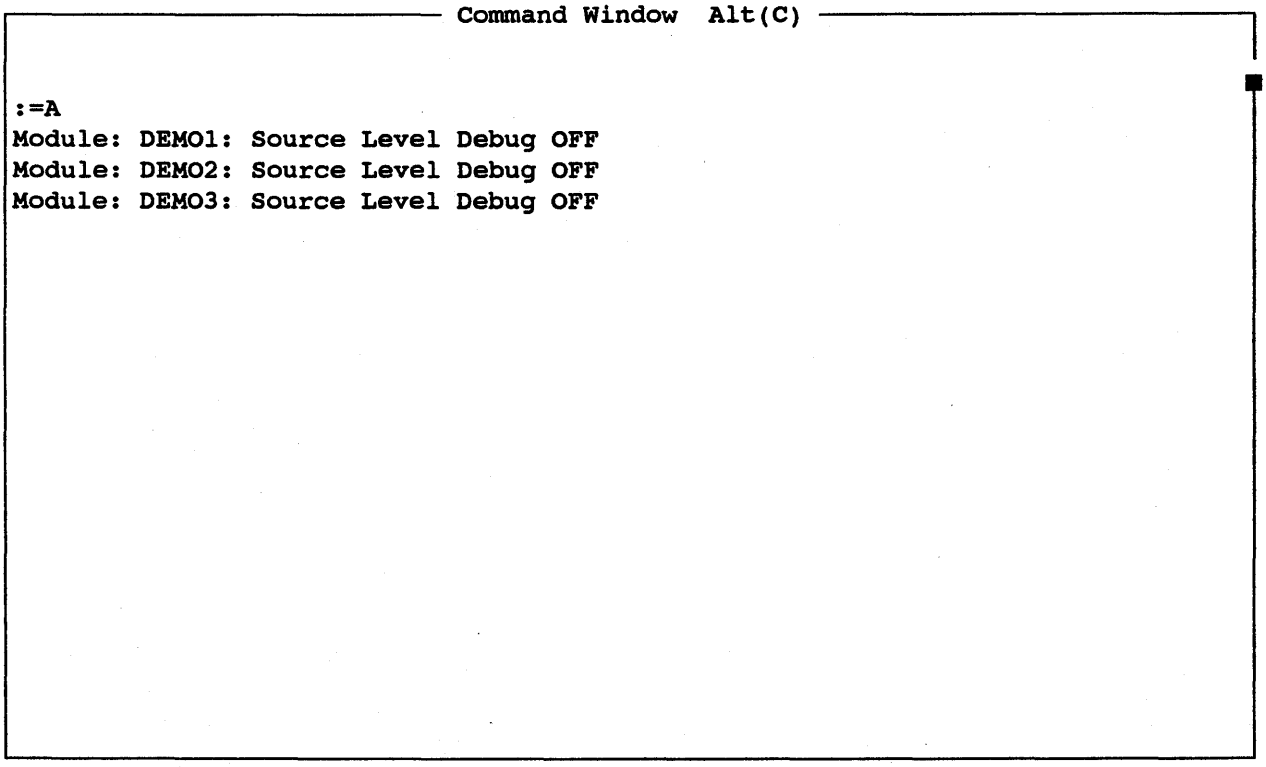
4.3.53 :=<ACDMS> (Perform Symbol Module Function) Command

The following commands are used to enable a particular level of debugging on a module or globally.

```
:=A [module]  Assembly Mode
:=C [module]  Clear Symbol Table
:=D [module]  Display Current Modes
:=M [module]  Mixed Mode
:=S [module]  Source Mode
```

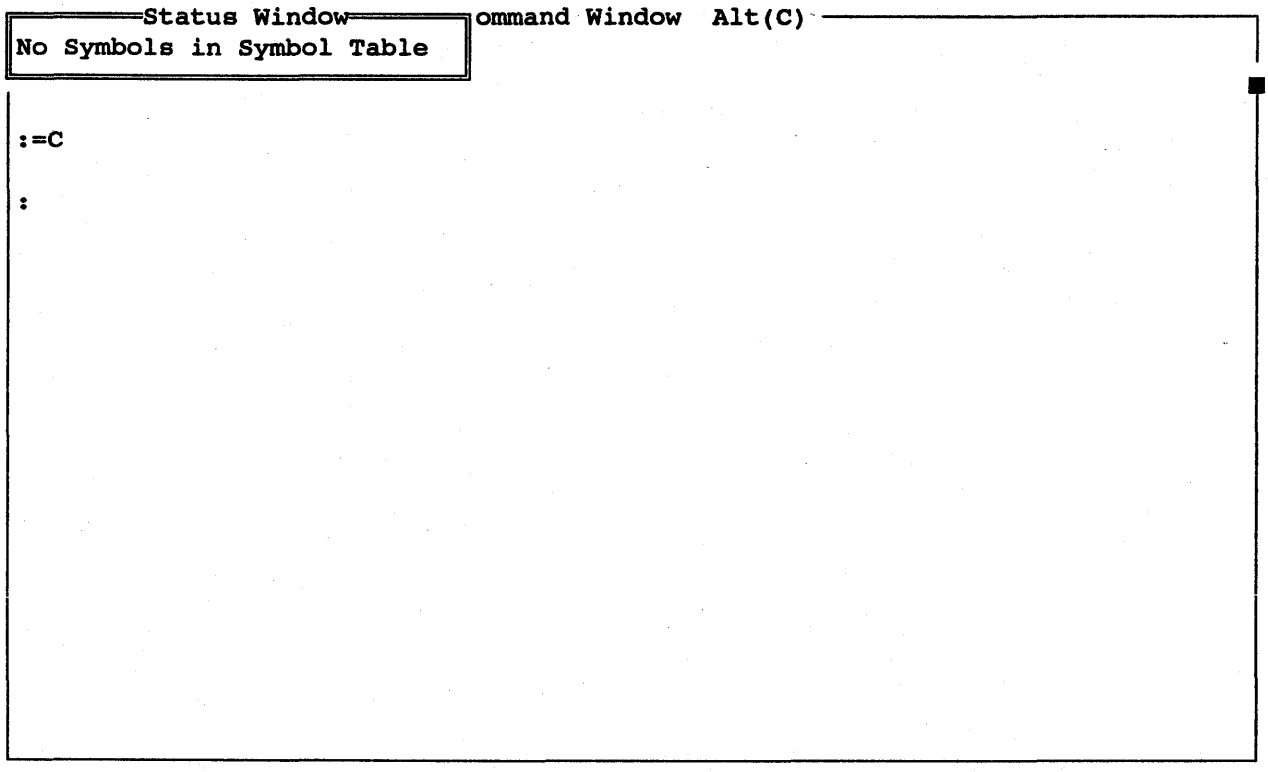
:=A [module] (Assembly Mode) Command

The ':=A' command sets the named module (or all modules if none is named) to Assembly Level Debug mode. This is the default mode for all modules when SourceGate is started.



:=C [module] (Clear Symbol Table) Command

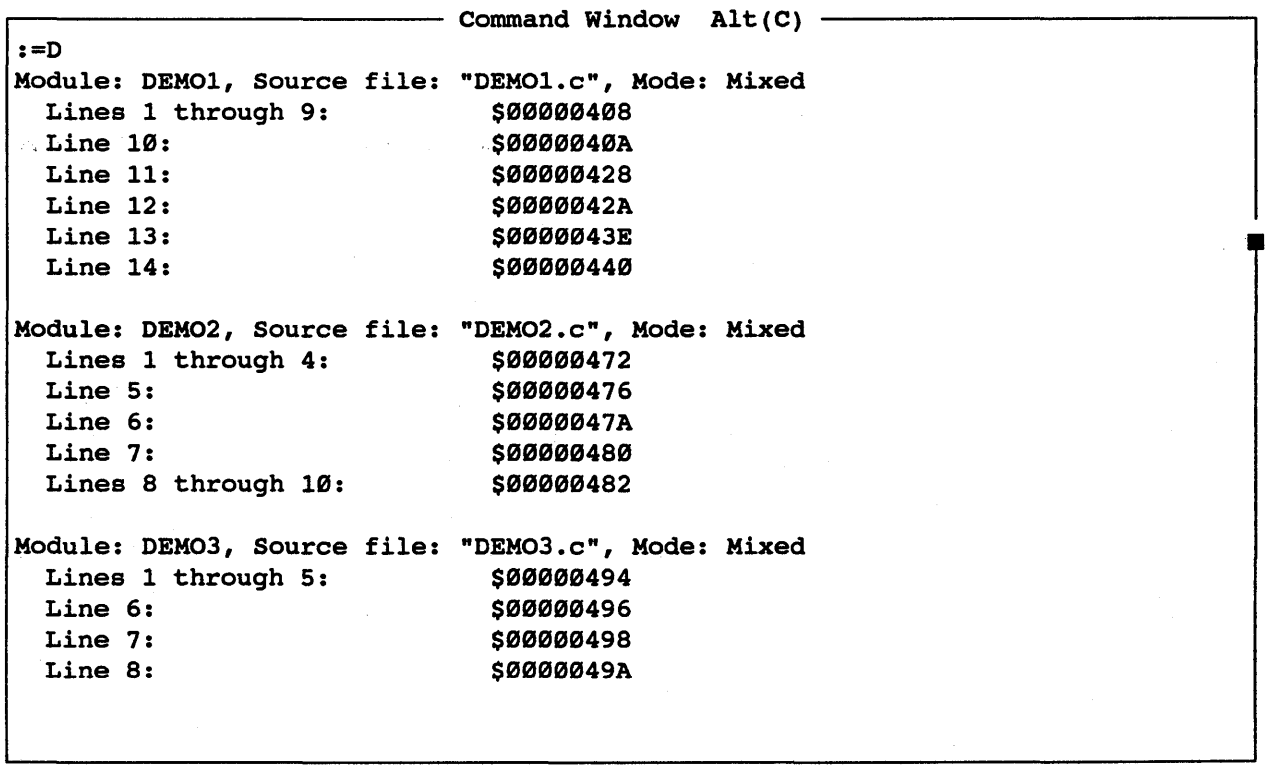
The ':=C' command clears all symbols from the internal symbol table for the named module (or all modules if none is named). The symbol information for the cleared module must be read back into the internal symbol file in order to perform symbolic/source debugging on that module.



:=D [module]

The ':=D' command displays source debug information for the named module (or all modules if none is named).

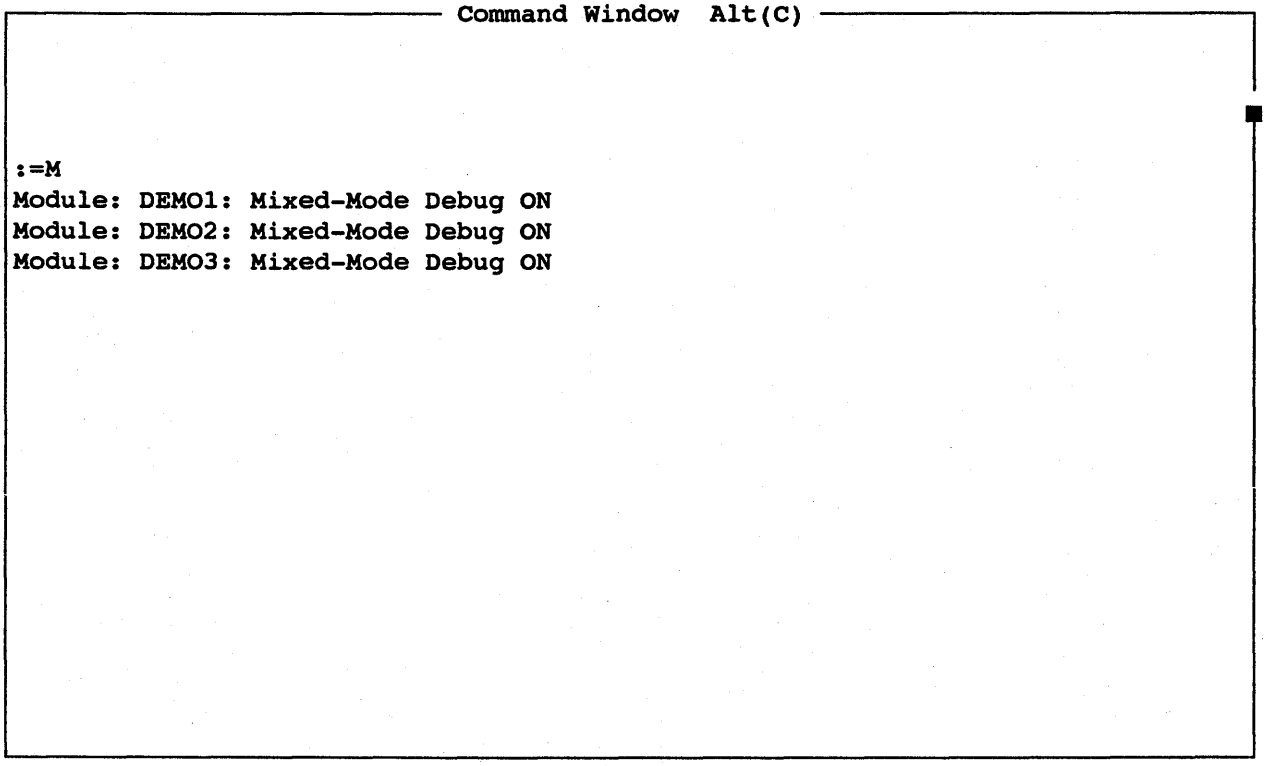
Source debug information consists of the line number map acquired by SourceGate through examination of the module's line number symbols (whose format is set with the SourceGate command line switch '-L'), the module's source code file name, and the current debug mode set for that module (Assembly, Mixed, or Source-only).



:=M [module] (Mixed Mode) Command

The ':=M' command sets the named module (or all modules if none is named) to Mixed Debug mode.

Once a module has been enabled for mixed debug mode, that module's source file will be opened, and the source lines for that module will be inserted at appropriate points into the output of all commands which disassemble code (i.e. the 'L' (List) command and the 'SS' (single-step) command, all register displays, and the Trace window display).



When the ':=M' command is used, SourceGate must be able to locate and successfully open each named module's source code file.

These source code files are assumed to reside in the current directory, and their names are assumed to be the module's name, with the extent of ".C".

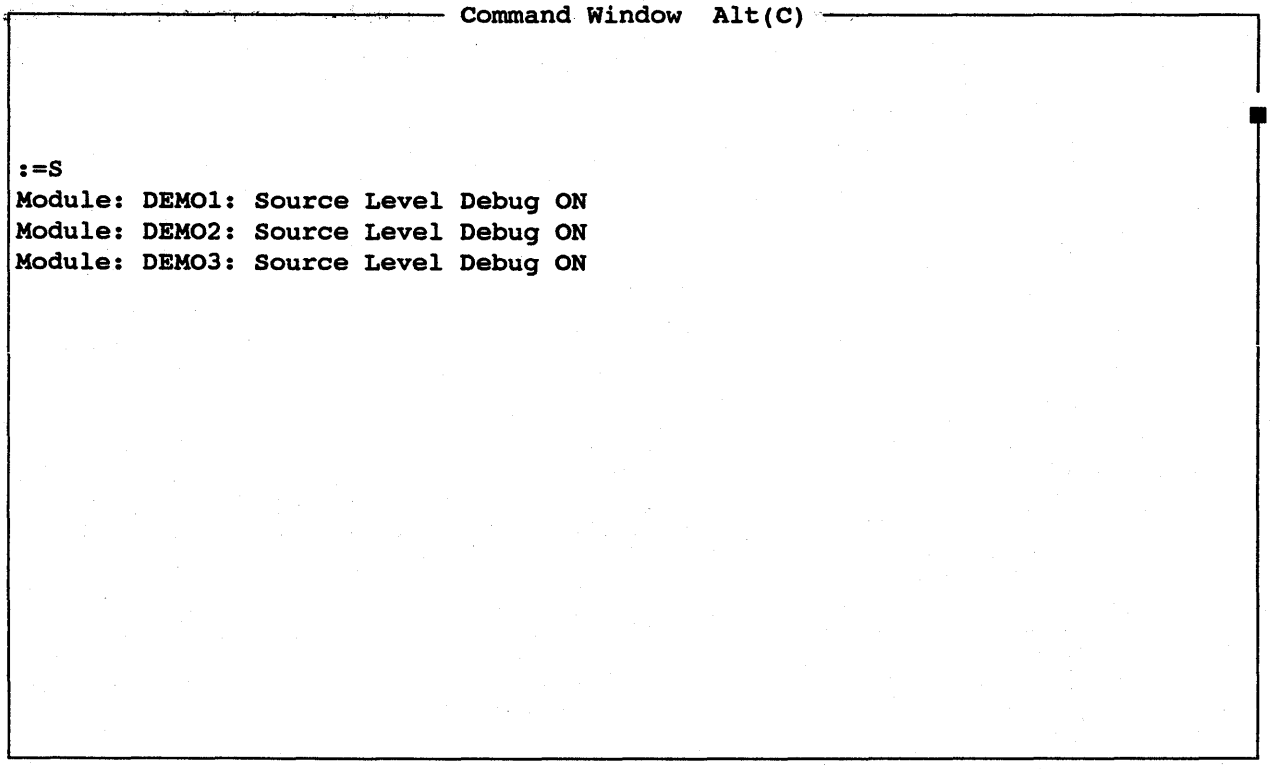
If any of these assumptions are false, SourceGate will prompt for a source code file name, giving the name of the module for which it cannot find the source code file. The full path name of the file, along with the corresponding drive designator, should be entered at this time.

If no source code file is available, entering no file name at the prompt will cause SourceGate to set that module back to Assembly mode debug (which requires no source code file), and continue on to the next module to be set to Mixed debug mode.

:=S [module]

The ':=S' command sets the named module (or all modules if none is named) to Source-only Debug mode.

When source-only debug mode is selected for a particular module, all assembly language displays for that module will be suppressed. (Some assembly may be displayed in the source only mode. This assembly is from the library routines of the compiler and is source code to the emulator since most of these routines are written in assembly language). In addition, that module's source file will be opened and the source lines for that module will be inserted into the output of all commands which disassemble code (i.e. the 'L' (list) command and the 'ss' (single-step) command, all register displays, and the trace window display).



When the ':=S' command is used, SourceGate must be able to locate and successfully open each named module's source code file.

These source code files are assumed to reside in the current directory, and their names are assumed to be the module's name, with the extent of ".C".

If any of these assumptions are false, SourceGate will prompt for a source code file name, giving the name of the module for which it cannot find the source code file. The full path name of the file, along with the corresponding drive designator, should be entered at this time.

If no source code file is available, entering no file name at the prompt will cause SourceGate to set that module back to Assembly mode debug (which requires no source code file), and continue on to the next module to be set to Source debug mode.

4.3.54 <filename (Open a Batch File) Command

```
Command Window  Alt(C)

<test
1:.MAIN + A  :.MAIN + 22
DEMO1:10:              abuffer[i] = tohex(i);     /* Fill ASCII buffer */
00040A DEMO1/LL10:    MOVE.W D2,D3
00040C     MOVEA.L #0000049E:.ABUFFER,A2
000412     MOVE.L D2,-(A7)
000414     JSR.L [00]00046A:.TOHEX
00041A     MOVE.B D0,00(A2,D3.W)
00041E     ADDQ.L #4,A7
000420     ADDQ.L #1,D2
000422

d:.MAIN  :.MAIN + 1F
000400 4E56 0000 48E7 3020 7400 3602 247C 0000  NV..H.0 t.6.$|..
000410 049E 2F02 4EB9 0000 046A 1580 3000 588F  ../.N....j..0.X.

g:.MAIN
```

The '<filename' command is used to temporarily redirect SourceGate user input from the keyboard to the named batch file. Any emulator command may be given in the batch file.

A '<' command given in a batch file causes the current batch file to be closed, and the named batch file opened. Batch file processing does not return to the old batch file once the new one finishes. A '<' by itself on a line causes the current batch file to be closed without opening a new one.

A batch file name is a standard file name, as given above. The default extent for this file is ".BAT".

4.3.55 ! (Terminate And Stay Resident) Command

Entering a '!' from the command line will exit the SourceGate program, leaving it resident in the computer's memory. After exiting SourceGate through the '!' command, control can be returned to SourceGate by typing 'EXIT' at the DOS prompt.

Exiting SourceGate with the '!' will leave the current configuration of the windows, commands, and symbols intact so that upon returning to SourceGate, the system will be just as it was before exiting.

Chapter 5 Tutorial

5.0 Introduction

This tutorial will walk you through initializing the software, setting break points and trigger points, setting up the watch window, emulating and then stopping the emulator to view the trace buffer, and setting up a performance analysis session.

Copy the following file from the demo disk into the HMI directory:

```
HMI>copy a:democopy.bat <cr>
```

At the HMI prompt, run the batchfile.

```
HMI>democopy <cr>
```

This batchfile copies the following demo files from drive a: into the host computer:

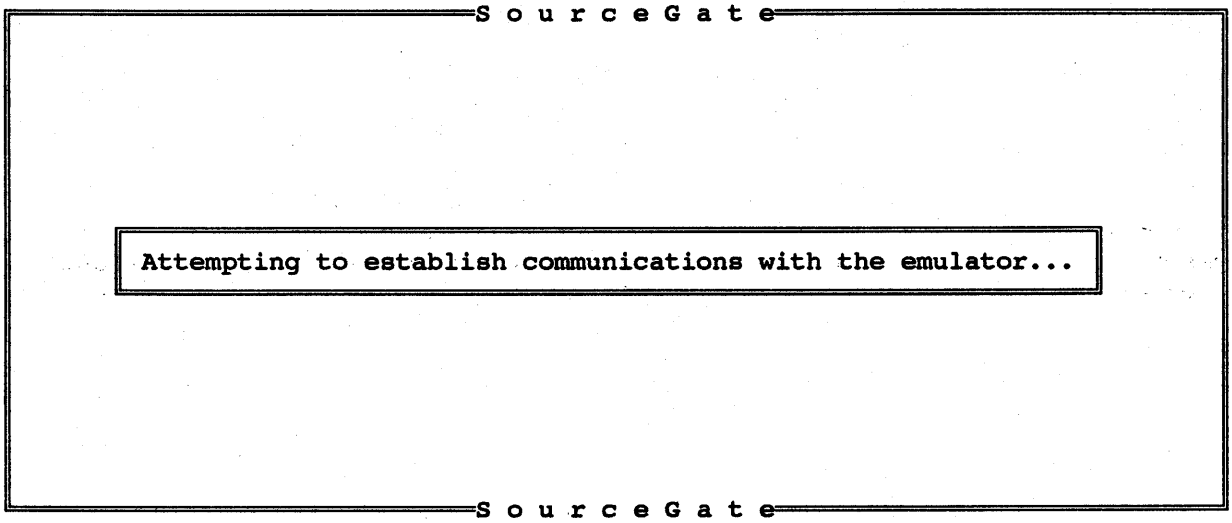
demo1.c	Source code for the demo program
demo2.c	Source code for the demo program
demo3.c	Source code for the demo program
demo.abs	The hex file to be loaded into the emulator
demo.bat	Batch file to setup demo
pac.set	Performance Analyzer setup screen for the demo program

5.1 Entering SourceGate

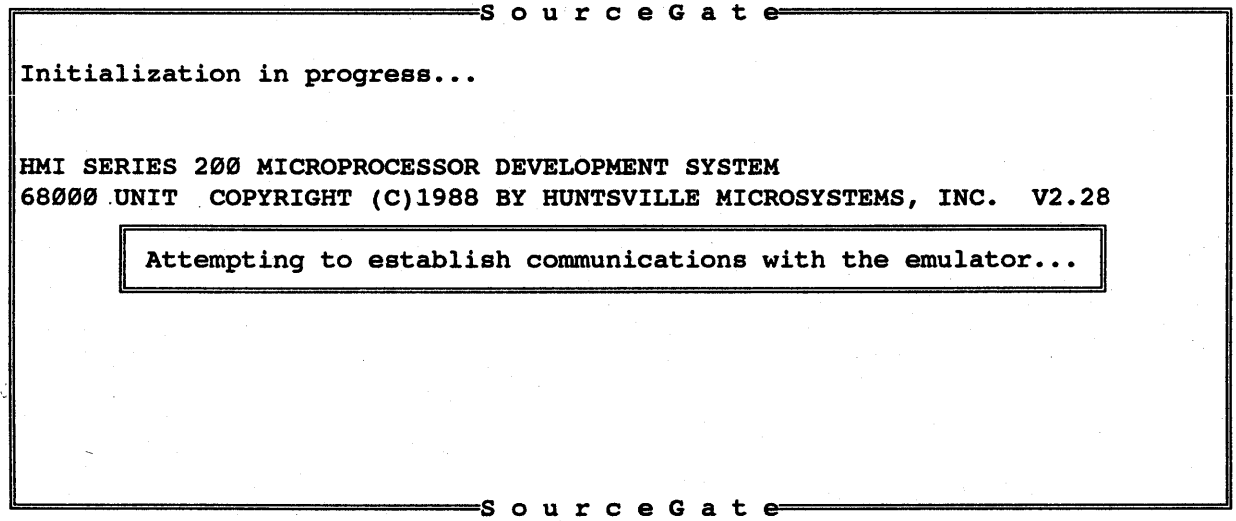
SourceGate establishes communication with the emulator and initiates the software by typing in the following command:

SG68K <cr>	- This enters the software. The default baud rate is set at 9600.

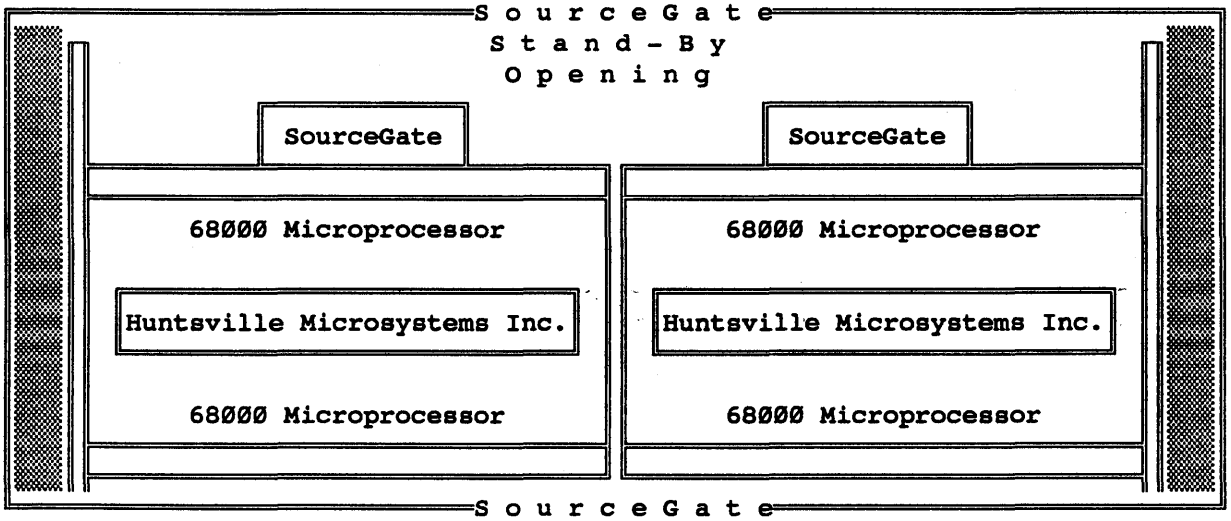
When SourceGate is started, it will display the following screen:



After communications have been established, the emulator will sign-on:



The screen will then display SourceGate "opening":



The messages "Setting Up Emulator" and "Reading Emulator Status" will appear. Once SourceGate is fully loaded, the Register Window and Command Window will appear along with a Status Window:

```
Status Window                    ter Window  Alt(R)
For Help Information: Press F1   Ø
                                 = 2709          NT S I7 NX NG NZ NV CY
D0  = 00004E71 D1  = 55550000 D2 = 00000001 D3 = 0000AAAA  VBR = 00000000
D4  = 00000000 D5  = FFFFFFFF D6 = FFFFFFFF D7 = FFFFFFFF  SFC =         1
A0  = 31FC0000 A1  = 060442C0 A2 = 31FC5555 A3 = 06044EF8  DFC =         0
A4  = 002031FC A5  = AAAA0604 A6 = 4EF80020 A7 = 00001000
```

```
Command Window  Alt(C)
```

You are now ready to begin a SourceGate session.

5.2 Configuring the System

To configure your system for the demonstration, the batch file demo.bat should be read into the emulator.

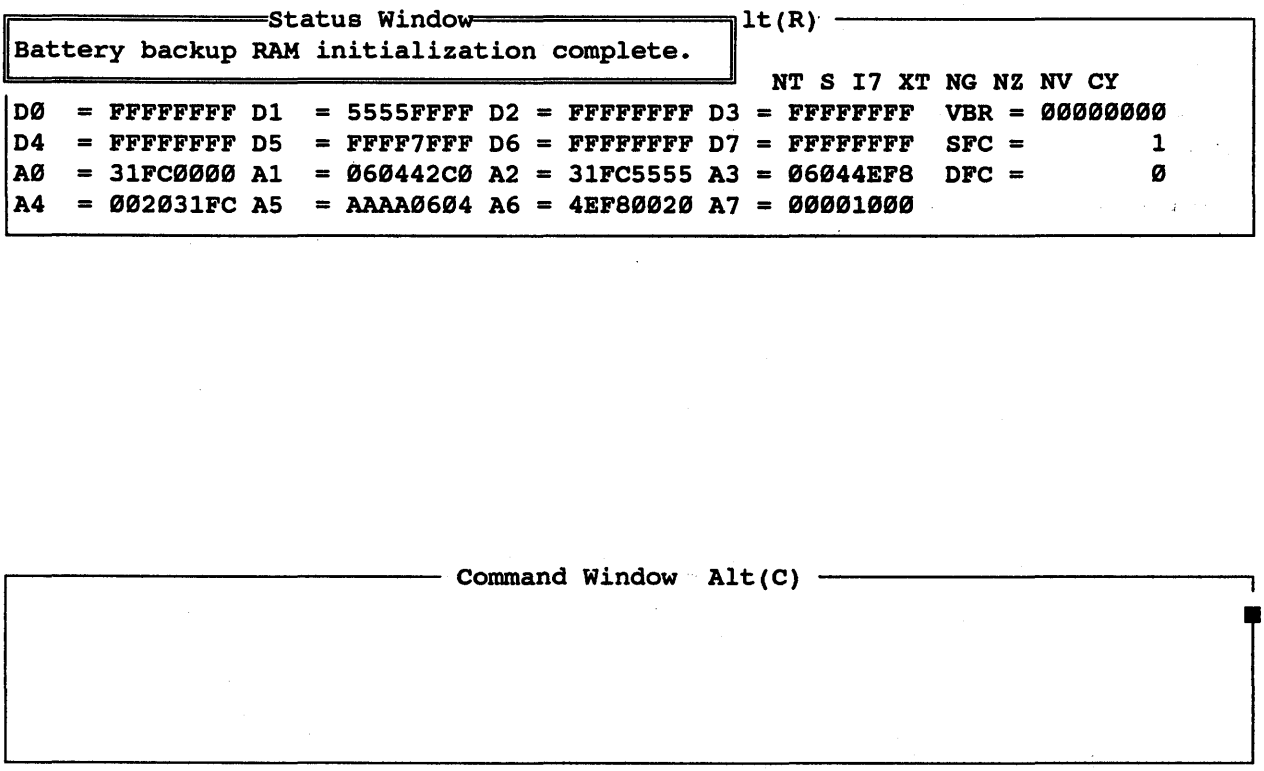
<demo <cr>	- This reads batch file demo.bat into the emulator

The following is executed by Batch File Demo.bat:

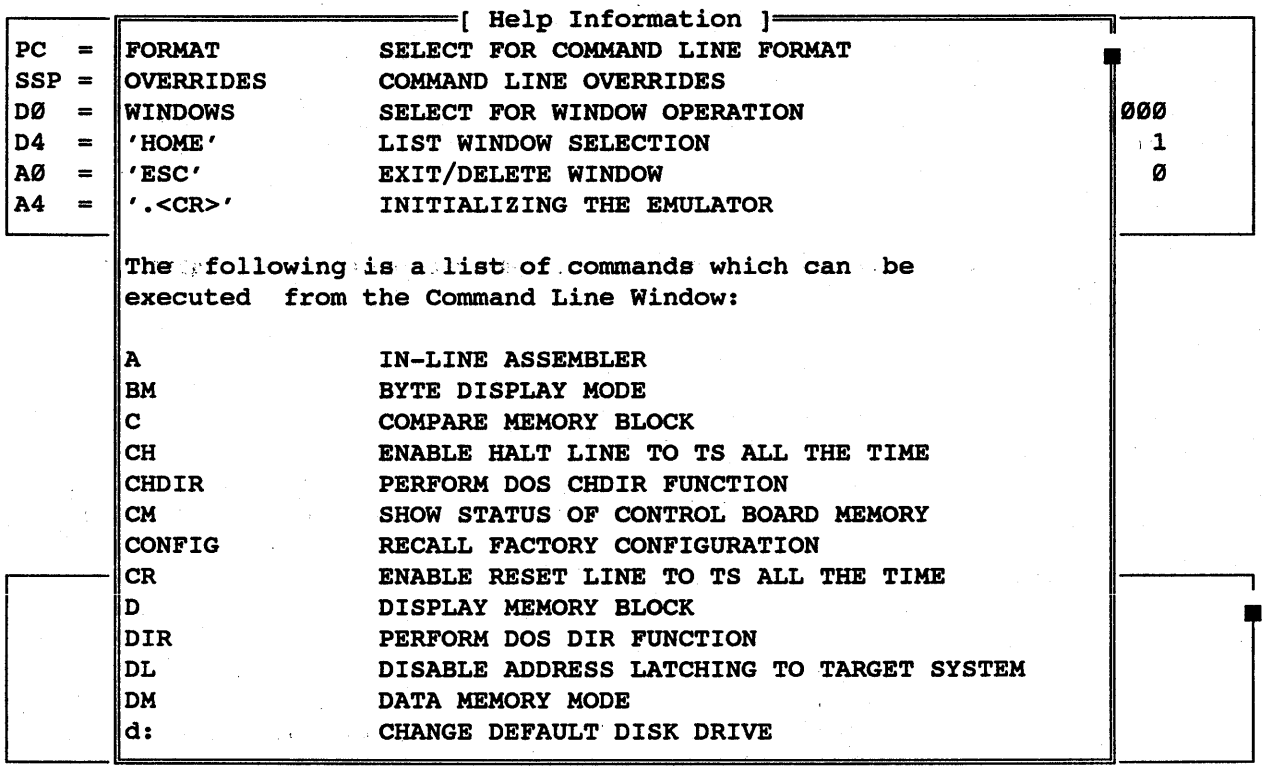
config	- This will return emulator values back to the factory configuration
f0,4 0 f2,2 1000 f6,6,400	- These instructions load the reset and stack pointer vectors into memory
rs	- This resets the emulator with the above reset vectors
rd,demo <cr>	- This reads into memory the sample file "demo.abs" which is a record file containing both symbol and hexadecimal data information

In addition, "demo.bat" will initialize the Performance Analysis Setup window to reflect the examples in the Performance window sections of this tutorial. Once "demo.bat" has finished loading, the cursor will return to the command window which will be cleared.

After the demo.bat file has cleared the command window, two status windows may be displayed in the upper-left-hand corner of the screen. These status windows will give a warning that the PC and/or SP values are invalid. This message was produced when the batch file executed the "config" command with an invalid PC or SP loaded in the processor. Entering "ESC" will delete these status windows and the following screen will be displayed:

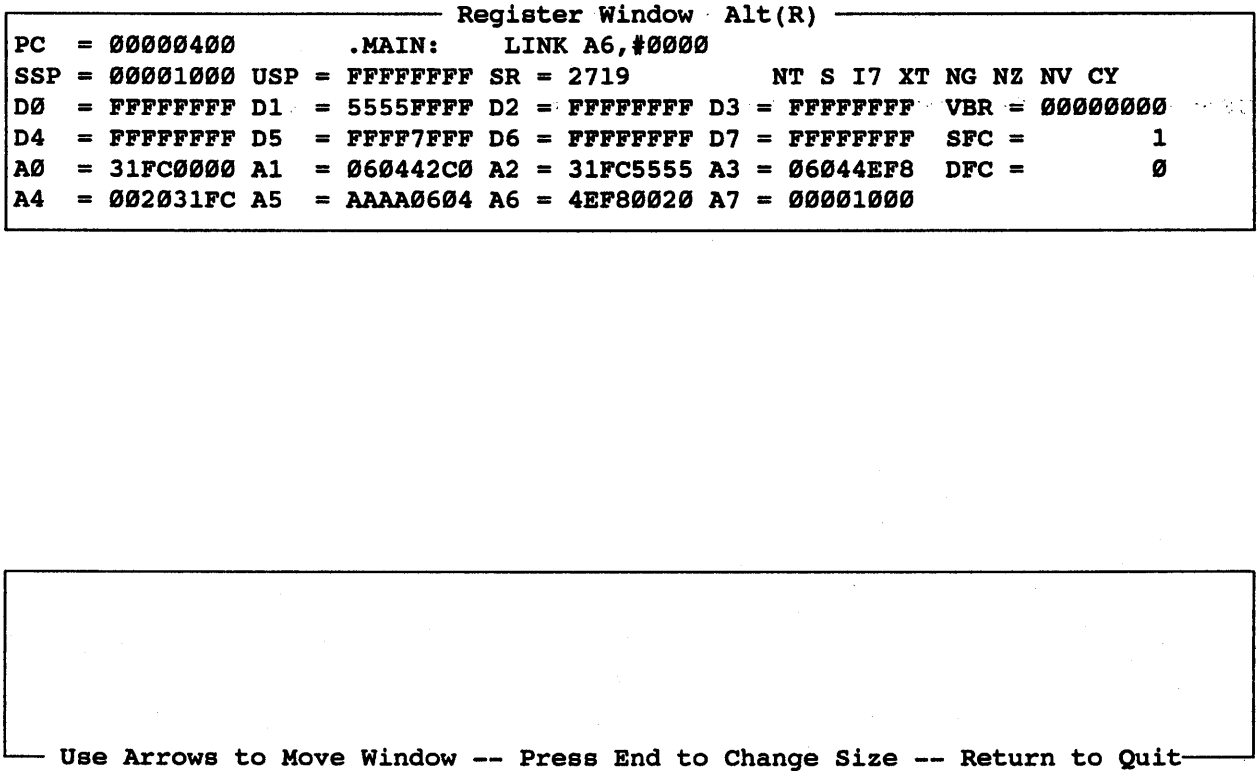

Status Window
Battery backup RAM initialization complete.
lt(R)
NT S I7 XT NG NZ NV CY
D0 = FFFFFFFF D1 = 5555FFFF D2 = FFFFFFFF D3 = FFFFFFFF VBR = 00000000
D4 = FFFFFFFF D5 = FFFF7FFF D6 = FFFFFFFF D7 = FFFFFFFF SFC = 1
A0 = 31FC0000 A1 = 060442C0 A2 = 31FC5555 A3 = 06044EF8 DFC = 0
A4 = 002031FC A5 = AAAA0604 A6 = 4EF80020 A7 = 00001000
Command Window Alt(C)

"F1" - Will display a help window with all available commands for the command window

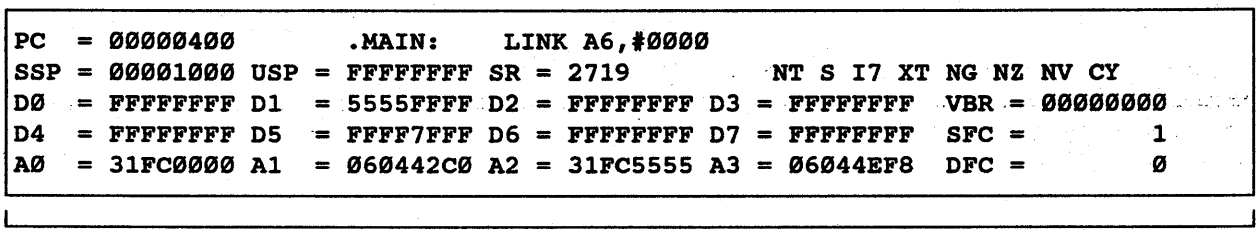


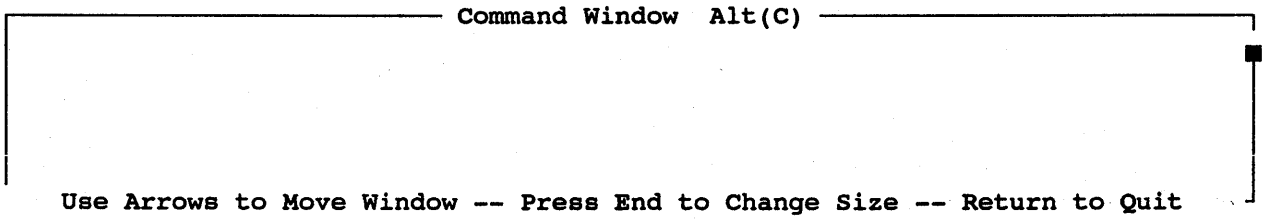
"ESC" - To delete the help window

"F2" - Begin sequence to move window



"UpArrow" - Use up arrow to move command window to top of screen





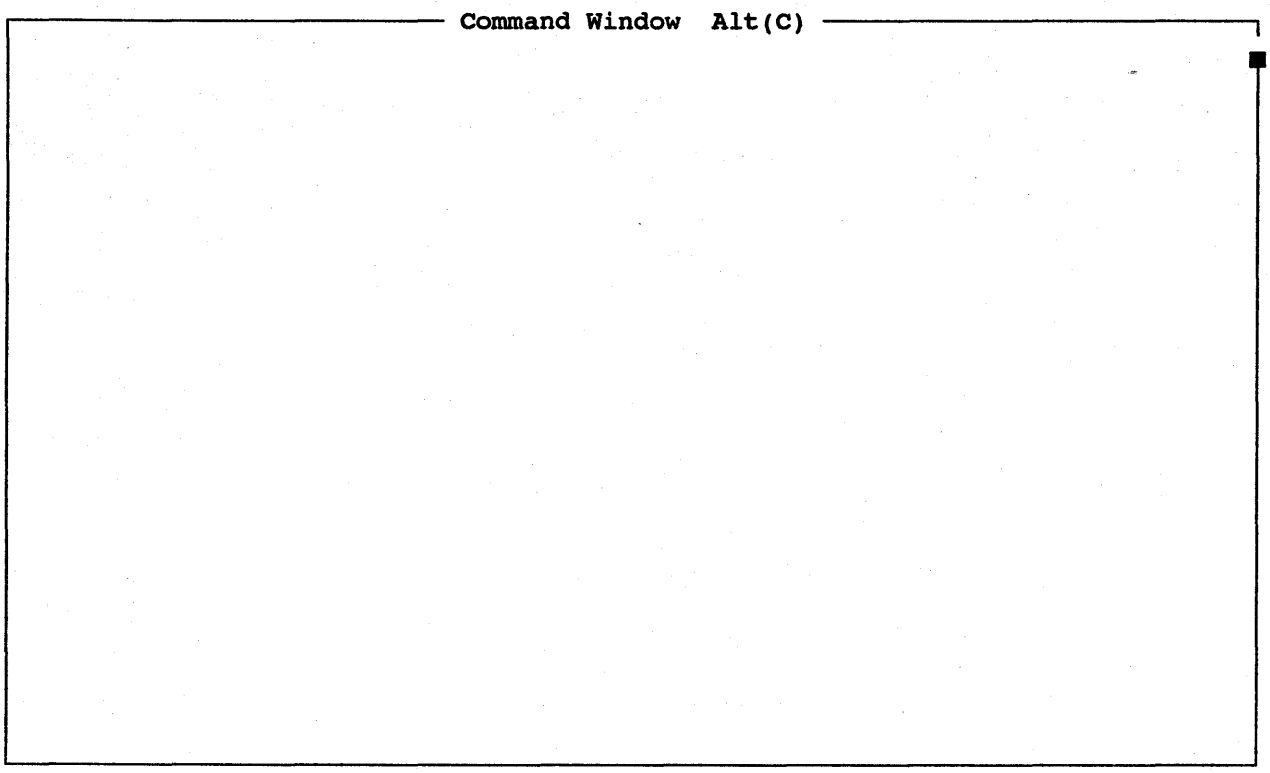
"END" - Begin sequence to size window

```
PC  = 00000400       .MAIN:    LINK A6,#0000
SSP = 00001000 USP = FFFFFFFF SR = 2719          NT S I7 XT NG NZ NV CY
D0  = FFFFFFFF D1  = 5555FFFF D2 = FFFFFFFF D3 = FFFFFFFF  VBR = 00000000
D4  = FFFFFFFF D5  = FFFF7FFF D6 = FFFFFFFF D7 = FFFFFFFF  SFC =        1
A0  = 31FC0000 A1  = 060442C0 A2 = 31FC5555 A3 = 06044EF8  DFC =        0
```

```
Command Window  Alt(C)

   Use Arrows to Change Size -- Press End to Move Window -- Return to Quit
```

"DnArrow" - Use down arrow to enlarge command window to full screen size

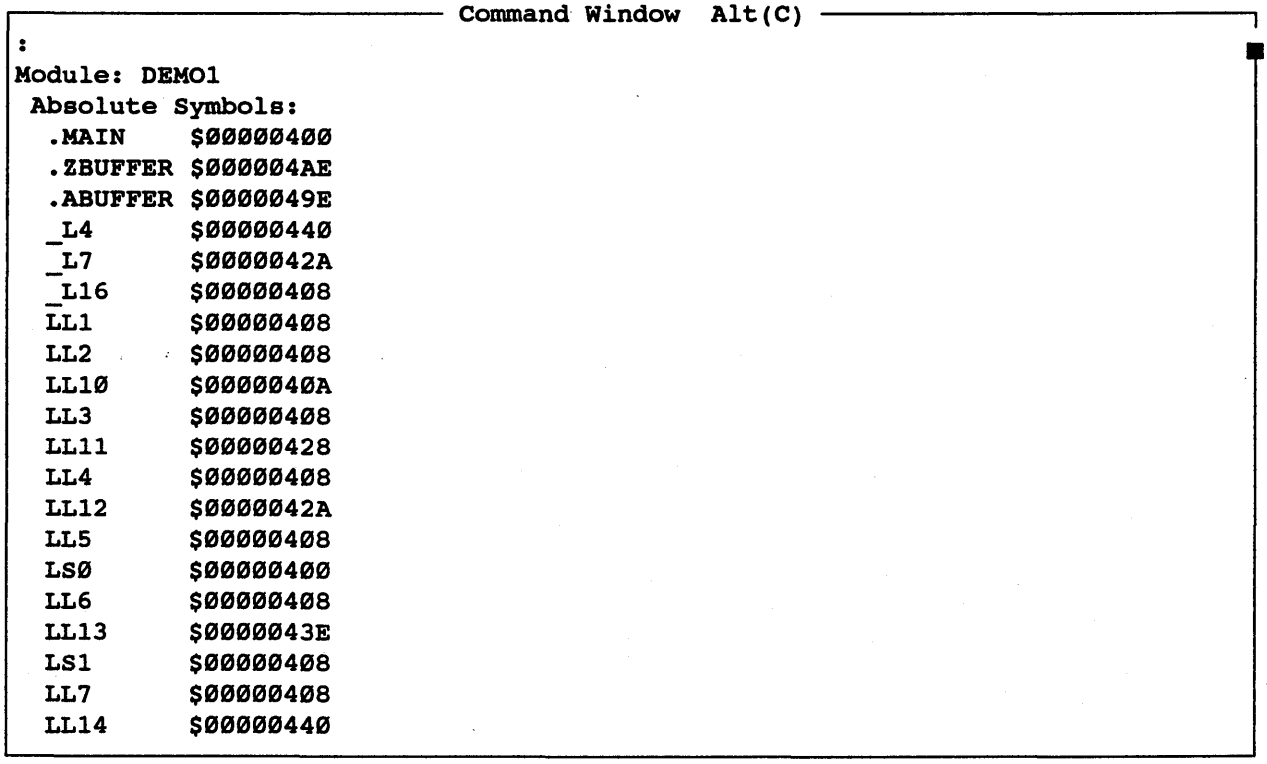
<cr> - After the command window is moved and sized, pressing <cr> will save this new window



"F4" - To zoom between the old command window and the newly sized command window

"F4" - Again to leave large command window on screen

: <cr> - To display current symbols in the symbol table. Use pause key to stop the scrolling and <cr> to resume the scrolling

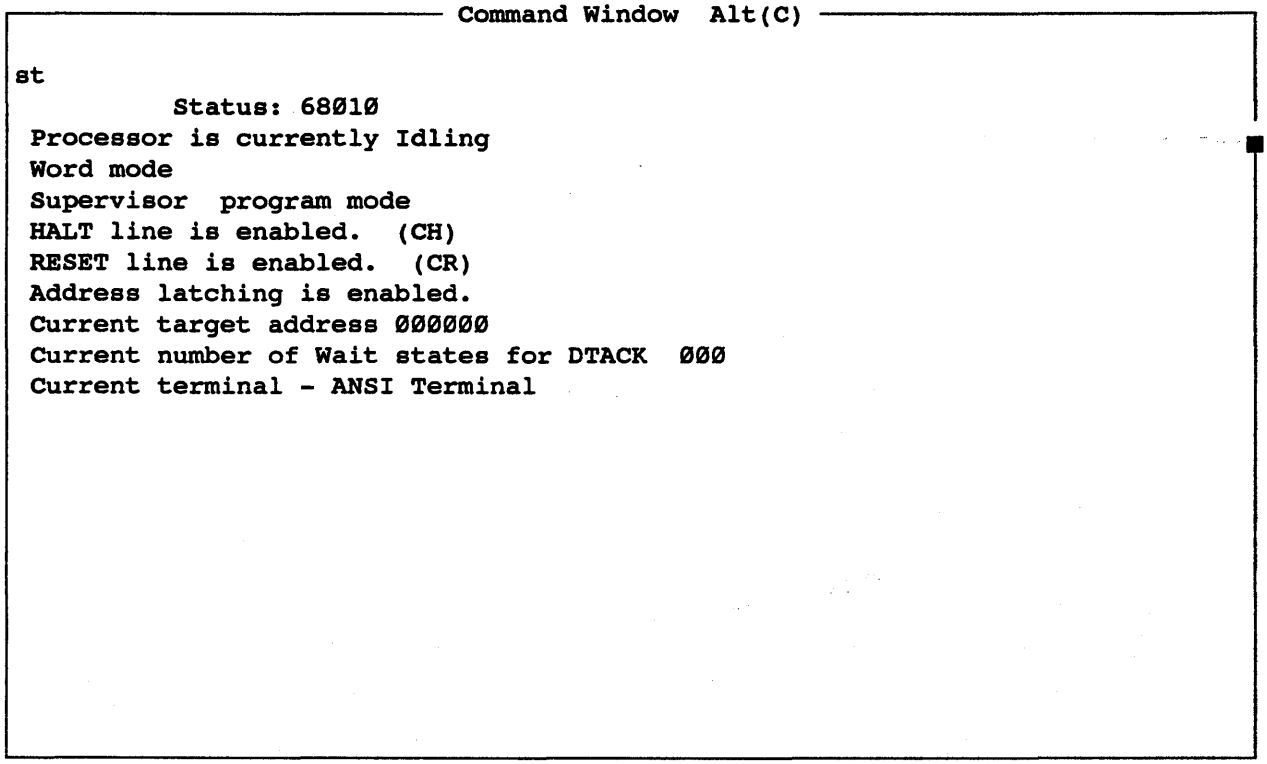


:test1=7000 <cr> - To add a symbol to the symbol table

: <cr> - To view the newly added symbol

```
Command Window  Alt(C)
:
Module: **_None_**
 Absolute Symbols:
  test1     $00007000

Module: DEMO1
 Absolute Symbols:
  .MAIN     $00000400
  .ZBUFFER  $000004AE
  .ABUFFER  $0000049E
  _L4       $00000440
  _L7       $0000042A
  _L16      $00000408
  LL1       $00000408
  LL2       $00000408
  LL10      $0000040A
  LL3       $00000408
  LL11      $00000428
  LL4       $00000408
  LL12      $0000042A
  LL5       $00000408
  LS0       $00000400
  LL6       $00000408
```

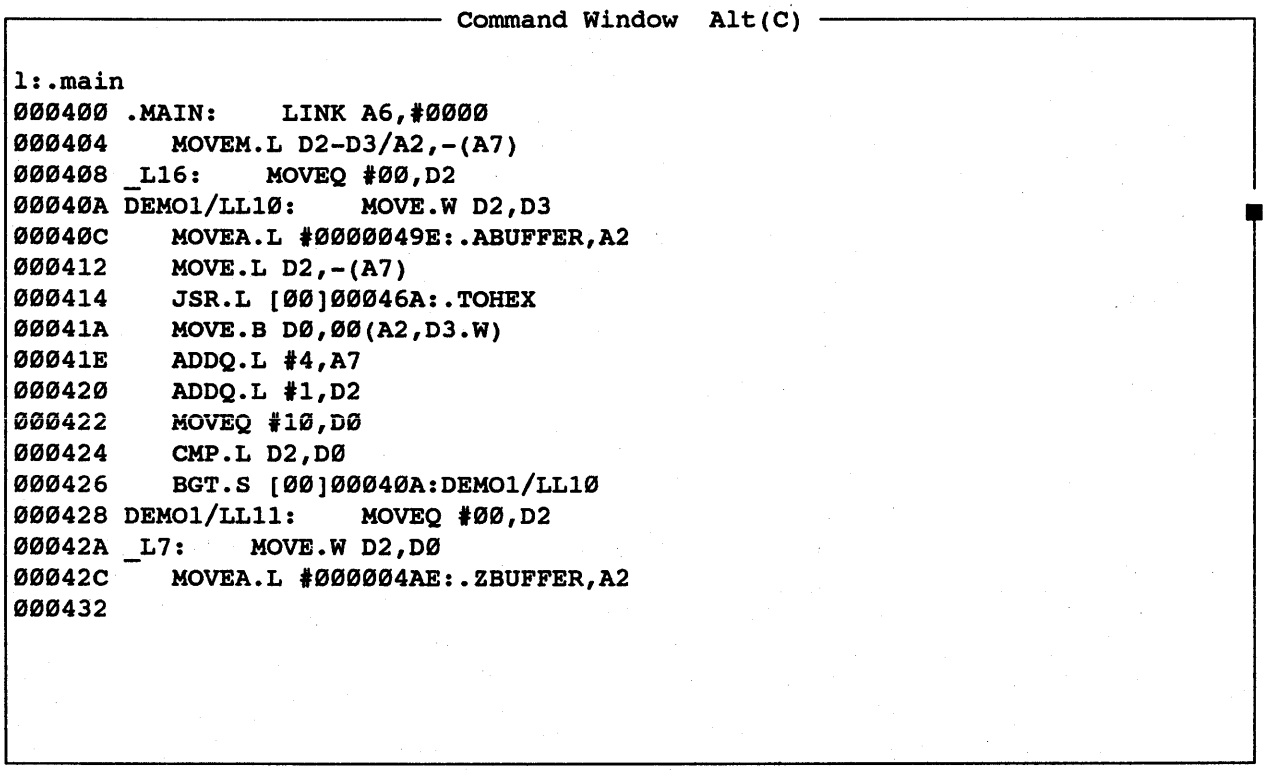
st <cr> - Show processor status



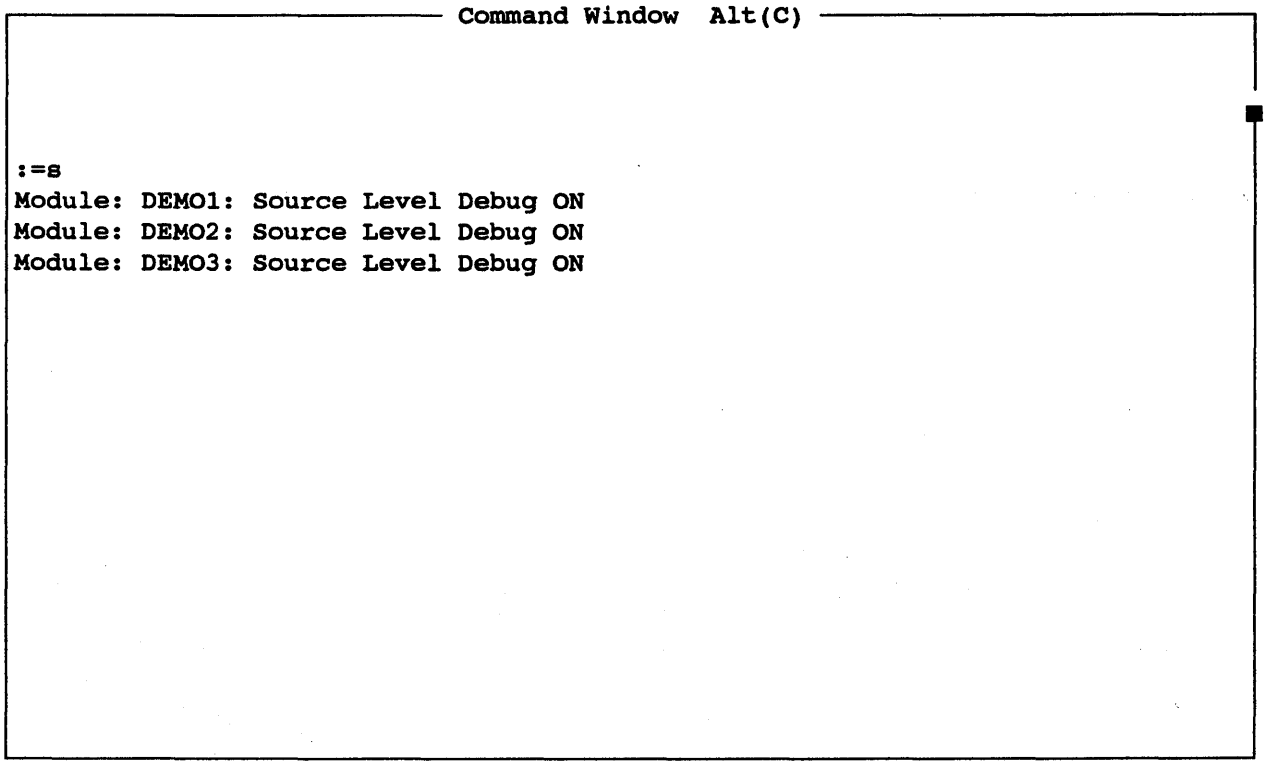
5.2.1 Viewing Source and Assembly

Code can be viewed in both source and assembly, assembly only, or source only. The default is assembly (:=a), to change to both (or mixed mode) type in :=m, and to change to source only, type in :=s.

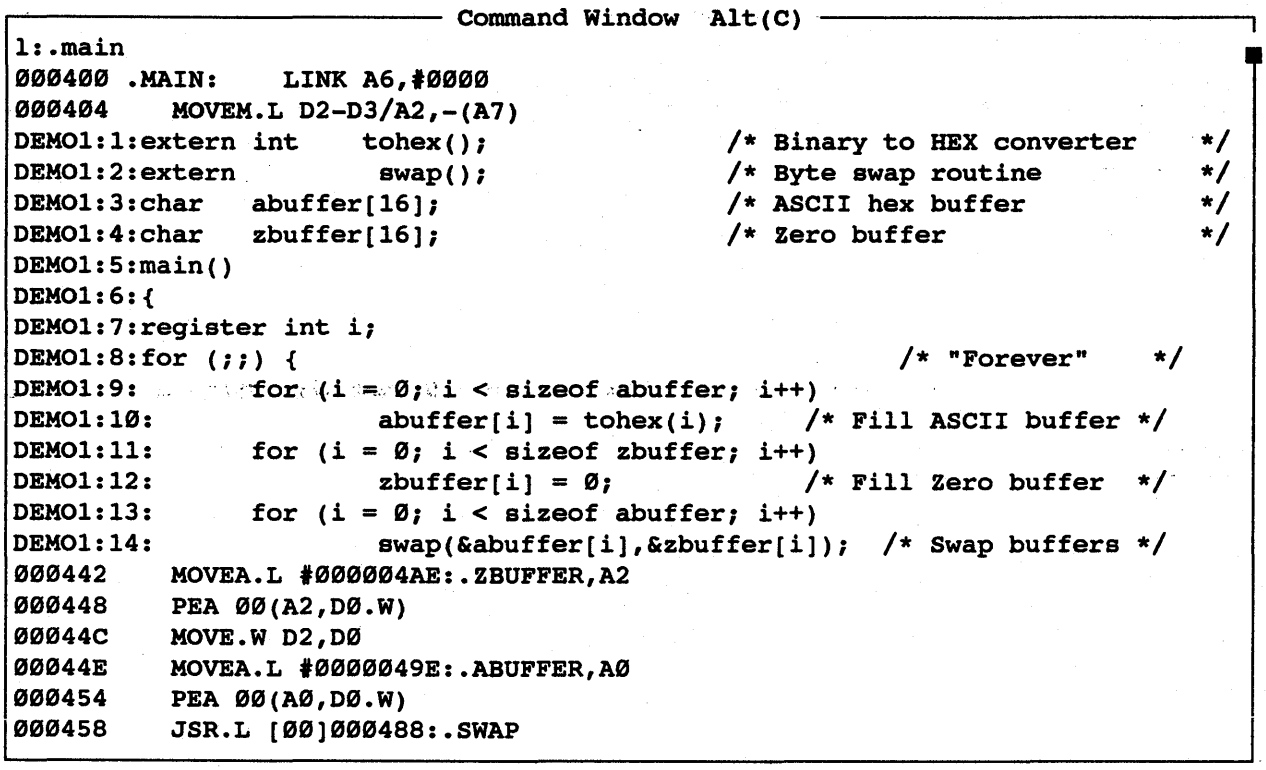
1:.main <cr> - List the current code at address 400 (.main). It will be the assembly listing - as this is the default



:=s <cr> - This allows the source listing to be displayed

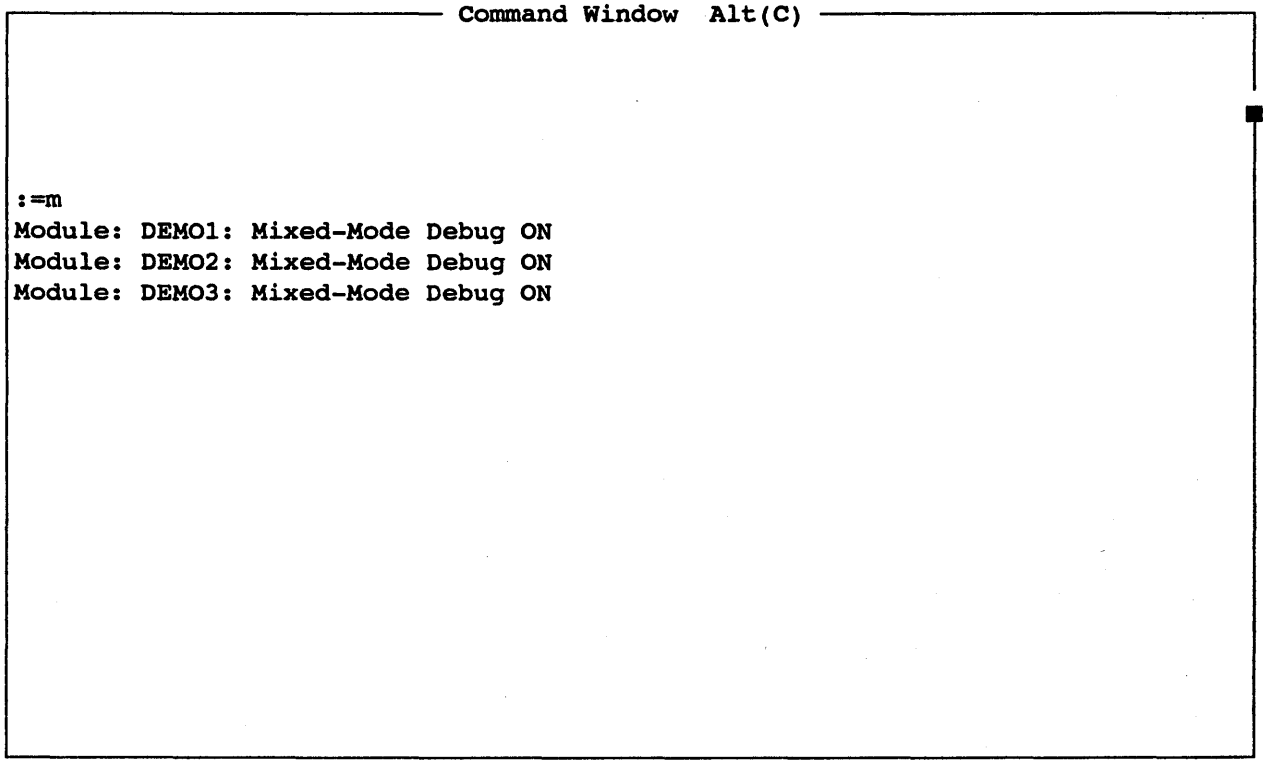


1:.main <cr> - List the current code showing source only. (See Note below)



Note: Some assembly may be displayed in this mode. This assembly is from the library routines of the compiler and is source code to the emulator since most of these routines are written in assembly language.

:=m <cr> - This allows the source and assembly listing to be displayed at the same time



1:.main <cr> - List the current code again showing both the source code and the assembly language produced from the source code

```
Command Window  Alt(C)
1:.main
000400 .MAIN:    LINK A6,#0000
000404    MOVEM.L D2-D3/A2,-(A7)
DEMO1:1:extern int    tohex();              /* Binary to HEX converter    */
DEMO1:2:extern         swap();              /* Byte swap routine          */
DEMO1:3:char   abuffer[16];                 /* ASCII hex buffer           */
DEMO1:4:char   zbuffer[16];                 /* Zero buffer                */
DEMO1:5:main()
DEMO1:6:{
DEMO1:7:register int i;
DEMO1:8:for (;;) {                                        /* "Forever"    */
DEMO1:9:       for (i = 0; i < sizeof abuffer; i++)
000408 _L16:    MOVEQ #00,D2
DEMO1:10:               abuffer[i] = tohex(i);     /* Fill ASCII buffer */
00040A DEMO1/LL10:    MOVE.W D2,D3
00040C    MOVEA.L #0000049E:.ABUFFER,A2
000412    MOVE.L D2,-(A7)
000414    JSR.L [00]00046A:.TOHEX
00041A    MOVE.B D0,00(A2,D3.W)
00041E    ADDQ.L #4,A7
000420    ADDQ.L #1,D2
000422    MOVEQ #10,D0
000424    CMP.L D2,D0
```

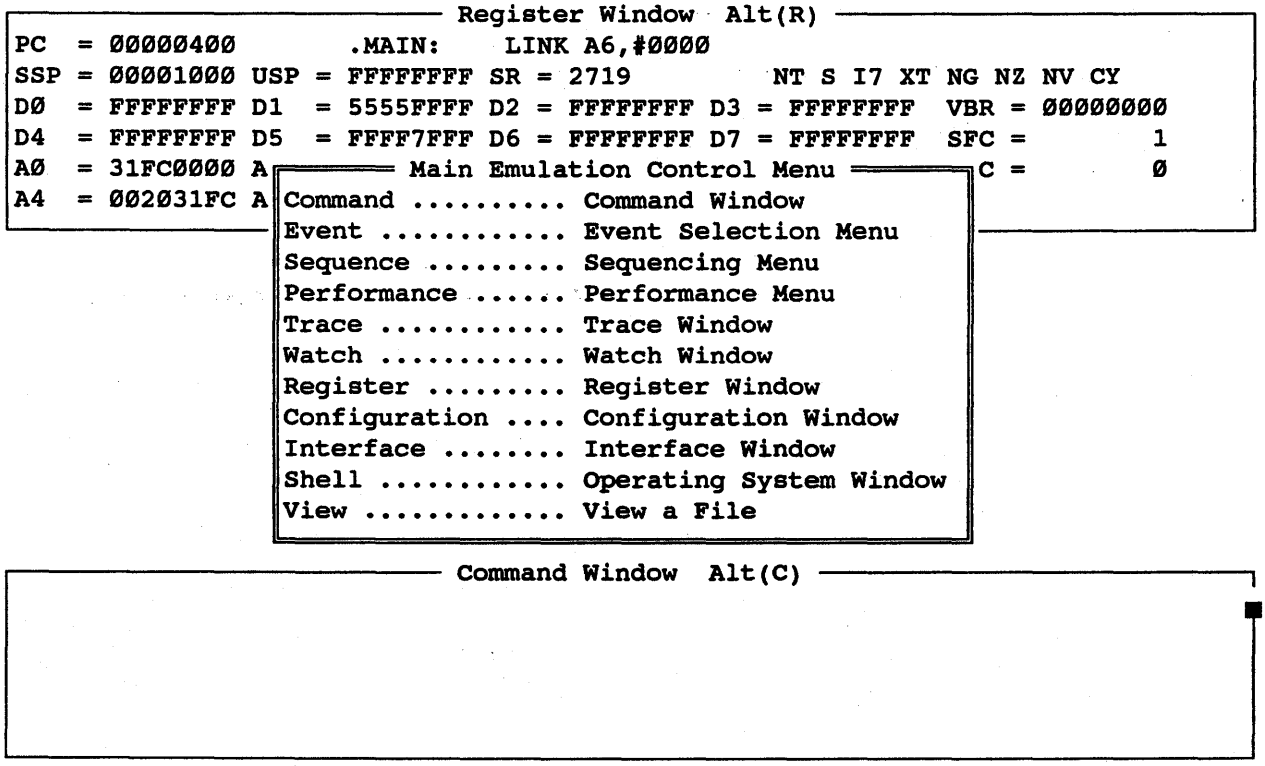
"F4" - Zoom back to original sized Command window

5.3 SourceGate Windows

The basic keys to control the windows in SourceGate are the following:

"HOME"	- Main Emulation Control Window
"ESC"	- To delete the current active window
"F1"	- Help Window will display help available information at the current level
"F2"	- To size and move the current active window on the screen.
"F4"	- To zoom and view two versions of a window
"ALT F1"	- Help window that lists all the windows that are available and the ALT keys to select these windows

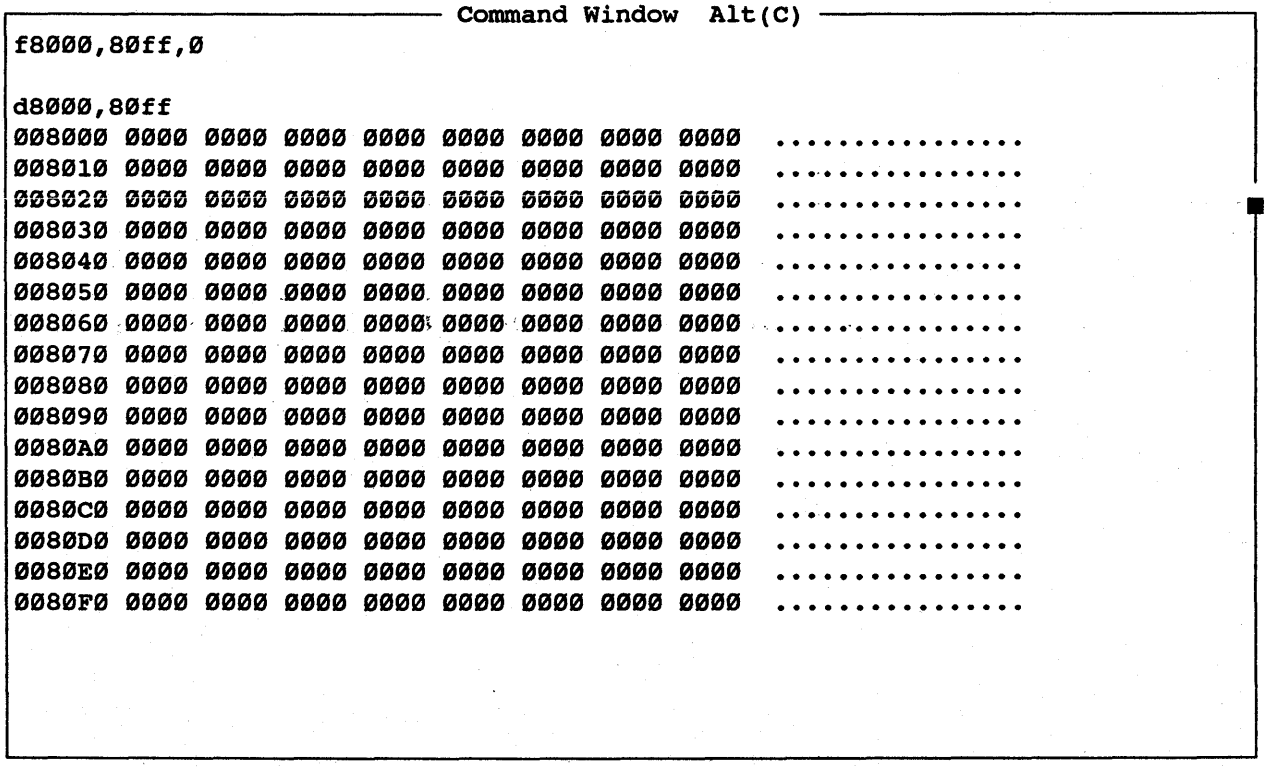
To view the Main Emulation Control Window type in the "HOME" key and the following list will be displayed:



5.3.1 The Command Window

The command window is where the majority of emulator commands are executed. Examples of some of the commands that are available in the command menu are shown.

"HOME"	- To view available windows
Arrow down to where Command is highlighted	- Arrow to the Command window
Command <cr>	- Select the Command window by pressing <cr>
"F4"	- For full sized command window
f8000,80FF,0 <cr>	- The fill command allows you to fill a memory block with a value
d8000,80FF <cr>	- Display memory block starting at address 8000 and ending address 80FFH which now contains all zeros



Use up arrow key to move highlighted bar to d8000,80FF	- Use the up arrow key to select commands that are to be executed again without having to re-type them again
<cr>	- Command selected with highlighted command bar is repeated
xpc=400	- Change PC value to 400H
ssx <cr>	- Single step through your code
Type in a <CR>	- To abort this command

Command Window Alt(C)

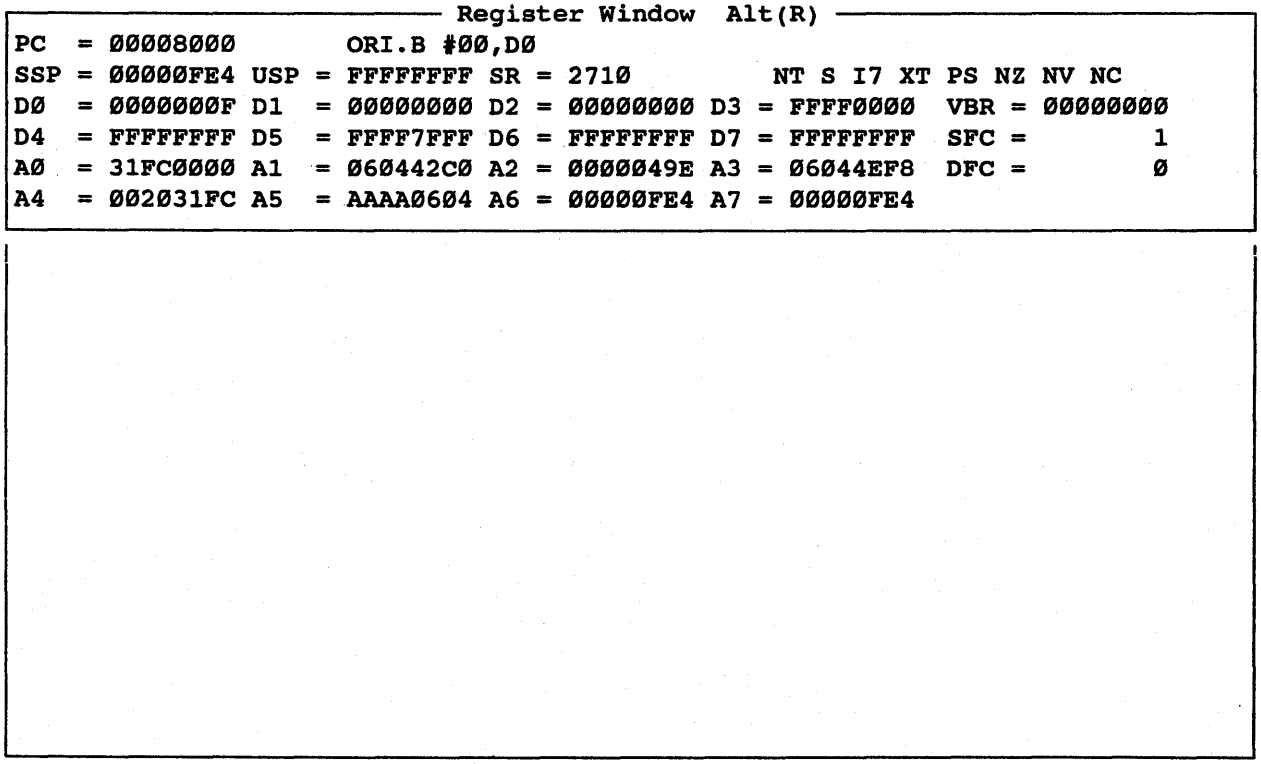
```
xpc=400

ssx
00000400 .MAIN:    LINK A6,#0000
00000404    MOVEM.L D2-D3/A2,-(A7)
DEMO1:1:extern int    tohex();              /* Binary to HEX converter    */
DEMO1:2:extern         swap();              /* Byte swap routine          */
DEMO1:3:char   abuffer[16];                 /* ASCII hex buffer           */
DEMO1:4:char   zbuffer[16];                 /* Zero buffer                */
DEMO1:5:main()
DEMO1:6:{
DEMO1:7:register int i;
DEMO1:8:for (;;) {                                        /* "Forever"    */
DEMO1:9:       for (i = 0; i < sizeof abuffer; i++)
00000408 _L16:    MOVEQ #00,D2
DEMO1:10:              abuffer[i] = tohex(i);     /* Fill ASCII buffer */
0000040A DEMO1/LL10:    MOVE.W D2,D3
0000040C    MOVEA.L #0000049E:.ABUFFER,A2
00000412    MOVE.L D2,-(A7)
00000414    JSR.L [00]00046A:.TOHEX
0000046A .TOHEX:    LINK A6,#0000
0000046E    MOVE.L 0008(A6),D1
DEMO2:1:int tohex(c)               /* Convert low digit of c to ASCII hex */
```

! <cr>	- Allows exit to DOS operating system DOS commands can be executed without leaving SourceGate
exit <cr>	- To exit DOS and get back to SourceGate
CNTRL-HOME	- Clear command window

5.3.2 The Register Window

The register window displays all the current register values. The register values can be modified by placing the cursor on the desired field and entering a new value.

"HOME"	- To view available windows
Arrow down to where Register is highlighted	- Arrow to the Register Window
Register Window <cr>	- Select the Register window by pressing return
Arrow over to PC	- Arrow to the PC option
8000 <cr>	- Change the value of the program counter to 8000H



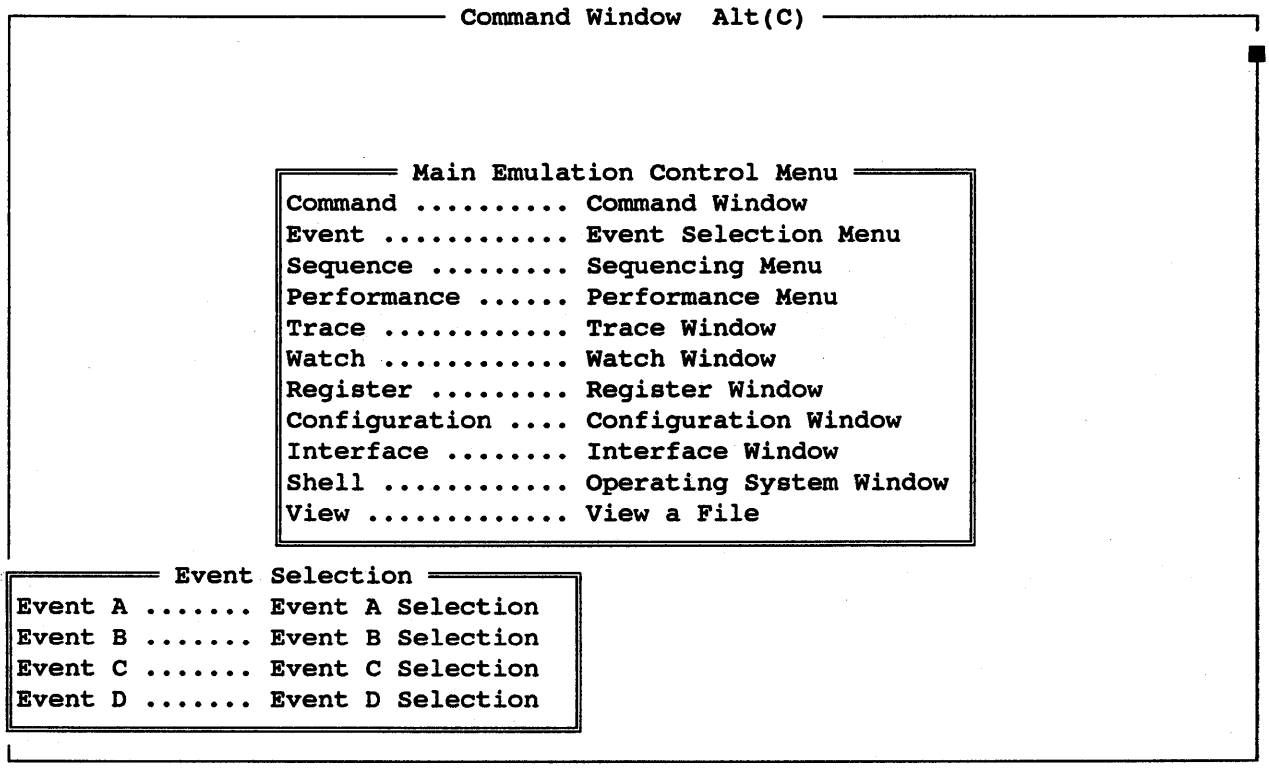
ESC	- To delete this window

5.3.3 The Event Windows

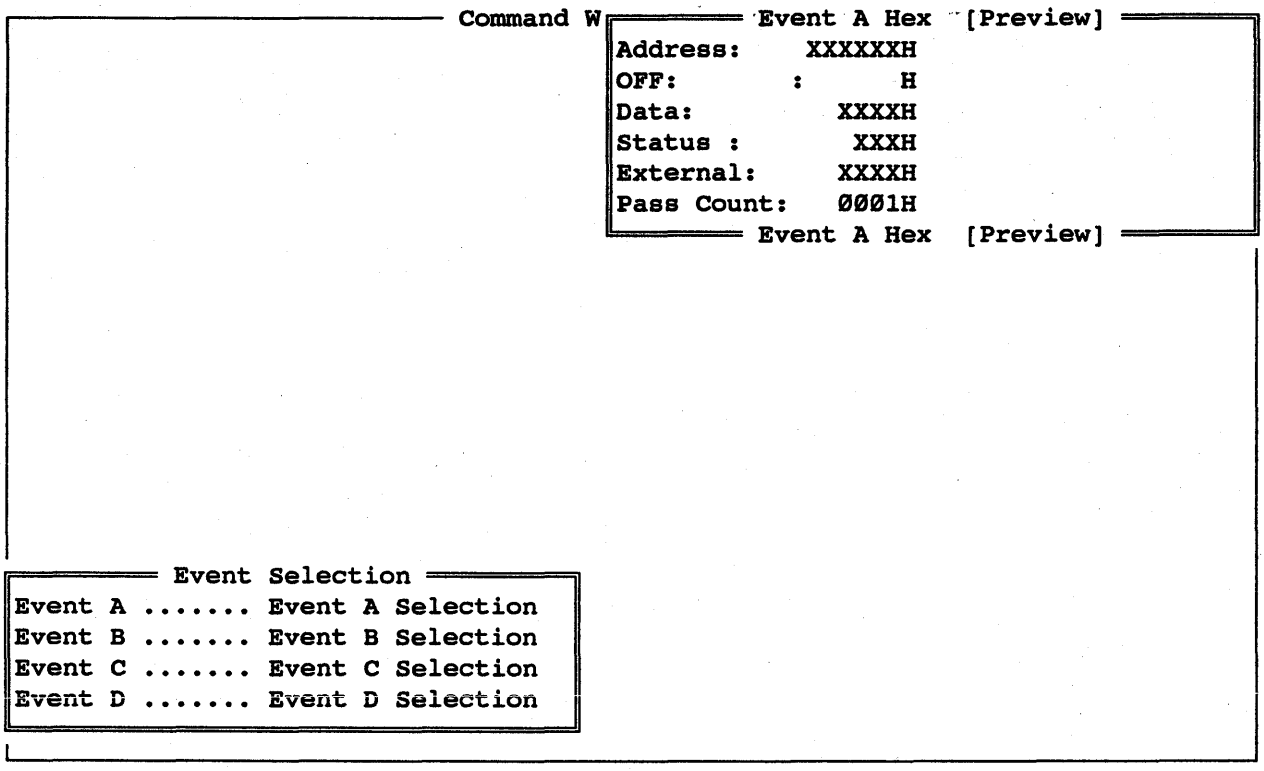
The event window allows you to define the the 4 events (A,B,C,D) as HEX, Binary, Status and external signals. The four events are broken down into smaller, compact windows with specific functions in SourceGate.

"HOME" - To view available windows

Arrow down to where Event is highlighted - Arrow to the Event window

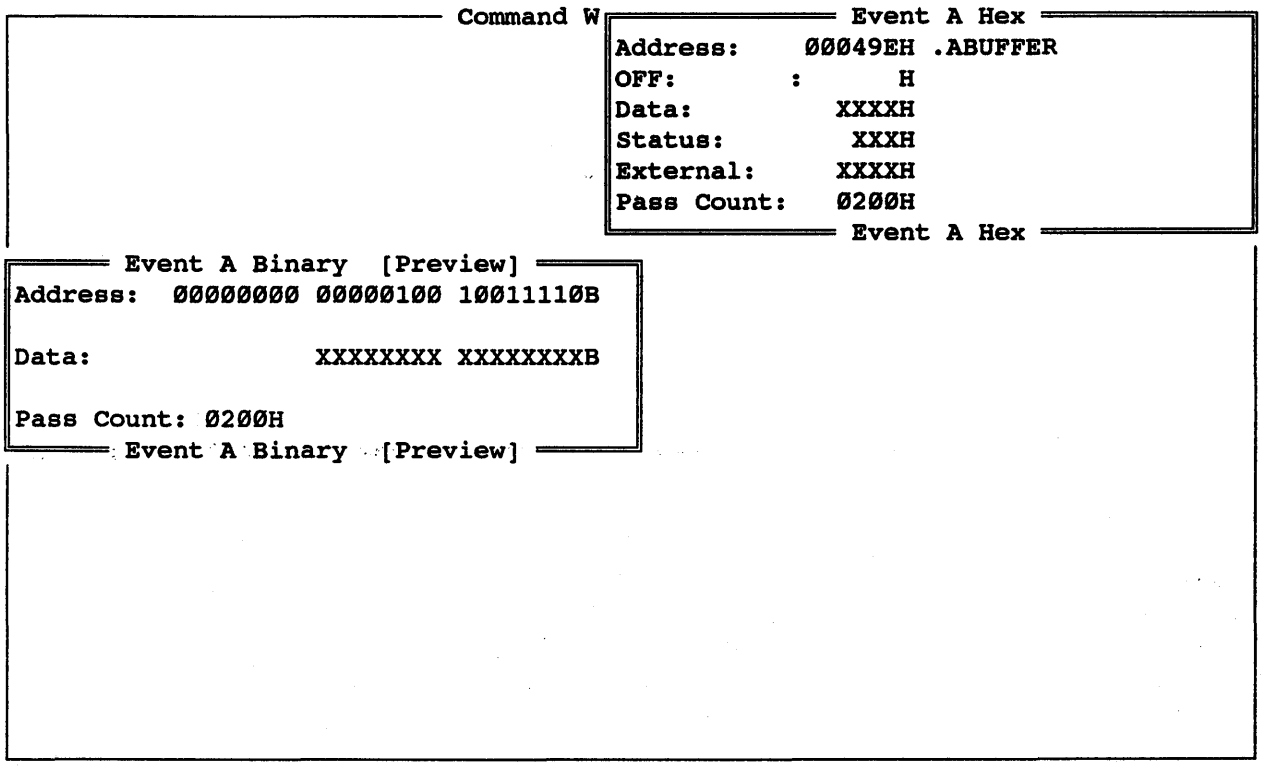


Event <cr>	- Select the Event Window by pressing return to enter
Arrow to where A is highlighted	- Arrow to Event A in the Event Selection Window



Event A <cr>	- Select Event A
Address 00049E	- Set address to be 49EH
Arrow down to Pass Count	- Arrow to Pass Count to set the value

Pass Count 0200 - Set the Pass Count to be 0200H

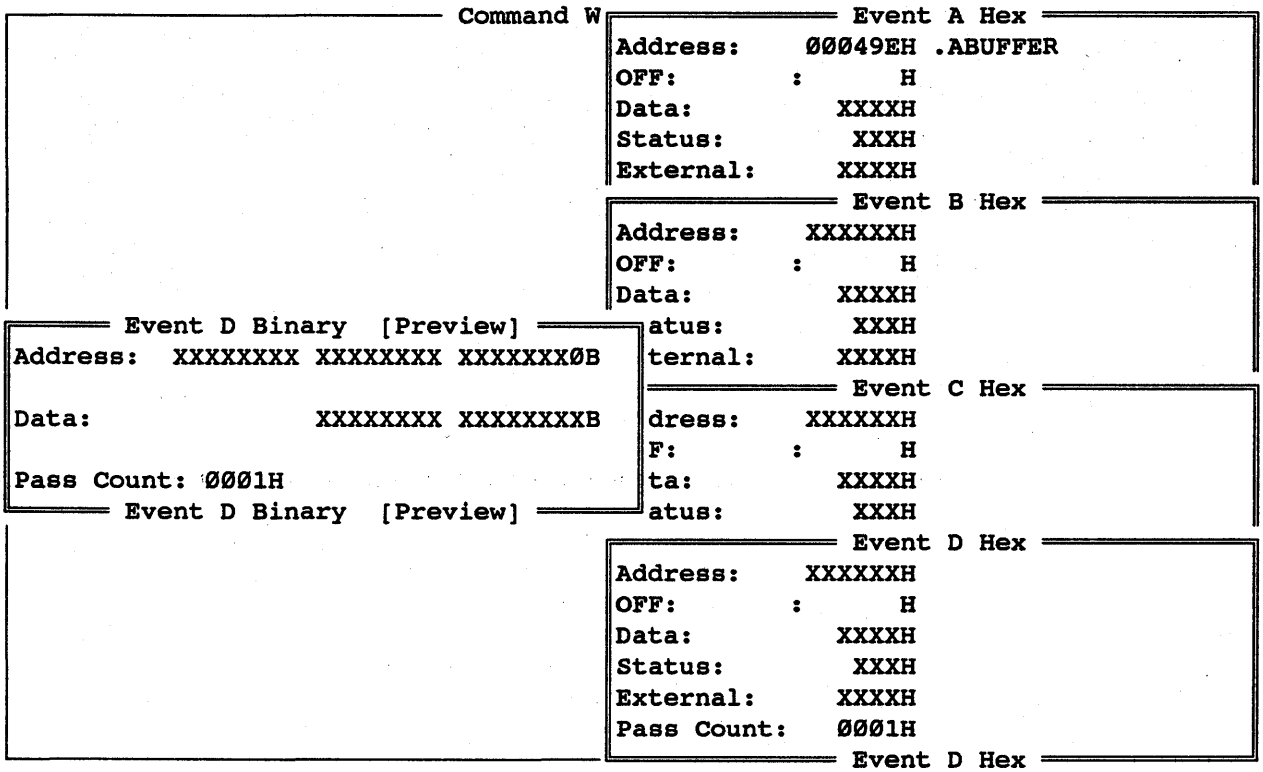


"ESC" - To delete this window

"HOME" - To view available windows

Arrow down to where Event is highlighted - Arrow to the Event window

Enter "TAB" - Enter TAB to view all events



"ESC" to delete Event D - Delete Event D

Set up Event C - Define parameters for Event C

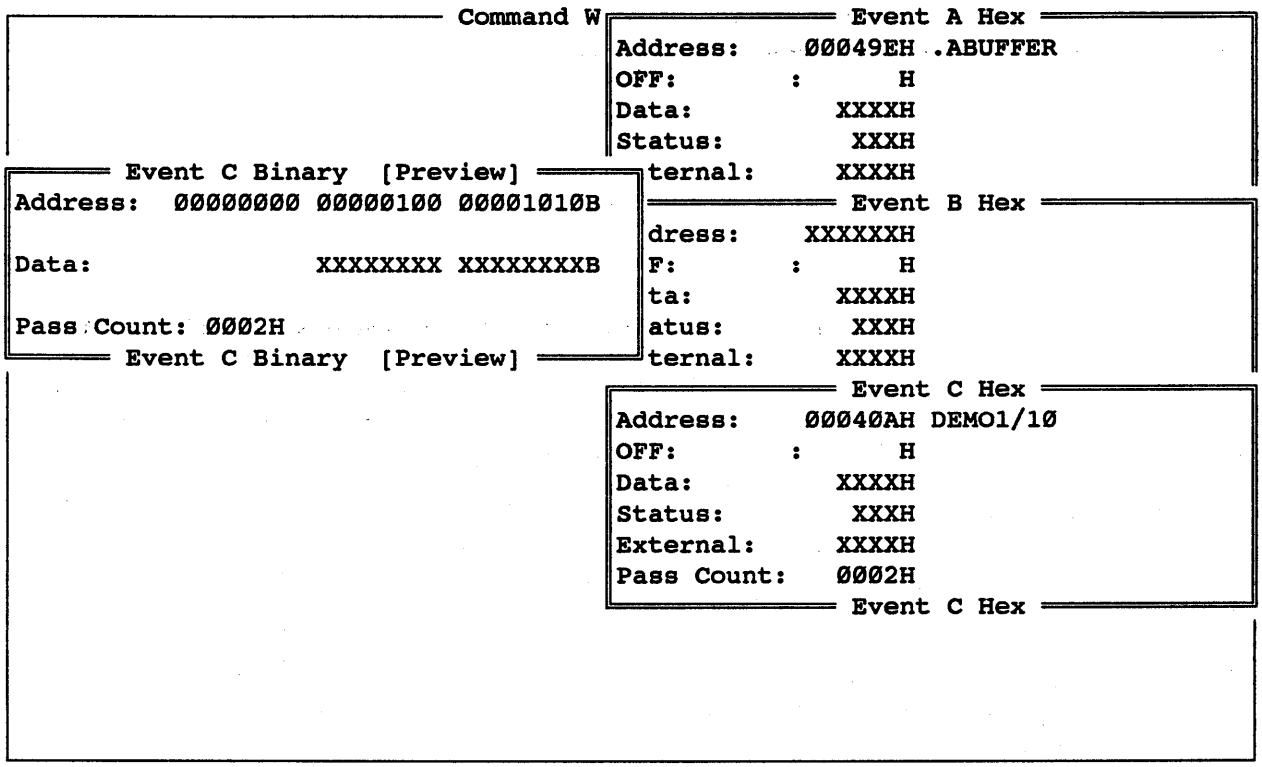
Type in a : - To define a symbol instead of the address

type in demo1/10 <cr> in highlighted space - Type in source code line number symbol "demo1/10" next to address field. "demo1/10" equals the address of line 10 in module demo1.

Address 00040A should appear - Address for source line "demo1/10"

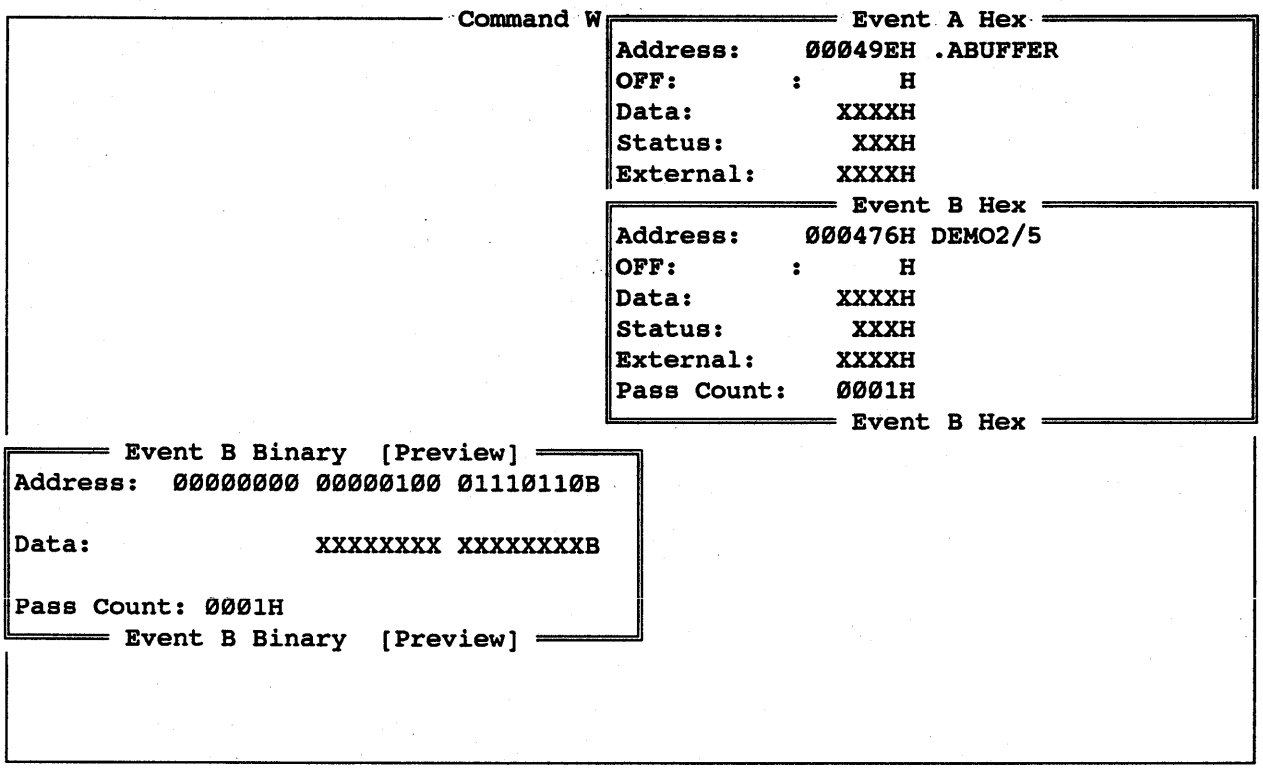
Arrow to Pass Count field

Pass Count 0002H - Change Pass Count to 0002H



"ESC" - To delete the window

Type in a :	- To define a symbol instead of an address
Type demo2/5 in high-lighted space	- Enter line 5 of module "demo2"
Address 000476H should appear	- Address of source code line "demo2/5" is displayed



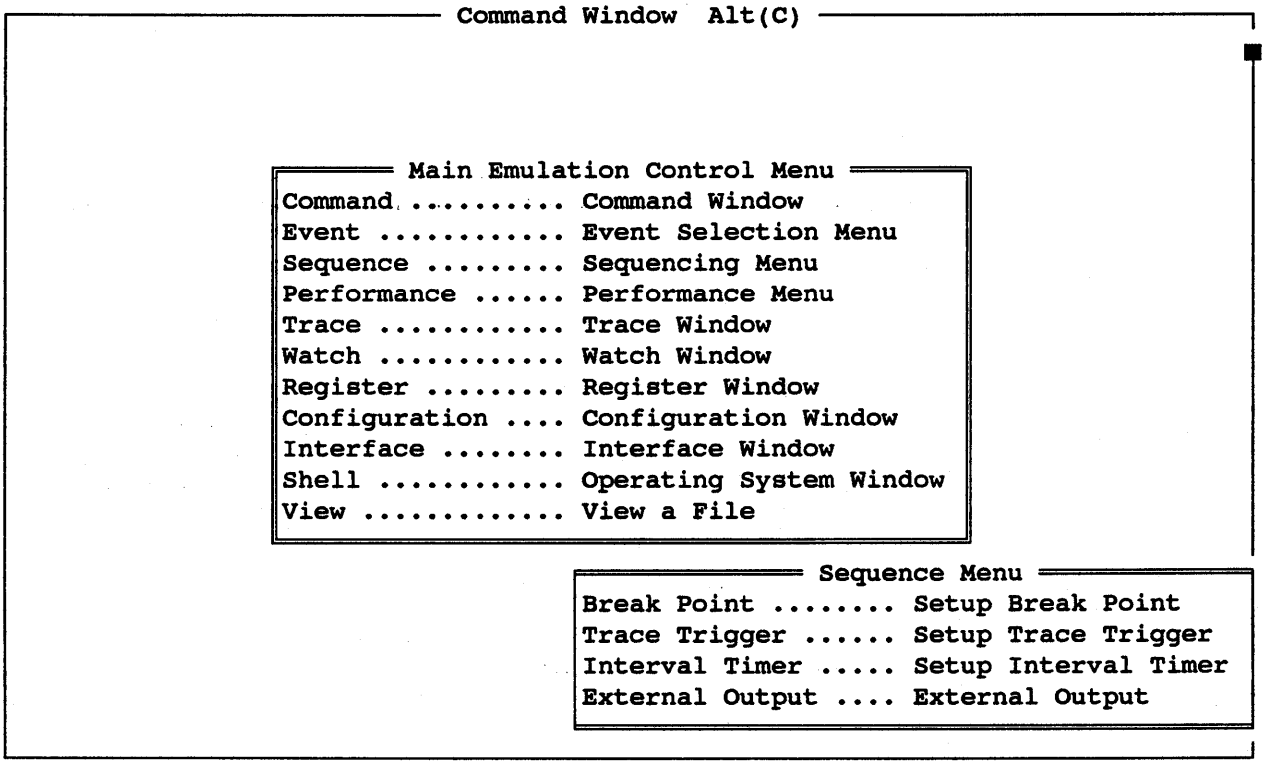
"ESC"	- To delete Event B window
"ESC"	- To delete Event A window

5.3.4 The Sequence Windows

The sequence functions of the emulator have been broken down into four windows including Break points, Trace Trigger, Interval timer, and External outputs.

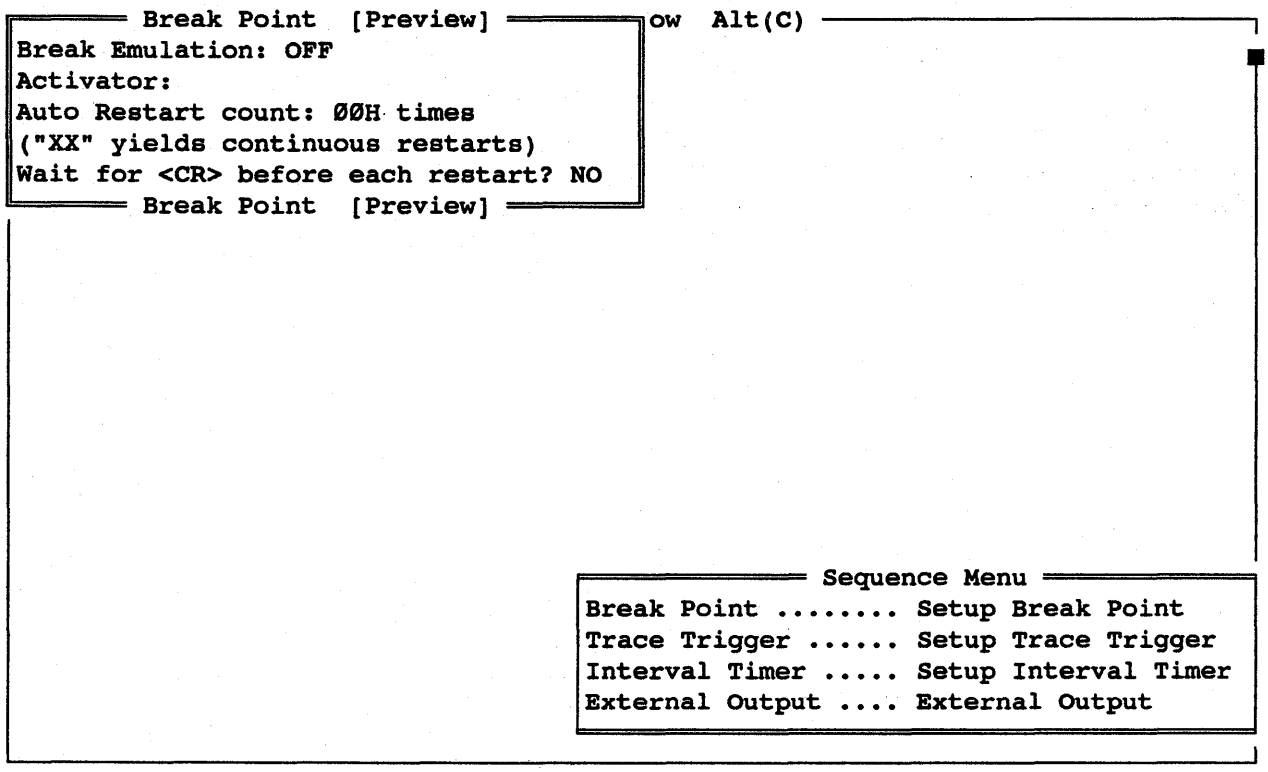
"HOME" - To view available windows

Arrow down to where Sequence is highlighted - Arrow to the Sequence window



Sequence <cr> - Select the Sequence Window by pressing return to enter

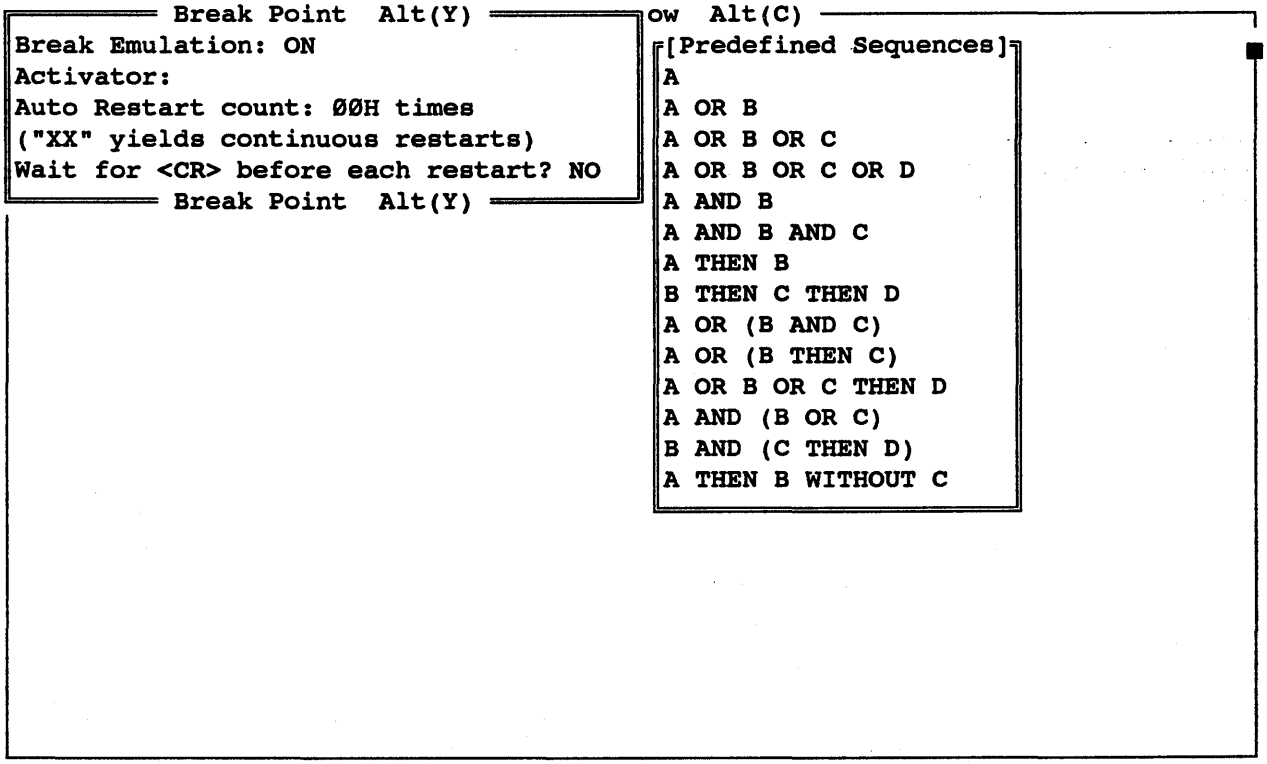
Arrow to where Break Point is highlighted - Arrow to the Break Point Option



Break Point <cr> - Enter the Break Point Option

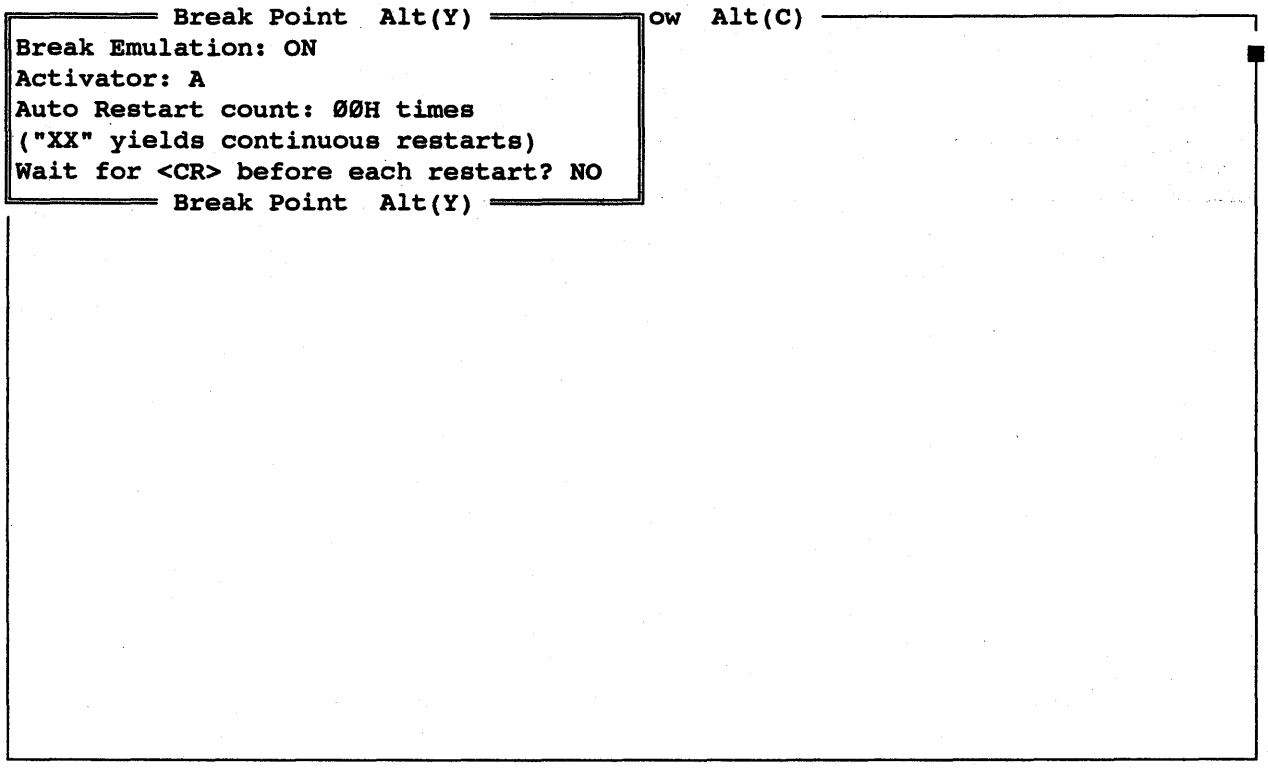
Break Emulation <cr> - Set the break emulation to be On or OFF

Arrow up for ON <cr> - Use arrow keys to select ON



Activator <cr> - Set the activator by pressing return for a list of predefined sequences

Arrow to defined sequence - Select A as the event to break on
A <cr>

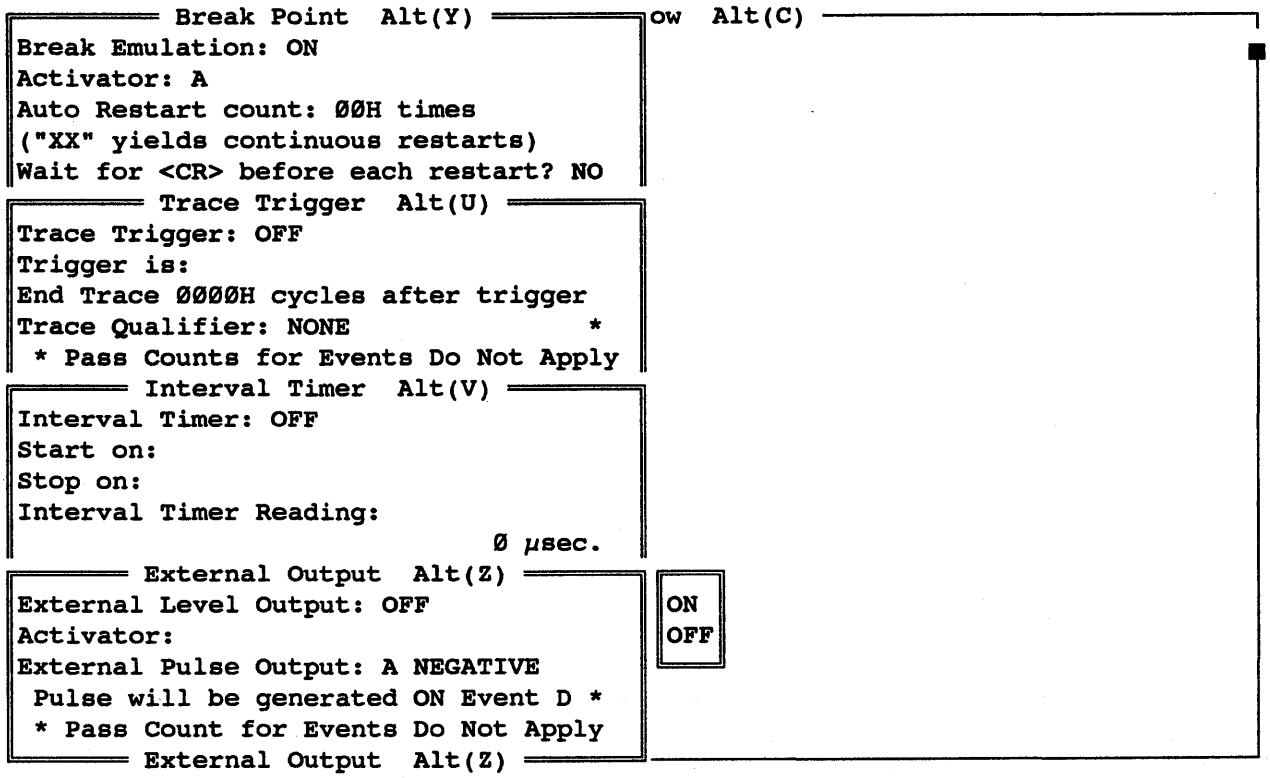


"ESC" - To delete the Break window

"HOME" - To view available windows

Arrow down to where Sequence is highlighted - Arrow to the Sequence window

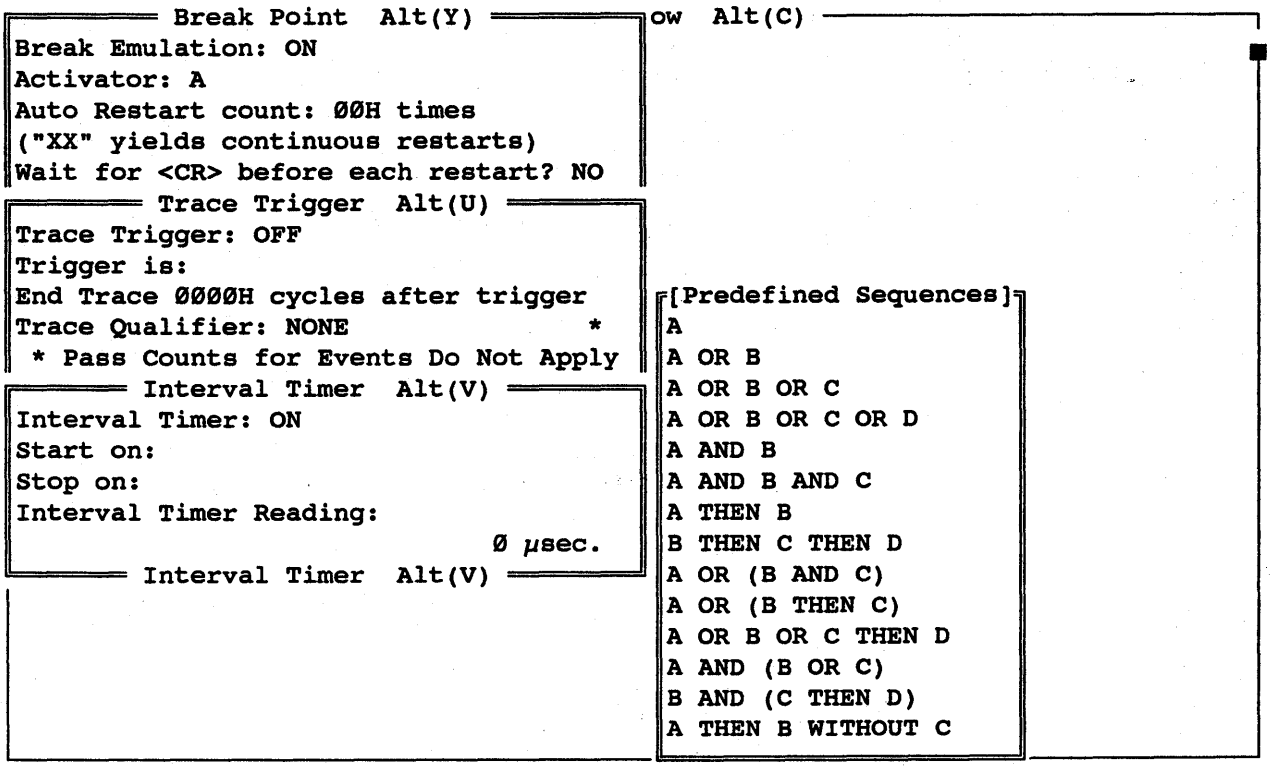
"TAB" - Press TAB to view all sequence options



"ESC" on External Output Window - Delete the External Output Window

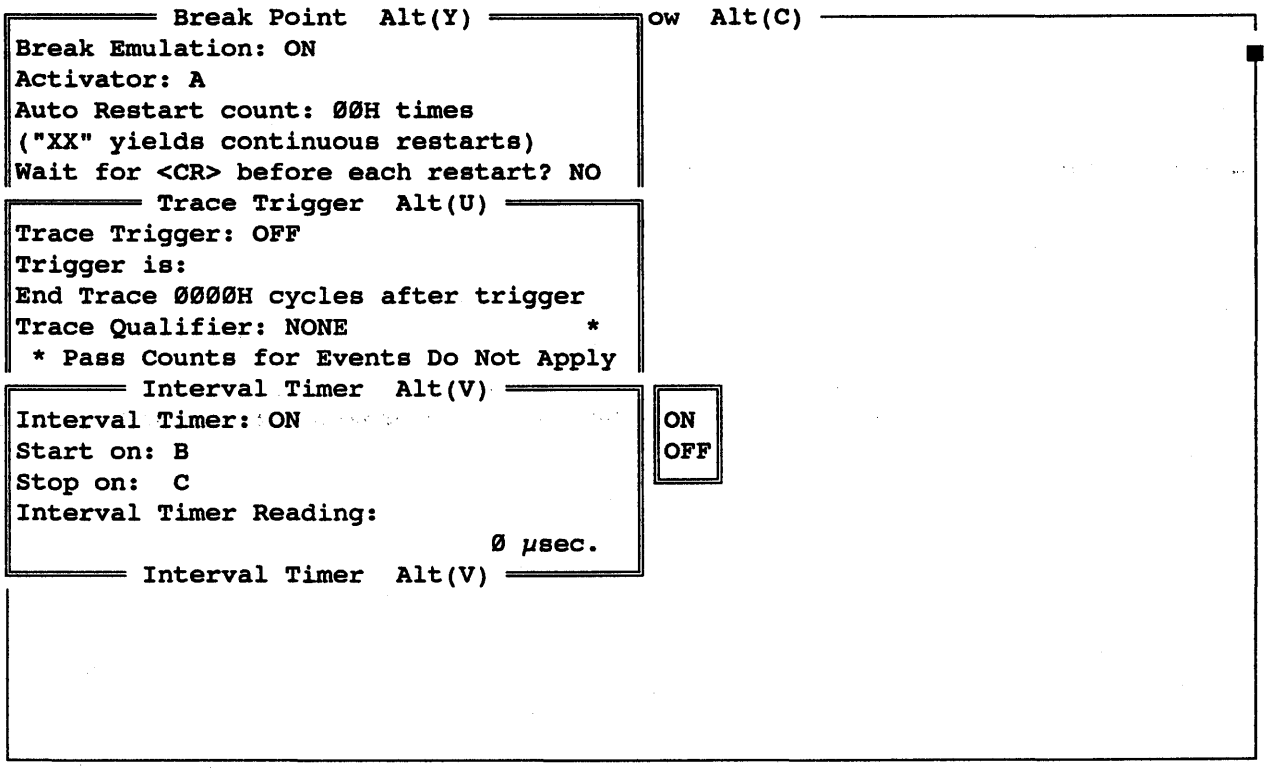
Interval Timer <cr> - Set the Interval Timer to be ON or OFF

Arrow up for ON <cr> - Use arrow keys to select ON



Start on: B <cr> - Timer will start counting on Event B

Stop on: C <cr> - Timer will stop counting on Event C

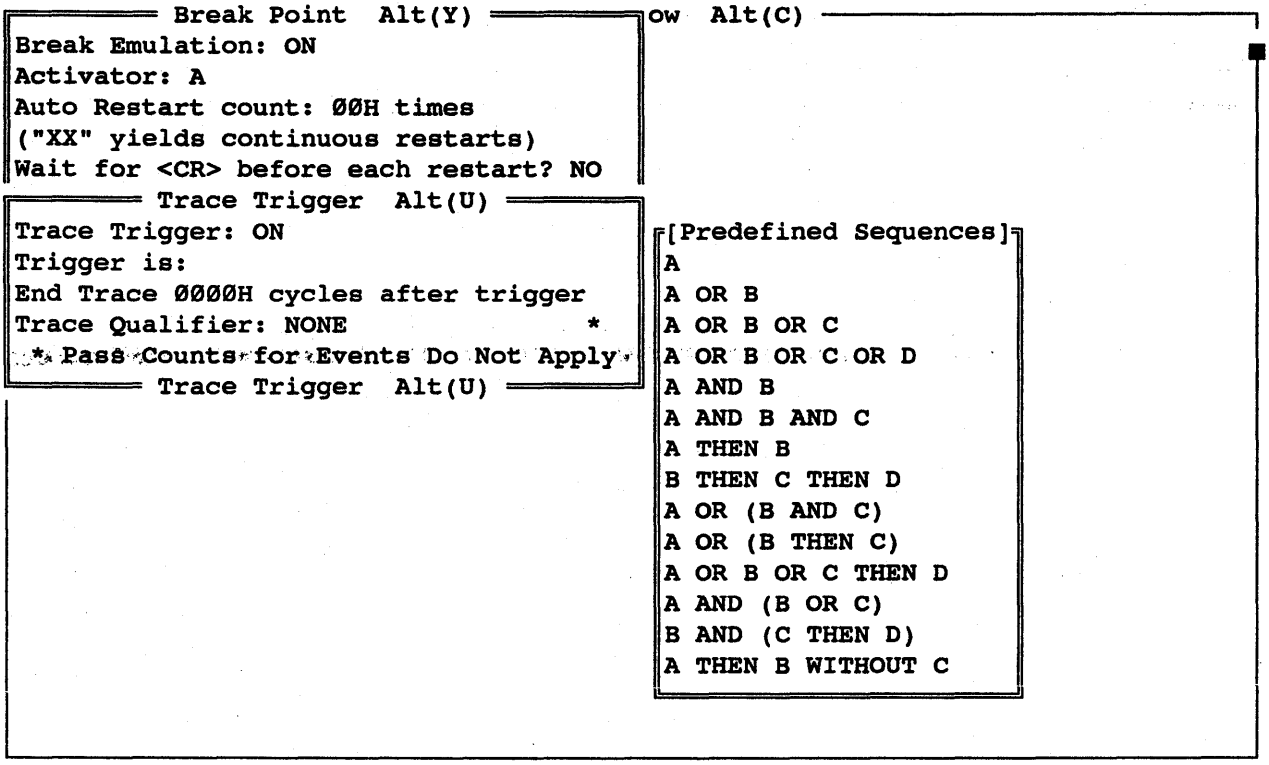


The timer value will be displayed in this window and in the Emulation Status window.

"ESC" - To delete the window

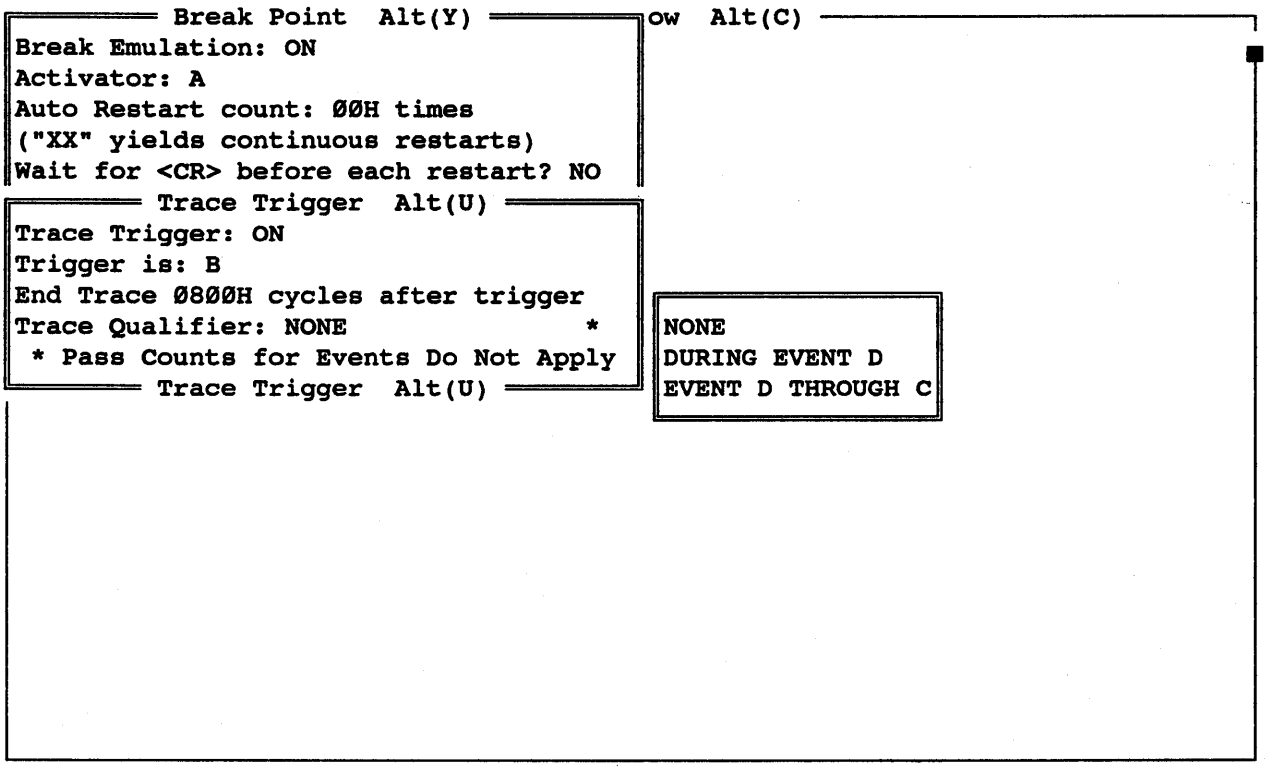
Trace Trigger <cr> - Set the Trace Trigger to be ON or OFF

Arrow up for ON <cr> - Use arrow keys to select ON



Trigger is : B <cr> - Type in B in activator field

End Trace 0800 - Type in 800H to be number of cycles traced after the trigger



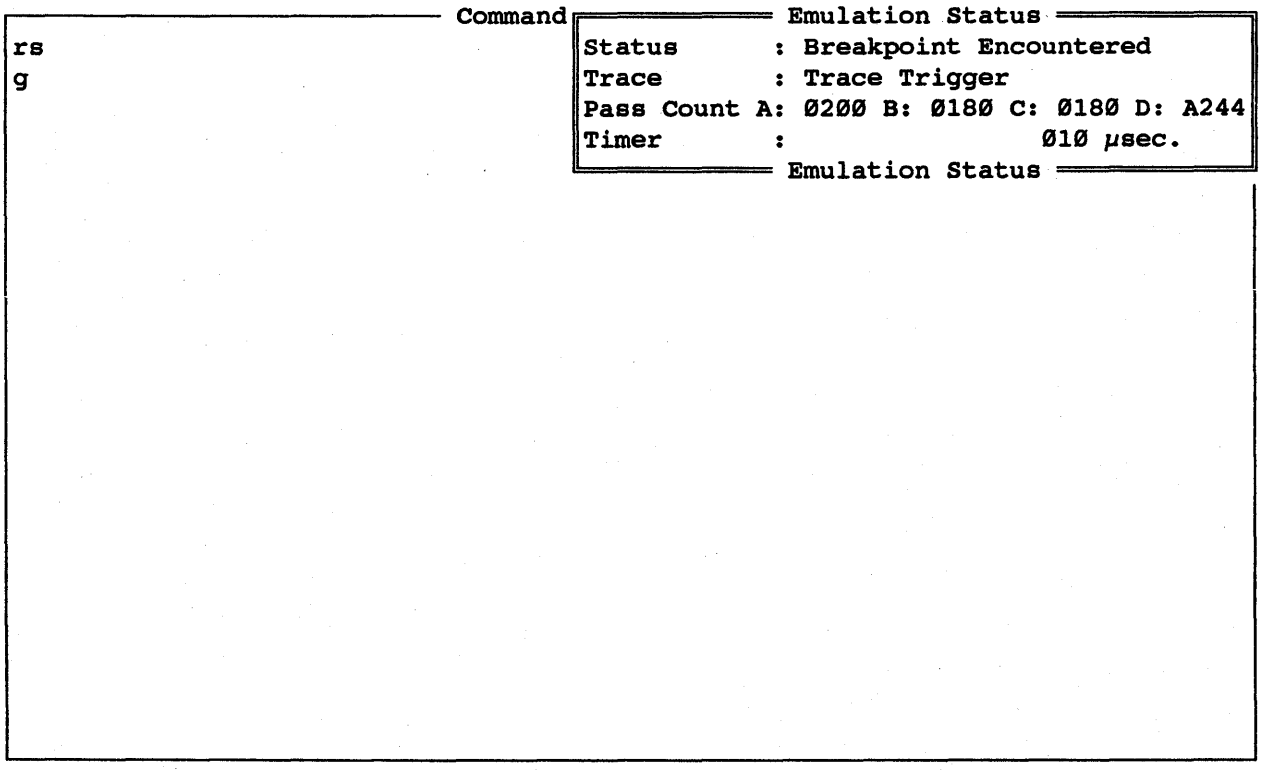
"ESC" - To delete Trigger window

"ESC" - To delete Break Point Window

5.3.5 The Trace Windows

The trace window will display the break and trigger trace buffer. The code can be viewed in disassembly, state, source, disassembly/state, disassembly/source or disassembly/state/source while in the trace buffer.

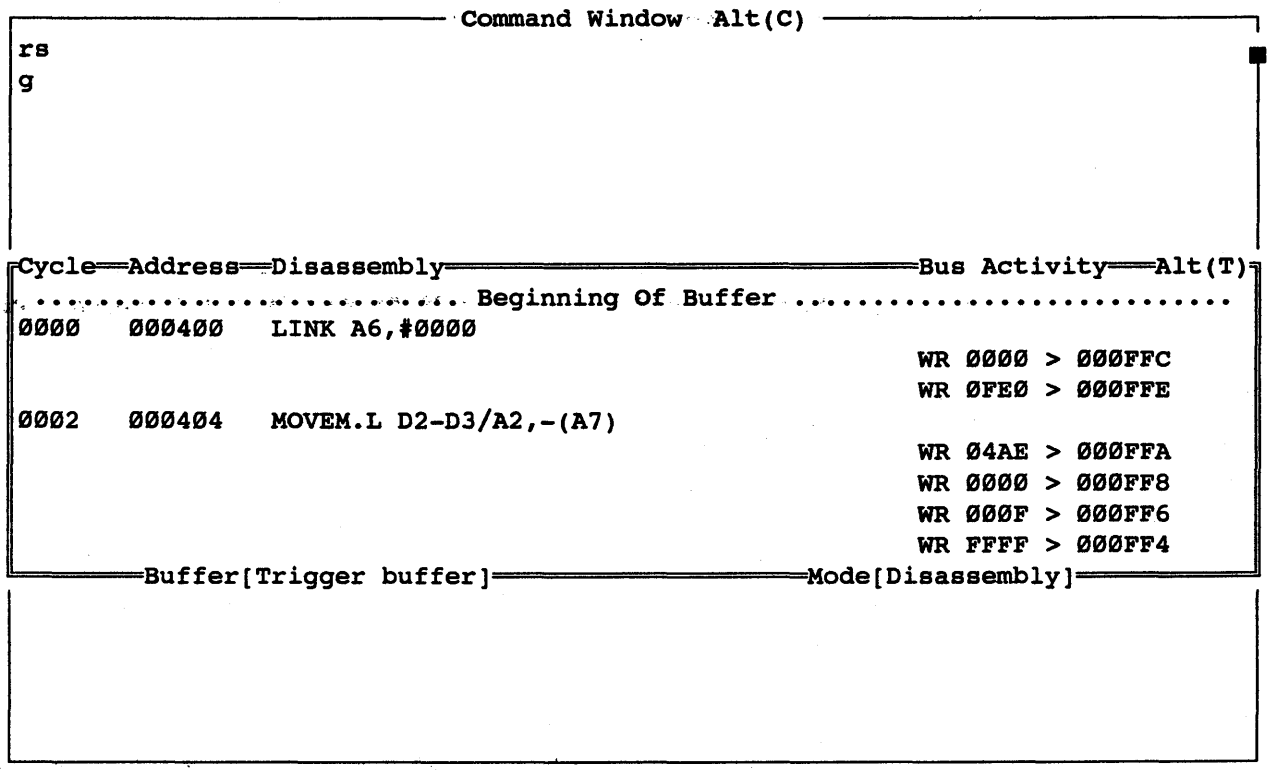
"ALT C"	- To return to command window
rs	- Resets the PC to 400H
g <cr>	- Starts the emulator at 400H This will put trace information into the trace buffers
Break Point Encountered Trace Trigger	- These messages will be displayed in the emulator status window



"HOME" - To view available windows

Arrow down to where Trace is highlighted - Arrow to the trace window

Trace <cr> - Select the Trace Window by pressing return



"F2" - Use F2 key to move and size window

Use arrow up key to move window to top of screen - Use the arrow keys to move the window

"END" - To switch to size mode

Use arrow down key to size the window to bottom of the screen - Use the arrow key to change the size

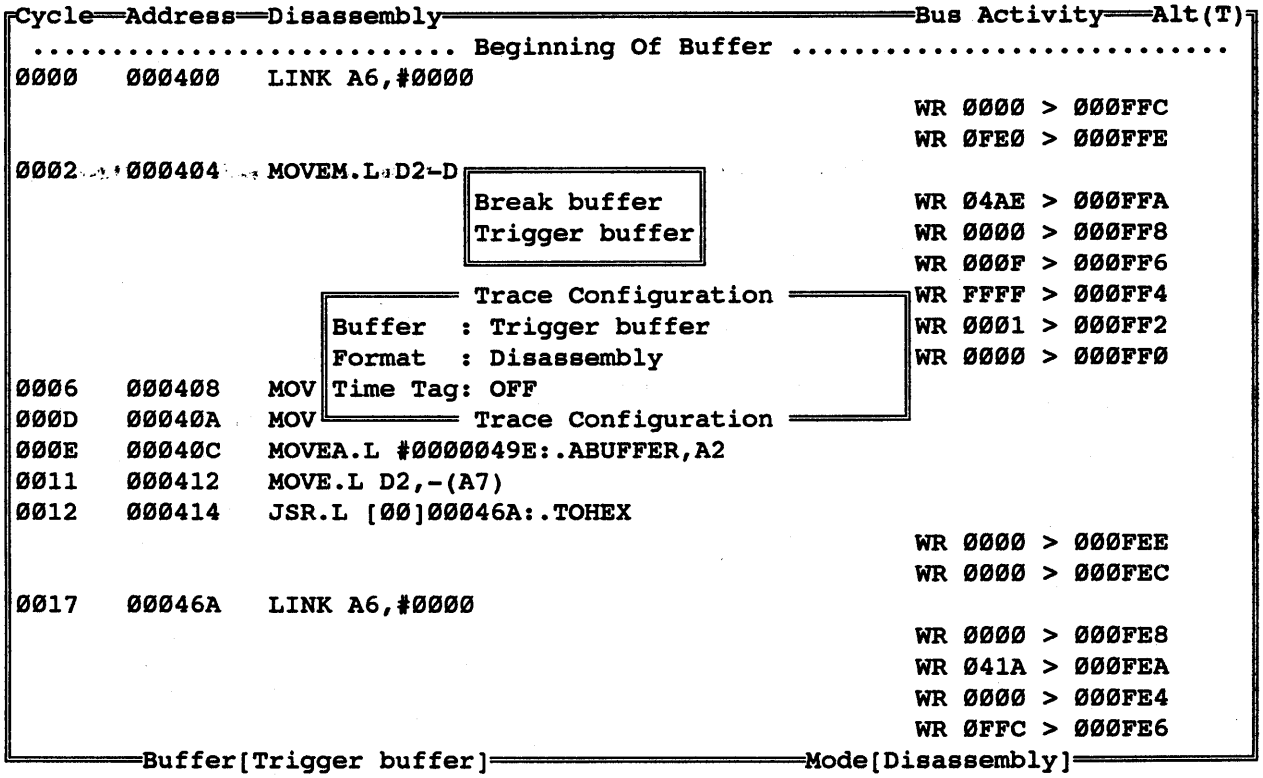
<cr> - To display newly sized Trace window

```
Cycle==Address==Disassembly=================================Bus Activity==Alt(T)
 ..... Beginning Of Buffer .....
0000    000400    LINK A6,#0000
                                                            WR 0000 > 000FFC
                                                            WR 0FE0 > 000FFE
0002    000404    MOVEM.L D2-D3/A2,-(A7)
                                                            WR 04AE > 000FFA
                                                            WR 0000 > 000FF8
                                                            WR 000F > 000FF6
                                                            WR FFFF > 000FF4
                                                            WR 0001 > 000FF2
                                                            WR 0000 > 000FF0
0006    000408    MOVEQ #00,D2
000D    00040A    MOVE.W D2,D3
000E    00040C    MOVEA.L #0000049E:.ABUFFER,A2
0011    000412    MOVE.L D2,-(A7)
0012    000414    JSR.L [00]00046A:.TOHEX
                                                            WR 0000 > 000FEE
                                                            WR 0000 > 000FEC
0017    00046A    LINK A6,#0000
                                                            WR 0000 > 000FE8
                                                            WR 041A > 000FEA
                                                            WR 0000 > 000FE4
                                                            WR 0FFC > 000FE6
=========Buffer[Trigger buffer]=============================Mode[Disassembly]
```

"F4" - To zoom in and ....

"F4" - .... out of different size Trace windows

<cr> - Entering a <cr> brings up the Trace Configuration window. Trigger Buffer is highlighted in the buffer field. This means that the trace displayed is the Trigger Trace Buffer. Notice that [Trigger Buffer] is displayed in the bottom border of the trace buffer window also.



Arrow down to the Format Field - Arrow to the Format Field

<cr>	- Entering a <cr> will allow you to change the Trace display format
Arrow down to where Disassembly/Source is highlighted	- Select the disassembly/source option as the format
Disassembly/Source <cr>	- Set the option to be both disassembly and source
ESC	- ESC deletes the Trace Configuration window and changes the Trigger Trace buffer to display both disassembly and source. Notice that [Disassembly/Source] is displayed in the bottom border of the trace buffer window also.

```
Cycle==Address==Disassembly=================================Bus Activity===Alt(T)
 ......................... Beginning Of Buffer .............................
0000   000400   LINK A6,#0000
                                                          WR 0000 > 000FFC
                                                          WR 0FE0 > 000FFE
0002   000404   MOVEM.L D2-D3/A2,-(A7)
                                                          WR 04AE > 000FFA
                                                          WR 0000 > 000FF8
                                                          WR 000F > 000FF6
                                                          WR FFFF > 000FF4
                                                          WR 0001 > 000FF2
                                                          WR 0000 > 000FF0
DEMO1:1:extern int    tohex();              /* Binary to HEX converter    */
DEMO1:2:extern        swap();               /* Byte swap routine          */
DEMO1:3:char   abuffer[16];                 /* ASCII hex buffer           */
DEMO1:4:char   zbuffer[16];                 /* Zero buffer                */
DEMO1:5:main()
DEMO1:6:{
DEMO1:7:register int i;
DEMO1:8:for (;;) {                                        /* "Forever"    */
DEMO1:9:       for (i = 0; i < sizeof abuffer; i++)
0006   000408   MOVEQ #00,D2
DEMO1:10:              abuffer[i] = tohex(i);     /* Fill ASCII buffer */
000D   00040A   MOVE.W D2,D3
========Buffer[Trigger buffer]==================Mode[Disassembly/Source]=======
```

B - Enter B to view beginning of Trigger Trace Buffer. Notice that "C" source code is displayed in the buffer

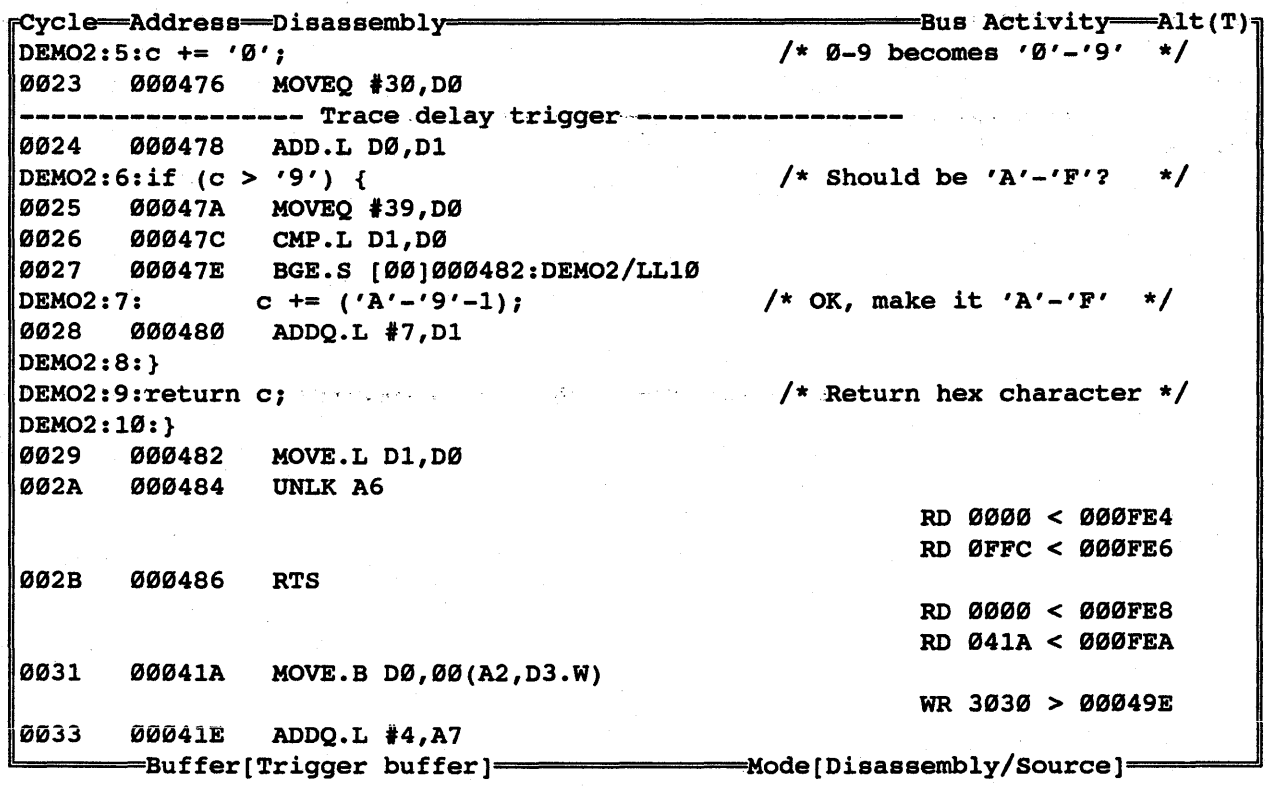
```
Cycle==Address==Disassembly=================================Bus Activity==Alt(T)
 ..... Beginning Of Buffer .....
0000    000400   LINK A6,#0000
                                                          WR 0000 > 000FFC
                                                          WR 0FE0 > 000FFE
0002    000404   MOVEM.L D2-D3/A2,-(A7)
                                                          WR 04AE > 000FFA
                                                          WR 0000 > 000FF8
                                                          WR 000F > 000FF6
                                                          WR FFFF > 000FF4
                                                          WR 0001 > 000FF2
                                                          WR 0000 > 000FF0
DEMO1:1:extern int    tohex();              /* Binary to HEX converter    */
DEMO1:2:extern         swap();              /* Byte swap routine          */
DEMO1:3:char   abuffer[16];                 /* ASCII hex buffer           */
DEMO1:4:char   zbuffer[16];                 /* Zero buffer                */
DEMO1:5:main()
DEMO1:6:{
DEMO1:7:register int i;
DEMO1:8:for (;;) {                                        /* "Forever"    */
DEMO1:9:        for (i = 0; i < sizeof abuffer; i++)
0006    000408   MOVEQ #00,D2
DEMO1:10:               abuffer[i] = tohex(i);      /* Fill ASCII buffer */
000D    00040A   MOVE.W D2,D3
=========Buffer[Trigger buffer]=================Mode[Disassembly/Source]=======
```

E - Enter E to view end of Trigger Trace Buffer

T - Enter T to view the code following the Trigger Point. The top of the display is the next instruction after the trigger point.

Arrow up to where Trace Delay Trigger is displayed

- The trigger point is the cycle located above Trace Delay Trigger



<cr> - Enter a <cr> to bring up the Trace Configuration Window

Arrow to Buffer Field - Set the buffer to view break point

<cr> - Entering a <cr> allows you to choose between the break buffer and trigger buffer

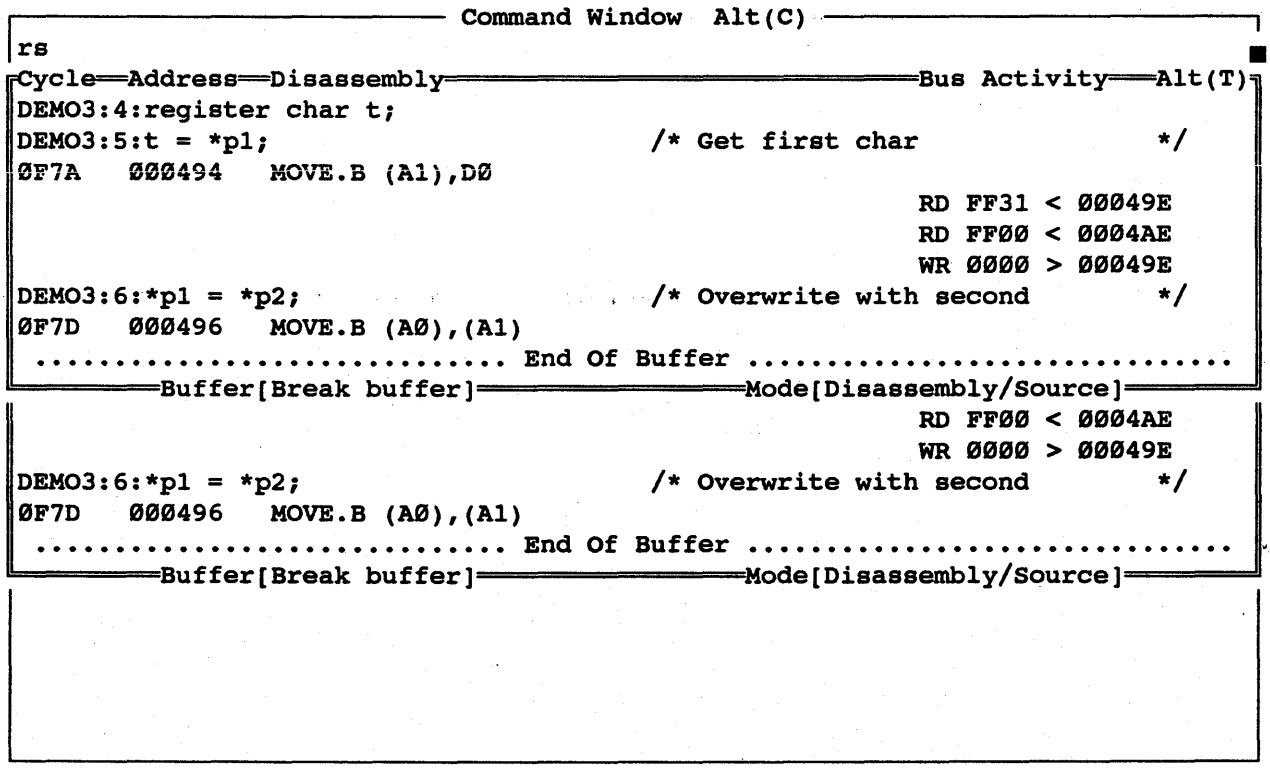
Break Buffer <cr>	- Choose Break Buffer to be displayed
ESC	- Delete the Trace Configuration Window
B	- View beginning of break buffer
E	- View end of break buffer

```
Cycle==Address==Disassembly=================================Bus Activity===Alt(T)
0F6E   000488   LINK A6,#0000
                                                          WR 0000 > 000FE4
                                                          WR 045E > 000FE6
                                                          WR 0000 > 000FE0
                                                          WR 0FFC > 000FE2
0F72   00048C   MOVEA.L 0008(A6),A1
                                                          RD 0000 < 000FE8
                                                          RD 049F < 000FEA
0F76   000490   MOVEA.L 000C(A6),A0
                                                          RD 0000 < 000FEC
                                                          RD 04AF < 000FEE
DEMO3:1:swap(p1,p2)                              /* Swap bytes at *p1 and *p2 */
DEMO3:2:register char *p1, *p2;
DEMO3:3:{
DEMO3:4:register char t;
DEMO3:5:t = *p1;                          /* Get first char                 */
0F7A   000494   MOVE.B (A1),D0
                                                          RD FF31 < 00049E
                                                          RD FF00 < 0004AE
                                                          WR 0000 > 00049E
DEMO3:6:*p1 = *p2;                        /* Overwrite with second          */
0F7D   000496   MOVE.B (A0),(A1)
 ............................. End Of Buffer ..............................
==========Buffer[Break buffer]=================Mode[Disassembly/Source]=========
```

5.3.5.1 Duplicate Trace Windows

Duplicate windows can be displayed on the screen at the same time. Several trace windows can be active on the same screen. This feature can be used to put a certain section of the trace in one window while scrolling the buffer in the 2nd window.

F4	- While viewing the trace buffer, switch to the smaller trace window display
"HOME"	- To view available windows
Arrow down to where Trace Window is highlighted	- Arrow to Trace window
Hold the "CNTRL" key while striking a <cr> on the Trace Window	- To create a duplicate Trace window
2nd Trace Window on top of first	- The duplicated Trace window is on top of the first Trace window
F2	- Use the F2 key to move and/or resize the trace windows as was done before



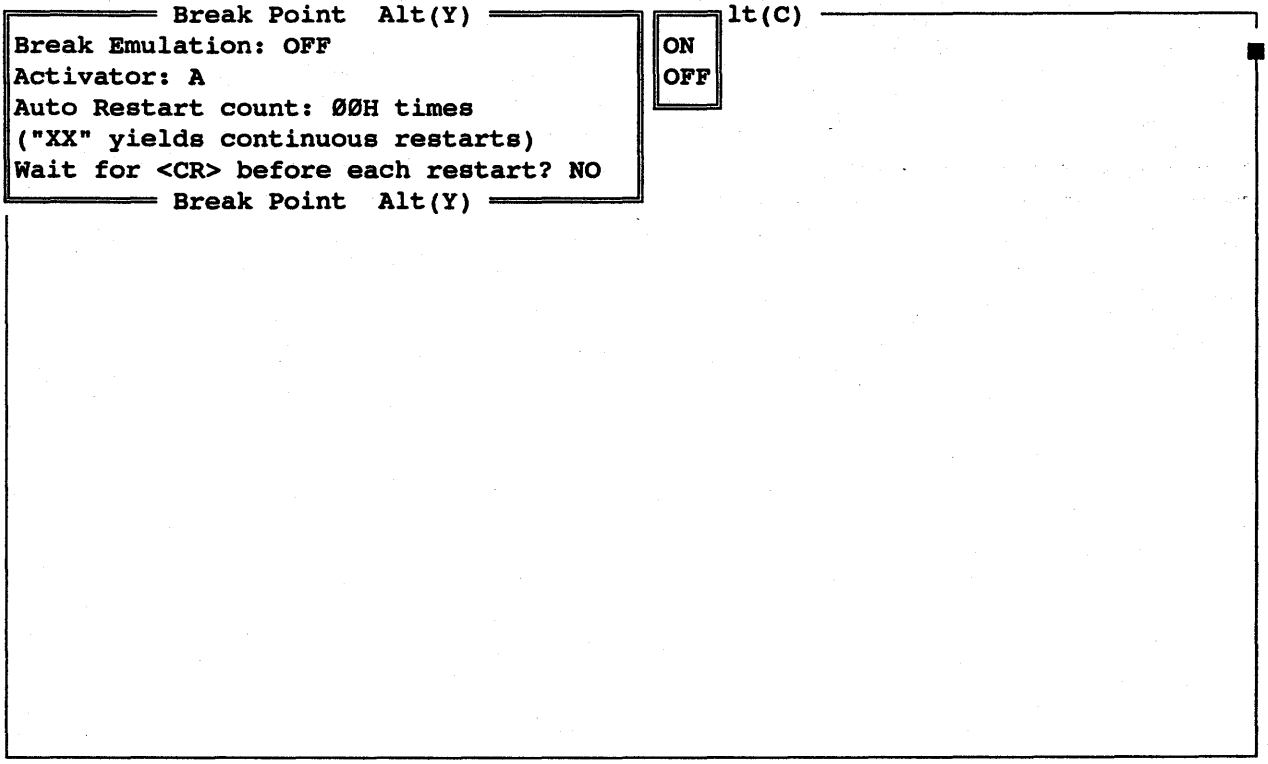
"CNTRL-PGUP" or "CNTRL-PGDN"	- To toggle between duplicate Trace windows
"ESC"	- Delete first Trace window
"ESC"	- Delete second Trace window

5.3.5.2 Freeze Trace

We will view the feature of the freeze trace which allows you to freeze or hold the trace buffer while the emulator continues to run. If the trigger condition plus delay has been met, then the freeze trace will display the trigger trace buffer. If the trigger condition plus delay has not been met, then the freeze trace will display a "snapshot" of the last 4K cycles of code executed before the freeze trace command was given.

"ALT F1"	- To view available windows using the ALT key to select
Arrow to "ALT Y"	- Find "ALT Y" in Alt key window
"ALT" (Y) <cr>	- Select ALT (Y) for Break Point
Break Emulation <cr>	- Set the break emulation to be ON or OFF

Arrow to OFF <cr> - Use arrow keys to select OFF



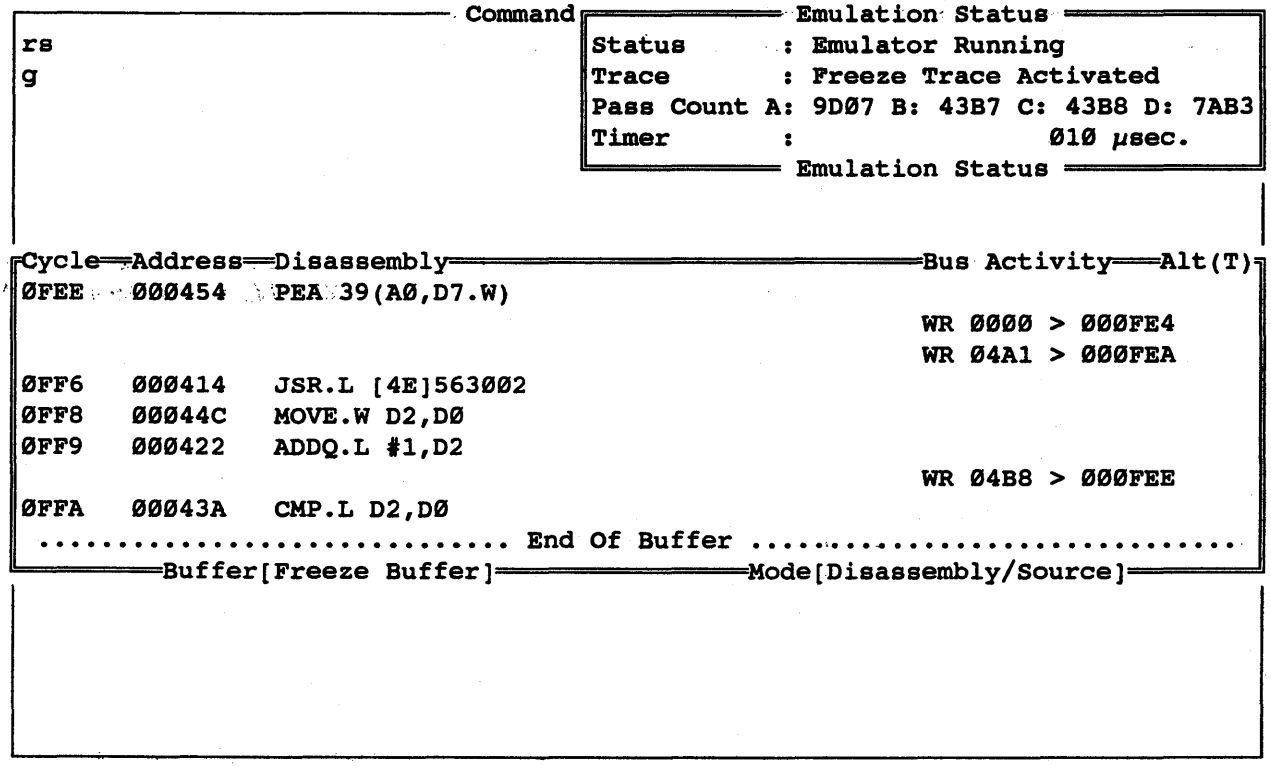
ESC - Delete the Break Point window

rs <cr> - Reset the PC to 4ØØH

g <cr> - Tell the emulator to go

"ALT" (T) for Trace Window - Select ALT (T) Trace Window

"F" - To freeze the trace buffer. This will display the trigger trace buffer since the trigger condition plus delay was met. Notice that [Freeze Buffer] is displayed in the bottom border of the Freeze window.

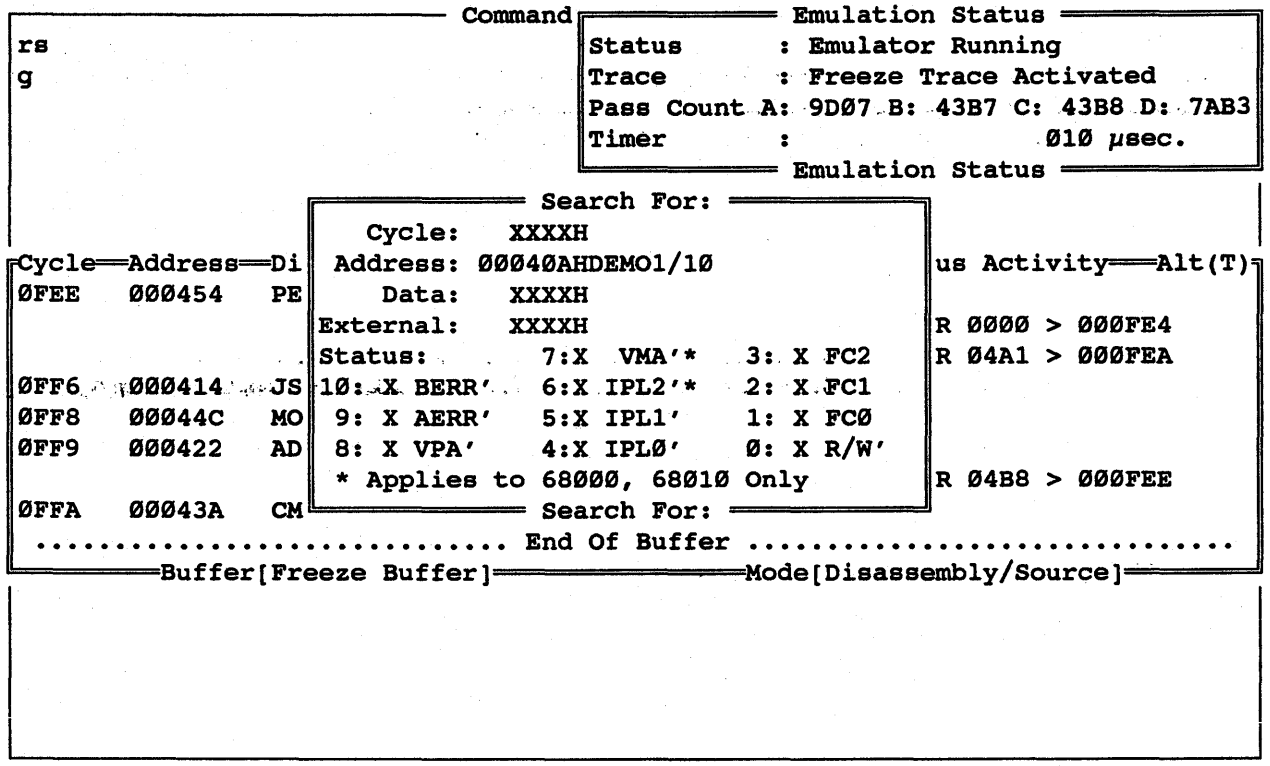


"CNTRL S" - To setup parameters for search

Arrow to Address field - Select the address option

Type in a ":" - To define a symbol to search for

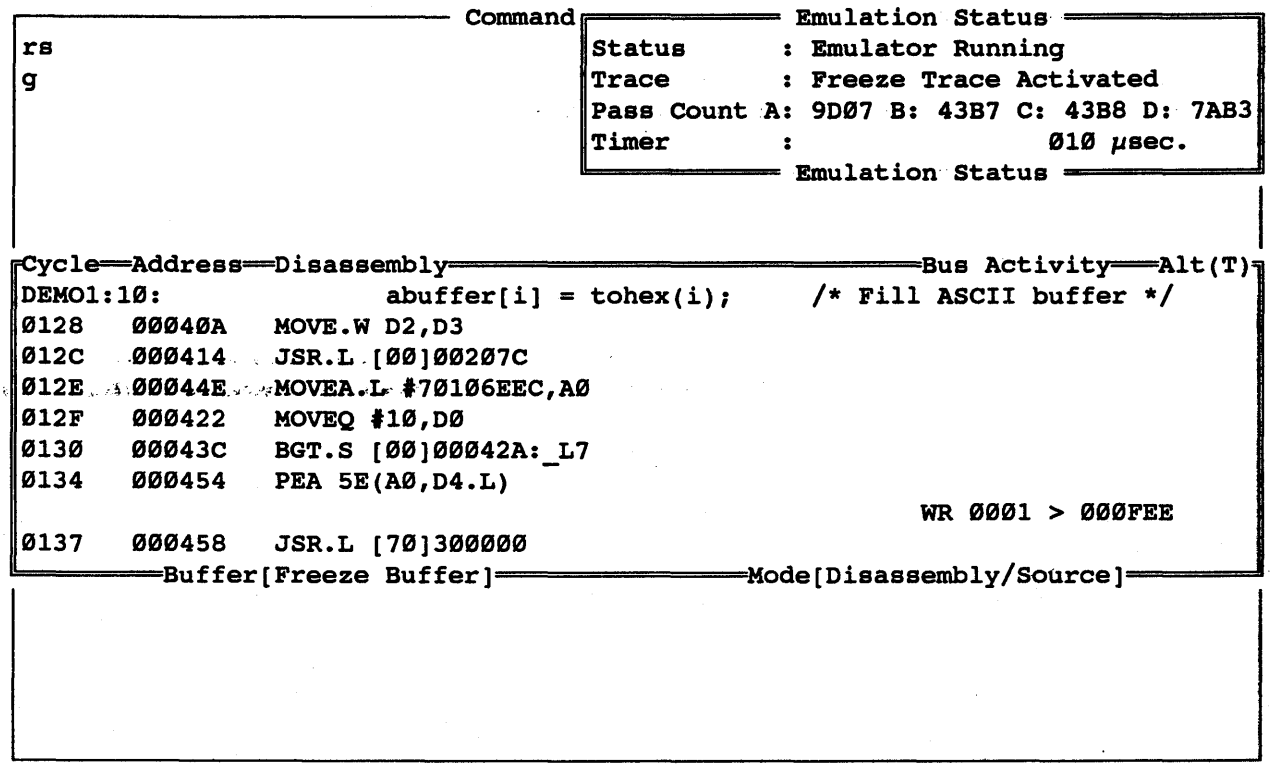
Type in demo1/10 <cr> next to address field - Type in source code line 10 of module demo1 to search for



ESC - Deletes the "Search For:" window

"B" - To go to top of trace buffer

"S" - To do the search. The top of the display shows line 10 of module demo1



"R" - To resume the trace

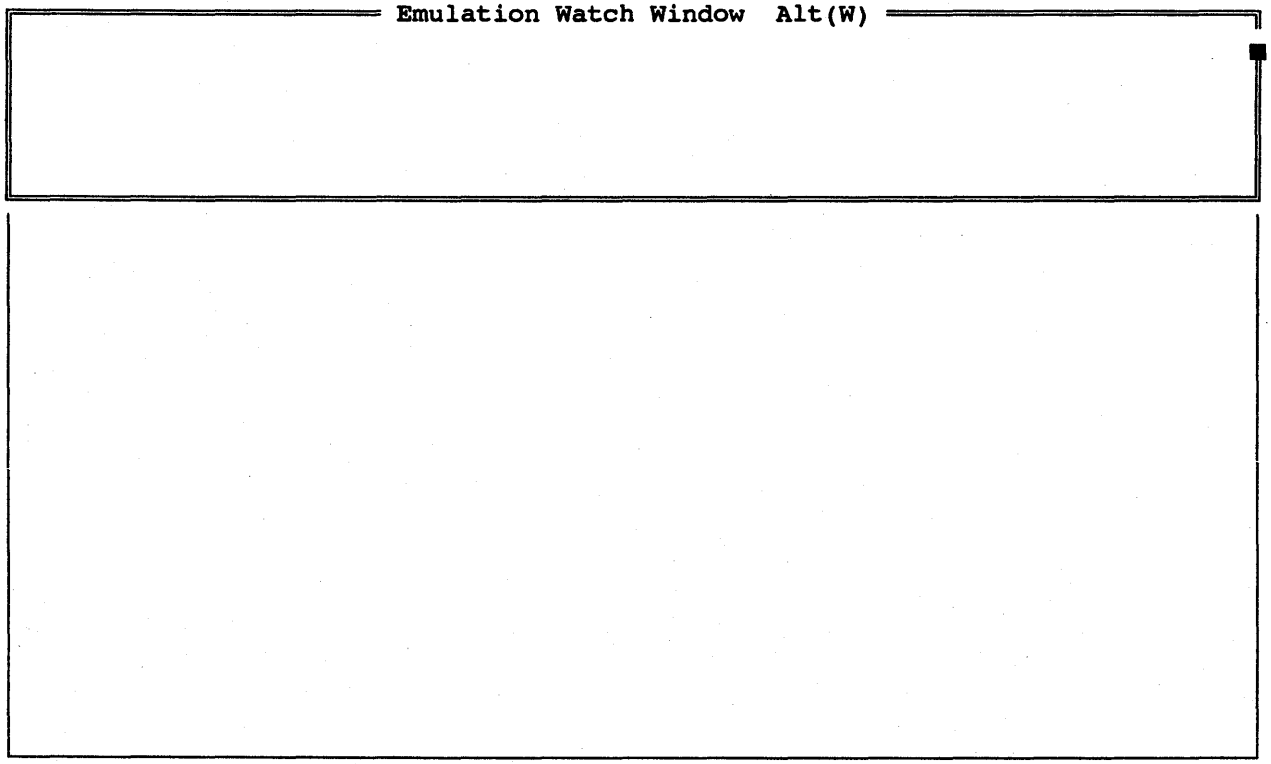
ESC - Deletes Trace window

rs <cr> - Resets the emulator

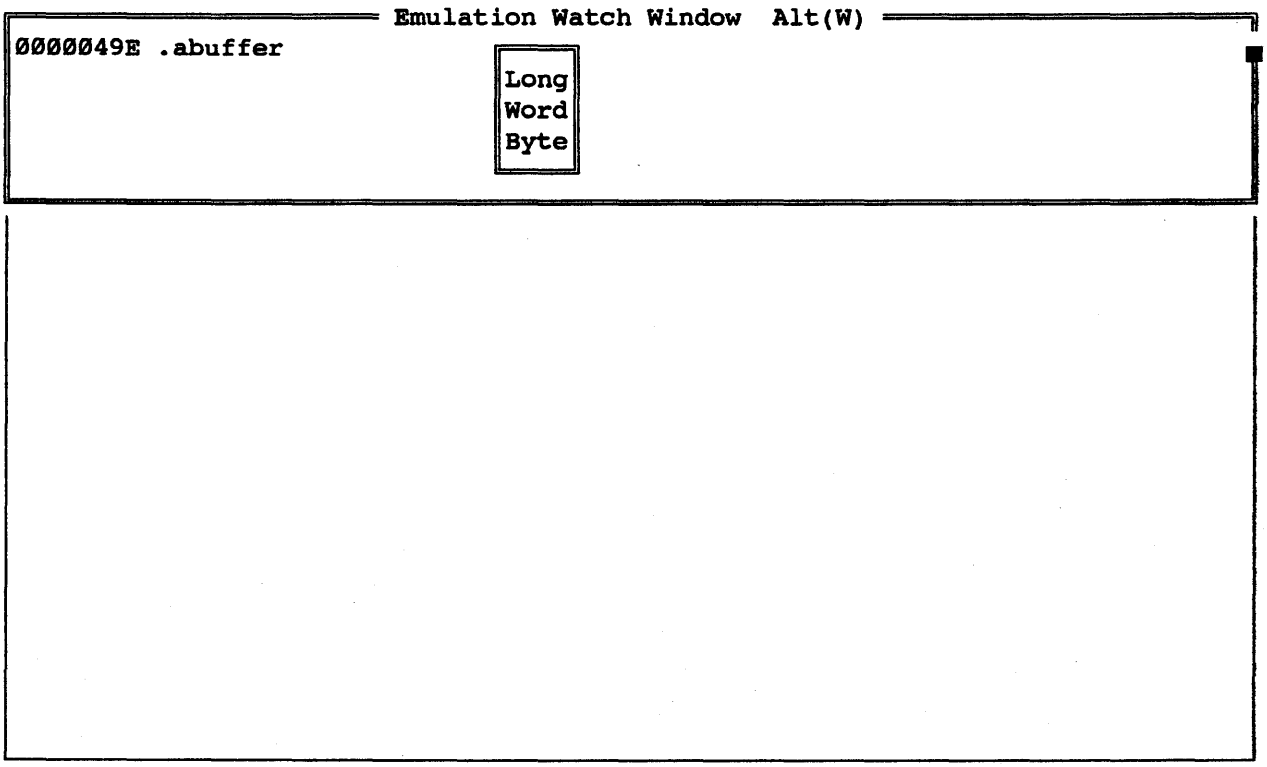
5.3.6 The Watch Window

The watch window is used to select certain memory locations which are to be monitored continuously.

"HOME"	- To view available windows
Arrow down to where to Watch is highlighted	- Arrow to the Watch Window
Watch Window <cr>	- Select the Watch Window

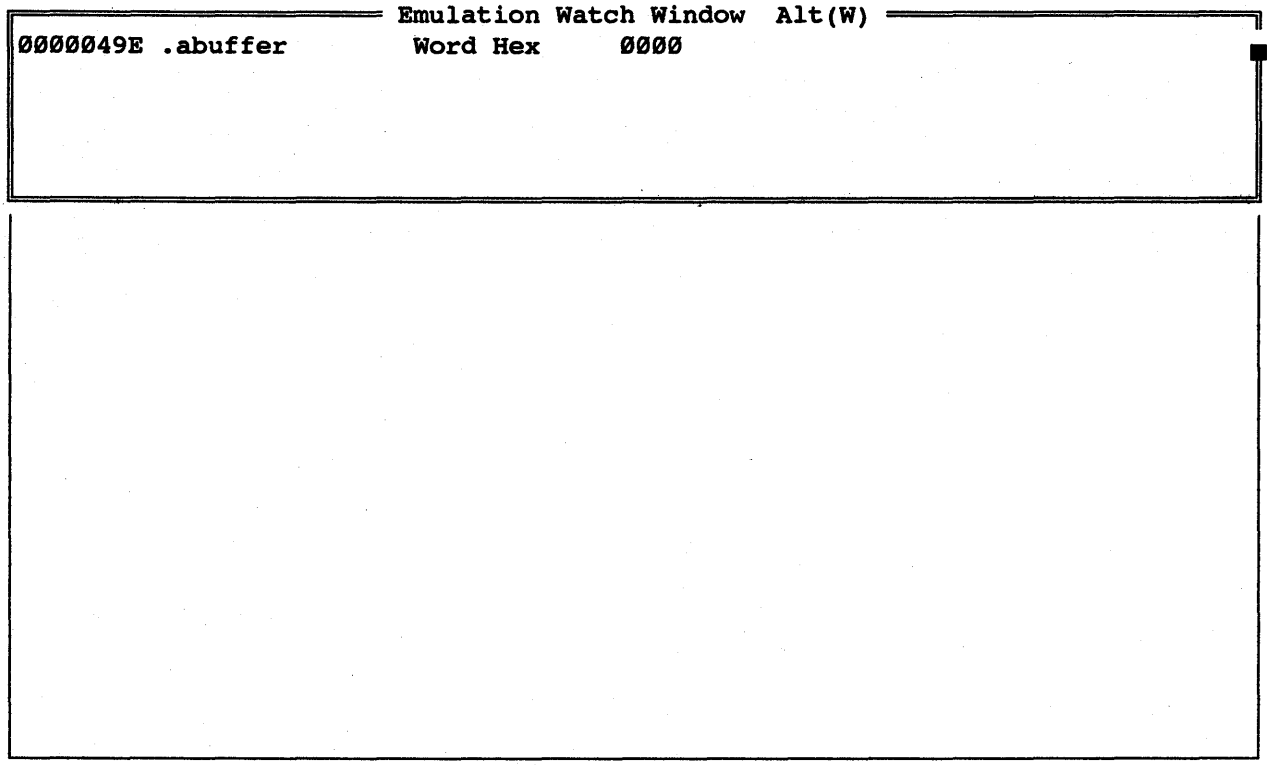


: - Enter symbol entry field



Arrow down to Word <cr> - Select the Word option

Arrow to Hex <cr> - Select the Hex option



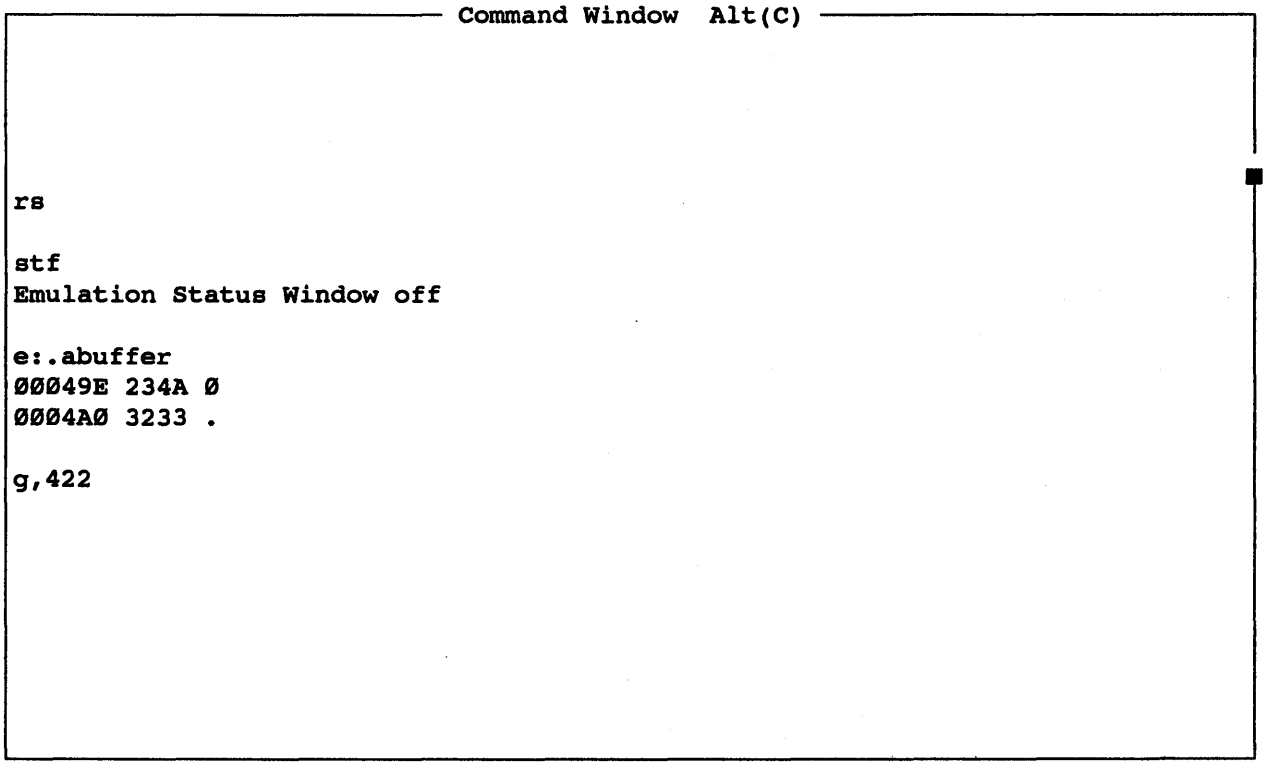
"ALT (C)" - To go to Command Window

rs - Sets PC to 400H

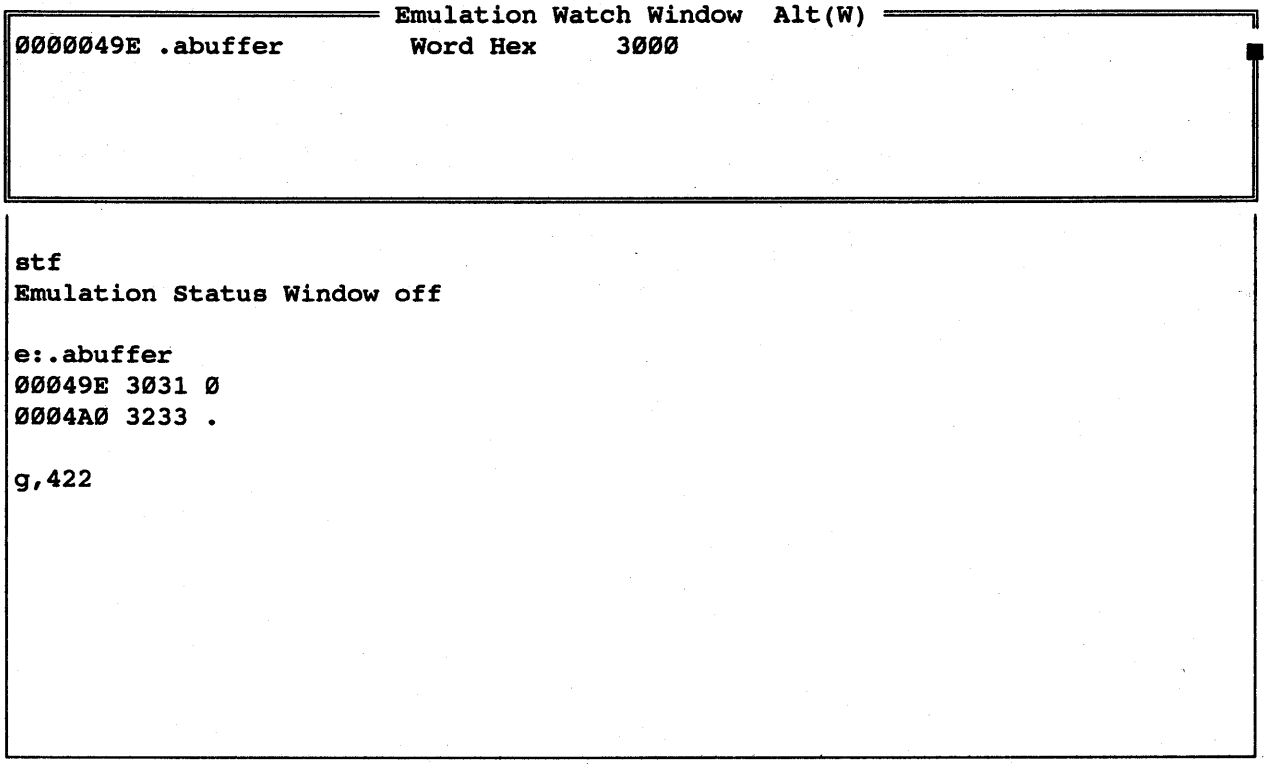
stf - Turn off status window

e:.abuffer <cr>
0 <cr> - Change .abuffer location to zero

g,422 - To start the emulator and set temporary breakpoint at 422



"Breakpoint Encountered"	- Will be displayed
"CNTRL-PGUP"	- To display Watch window
Watch Window has been updated	- Whenever the emulator is stopped or single-stepped, the watch window value is updated and each digit that has changed since the last update is highlighted



"ESC" - To delete Watch Window

"CNTRL-HOME" - To clear Command Window

5.3.7 The Configuration Window

The configuration window is used to define emulator parameters for proper interface to the target system.

"HOME"	- To view available windows
Arrow to where Configuration is highlighed	- Arrow to the Configuration Window
Configuration <cr>	- Select the Configuration window by pressing return

```
.Idling.....................................................................

  Memory Map -      Unallocated Memory = 0000K  |       Processor: 68000
 Block  Combine     Base      Ending    Block  |  Clock     Emulator
  Number  Type      Address   Address   Size   |  Target System (BR,IPL0-2,
    1     UP UD SP SD 000000H  03FFFFH  0256K  |  BGACK,BERR):  Enabled
    2     -- -- -- -- 000000H  000000H  0000K  |  Allow bus arbitration:
    3     -- -- -- -- 000000H  000000H  0000K  |   Only During Emulation
    4     -- -- -- -- 000000H  000000H  0000K  |  DTACK source is
BLK 1   S/U = SUPV/USER  P/D = PROG/DATA       |      From the Emulator

      E = Emulator, T = Target System, RO = Read Only, RW = Read/Write
000000 E RW  02C000 E RW
004000 E RW  030000 E RW
008000 E RW  034000 E RW
00C000 E RW  038000 E RW
010000 E RW  03C000 E RW
014000 E RW
018000 E RW
01C000 E RW
020000 E RW
024000 E RW
028000 E RW
```

Arrow to Clock Source Field	- Clock Source can be Emulator or Target system
Toggle space bar from Emulator to Target system back to Emulator	- Toggle between Emulator and Target system as the Clock Source

Use space bar to toggle the fields for options	- Using the space bar will toggle the available selections for each option
No changes are needed for this demo	
"ESC"	- To delete the window

5.3.8 The Interface Window

The interface window shows the current configuration of the Main and Auxiliary RS232 ports.

"HOME"	- To view available windows
Arrow down to where Interface is highlighted	- Arrow to the Interface Window
Interface Window <cr>	- Select the Interface window by pressing return

.Idling.....

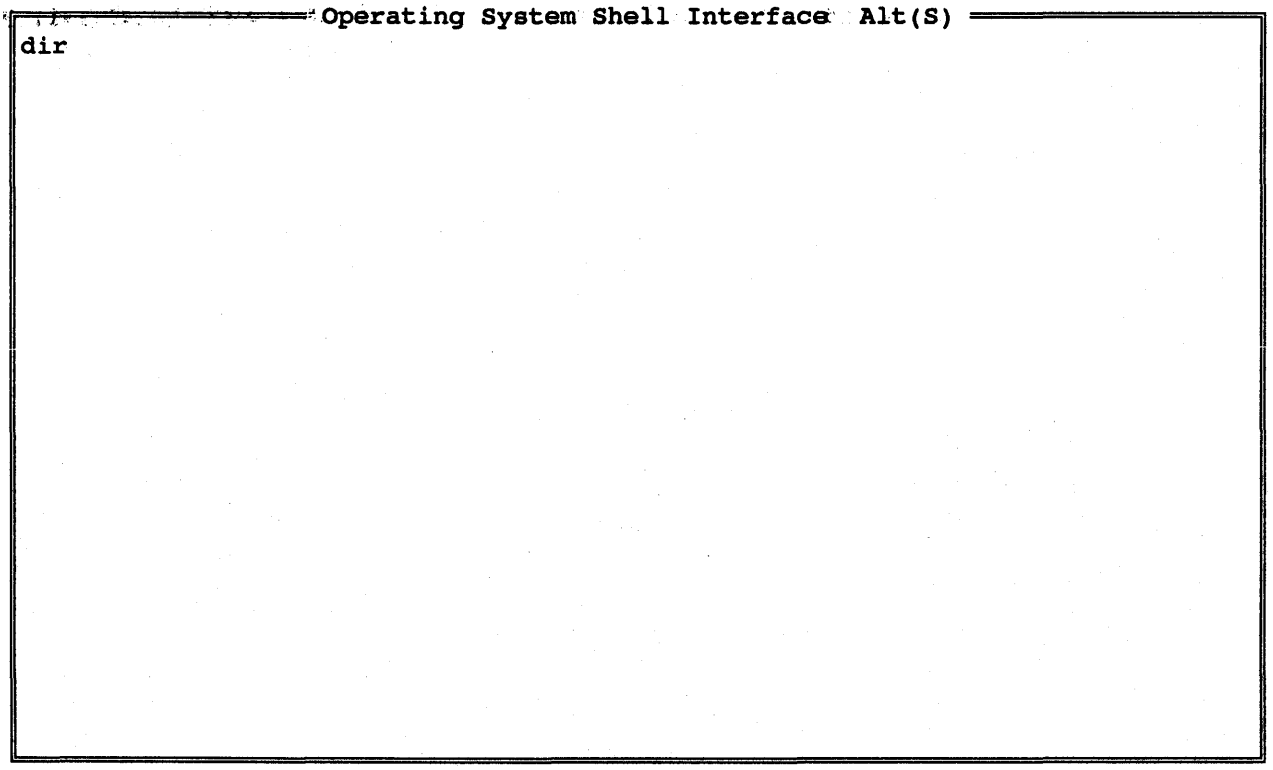
Interface Menu				Trace Print Format	
MAIN		AUXILIARY			
Autobaud ?	YES DCE		DCE	Include Headers	YES
Baud rate	9600	Baud rate	9600	Page Length	066
Data bits	8	Data bits	8	Form Length	056
Parity	NONE	Parity	NONE		
Stop bits	2	Stop bits	2		
Protocol	XON/XOFF	Protocol	CTS/RTS		

Use space bar to toggle the field for options	- Using the space bar will toggle the available selections for each option
No changes are needed for this demo	
"ESC"	- To delete the window

5.3.9 The Operating System Shell Window

The operating system shell interface allows operating system commands to be executed without leaving SourceGate.

"HOME"	- To view available windows
Arrow down to where Shell is highlighted	- Arrow to Operating System shell
Shell <cr>	- Select the Operating System shell
dir	- Enter the DOS "DIR" command to be executed

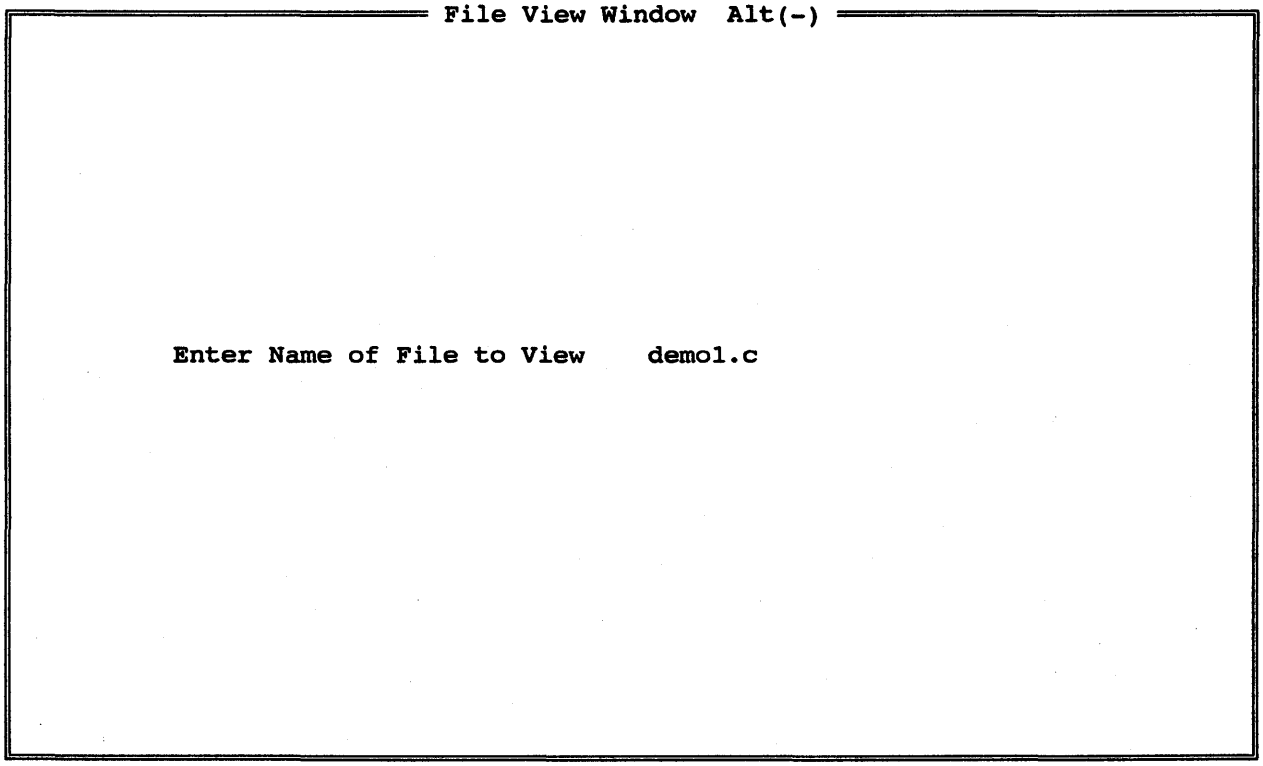


Strike any key to return to the Shell	- Go back to the Operating System shell window
"ESC"	- To delete the Window

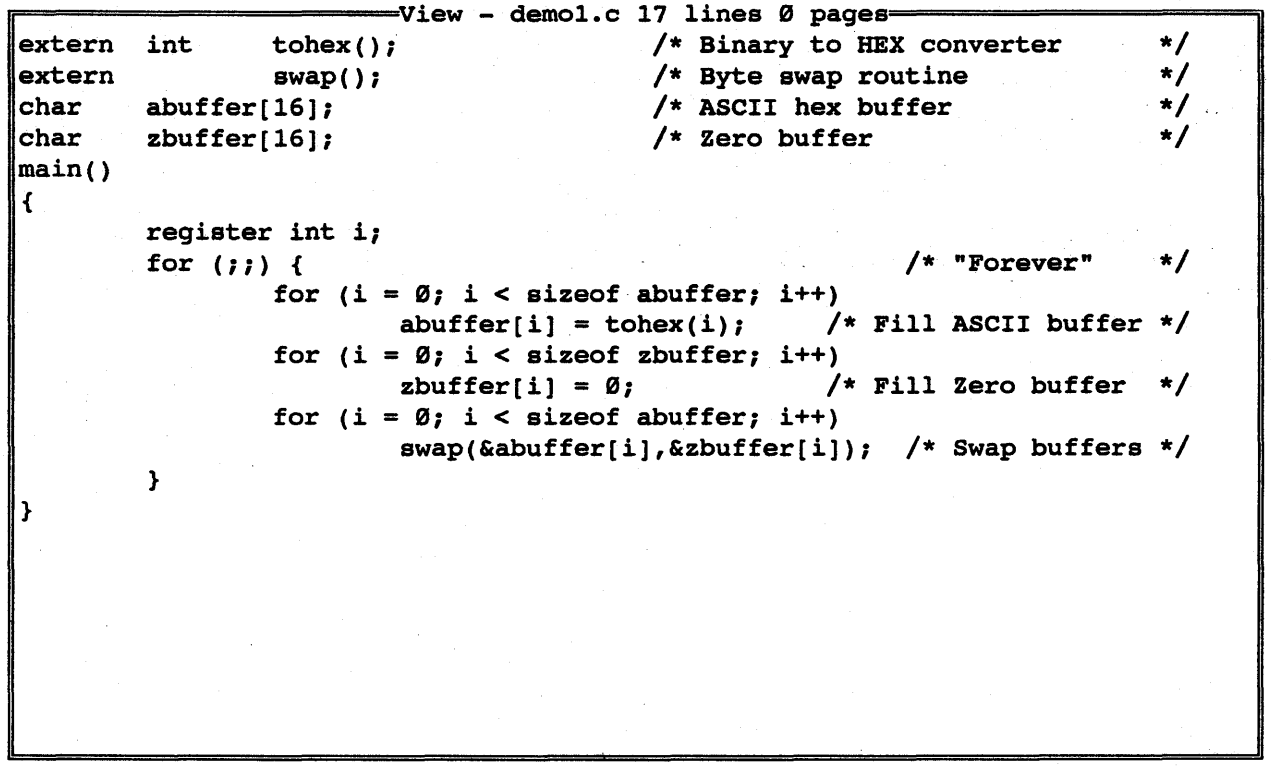
5.3.10 The View Window

The View Window is used to view selected files without leaving the emulator.

"HOME"	- To view available windows
Arrow down to where View is highlighted	- Arrow to View Window
View <cr>	- Select the View Window
Demo1.c	- Enter the name of file to view.



<cr> - Display contents of Demo1.c

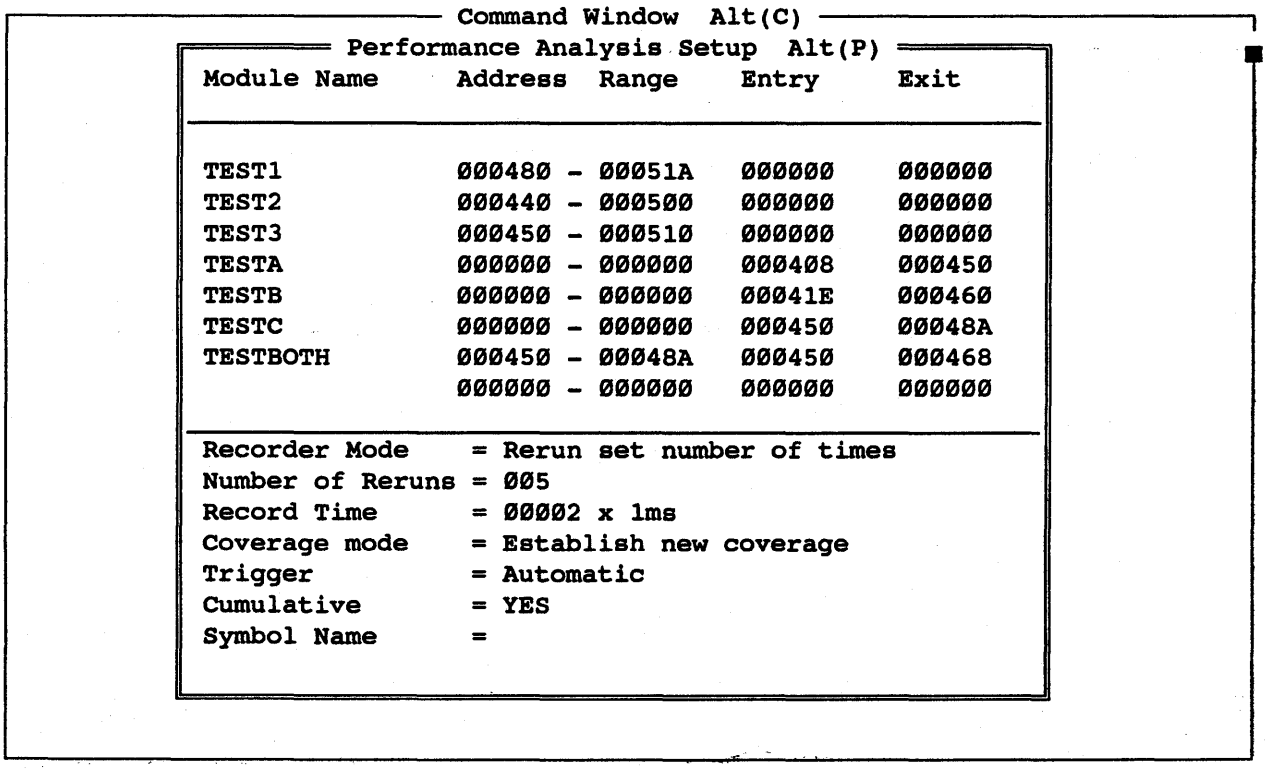


"ESC" - To delete the window

5.3.11 The Performance Windows

The Performance Analyzer is optional and the examples in this section will not function correctly unless you have the Performance Analysis card installed. However, screens with dummy values can be observed by following the steps in this section.

"HOME"		- To view available windows
Arrow down to where Performance is highlighted		- Arrow to Performance Window
Performance Window	<cr>	- Select the Performance Window
Setup <cr>		- Select the Setup option



The following were defined with the demo.bat program:

Module	Addr	Range	- Sets up the address-ranges for modules: Test1, Test2 and Test3
Test1	480	51A	
Test2	440	500	
Test3	450	510	

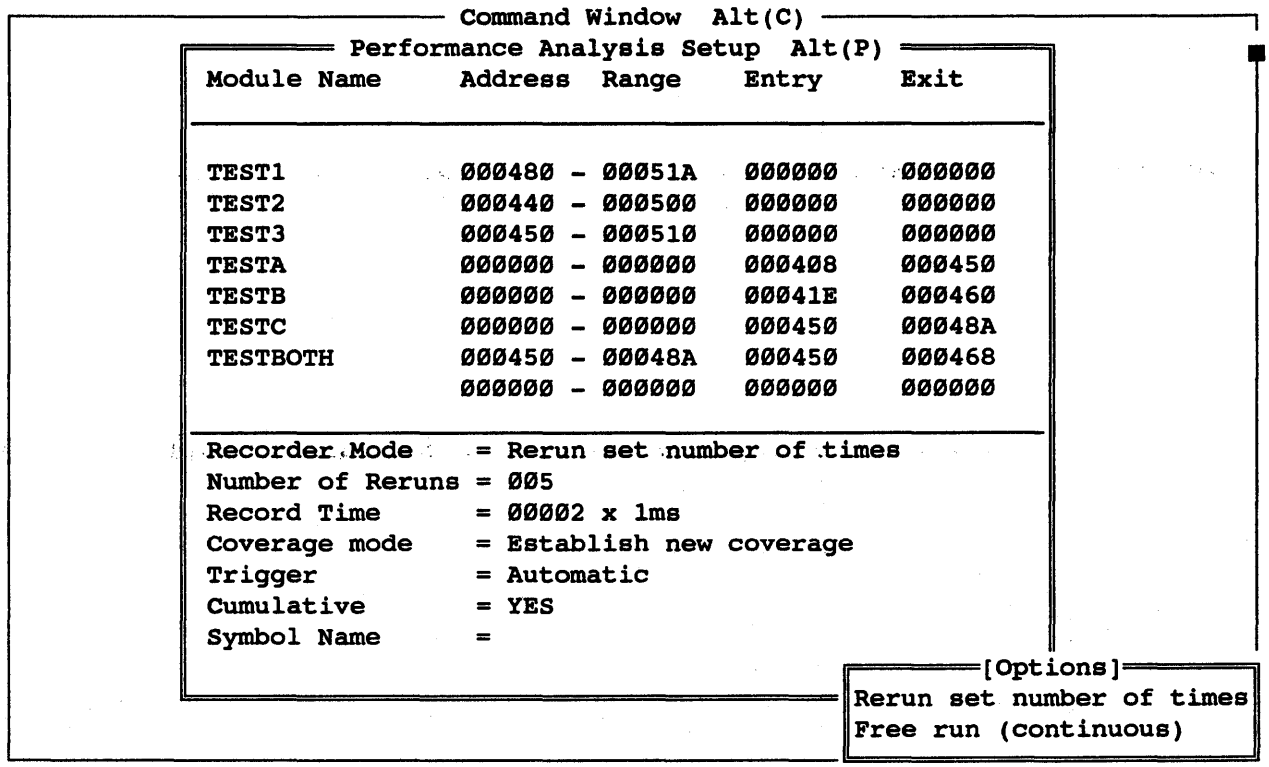
Module	Entry	Exit	- Sets up the entry-exit points for modules: TestA, TestB and TestC. Code must enter and exit at defined points to count as an execution
TestA	408	450	
TestB	41E	460	
TestC	450	48A	

Module	Addr	Range	- Sets up the addr and range for module Testboth. This defines range outside of entry-exit to be counted.
Testboth	450	48A	

Module	Entry	Exit	- Sets up the entry and exit points for Testboth. Code must be enter and exit at defined points, in addition to being in the address-range defined above, to count as an execution
Testboth	450	468	

Arrow down to Recorder mode - Arrow to Recorder Mode

<cr> for options - View the options



Arrow up to Rerun set number of times <cr> - Select the rerun set number of times option

Arrow down to Number of reruns - Arrow to Number of reruns

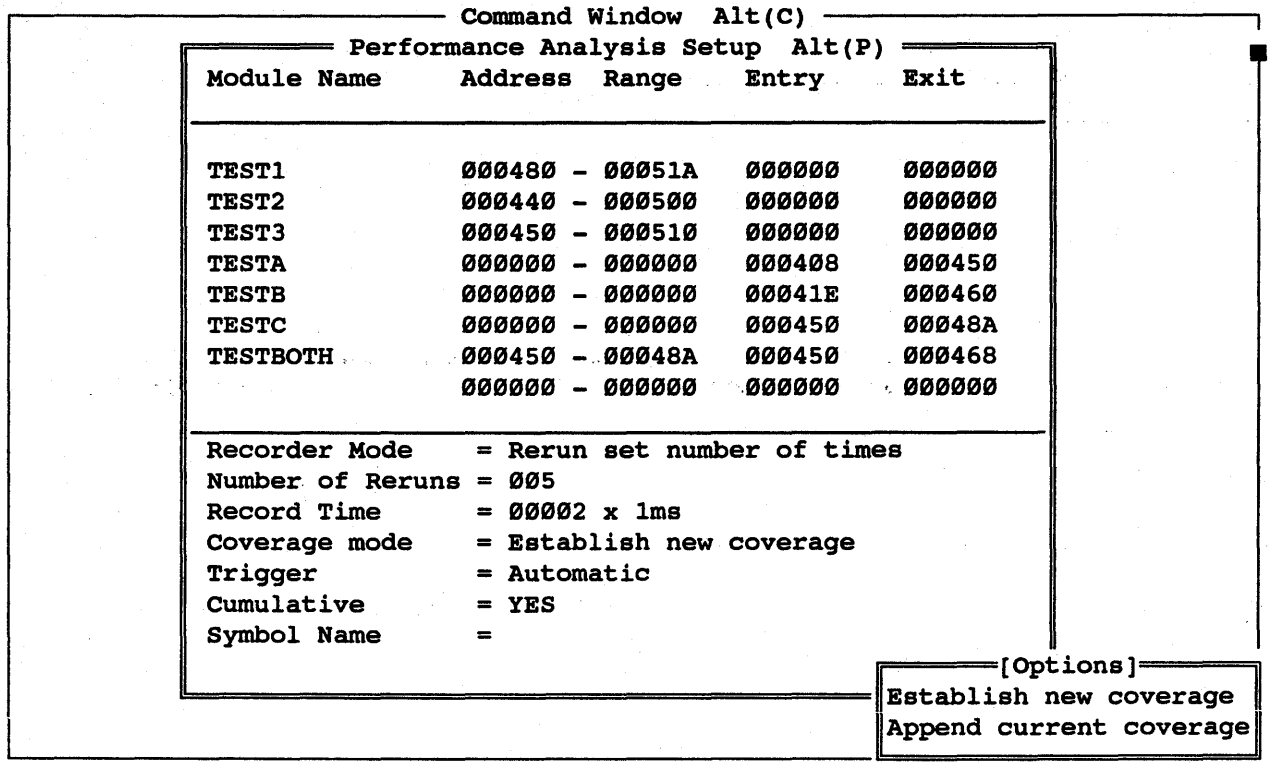
5 <cr> - Set the number of reruns to be 5

Arrow down to Record time - Arrow to record time option

2 <cr> - Set the record time multiplier to be 2 times 1ms

Arrow down to Coverage Mode - Arrow to coverage mode option

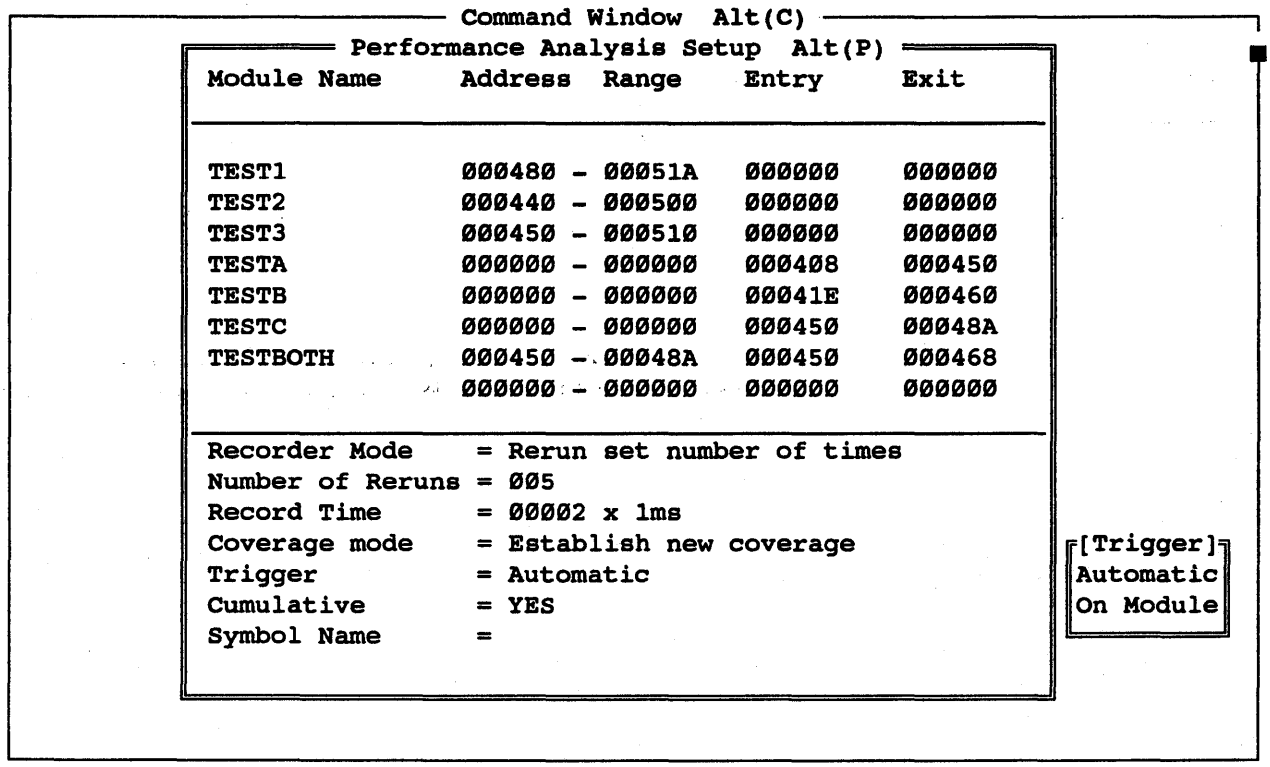
<cr> for available options - View the options



Arrow up to Establish New Coverage <cr> - Set the coverage mode with establish new coverage

Arrow down to Trigger - Arrow to the Trigger field

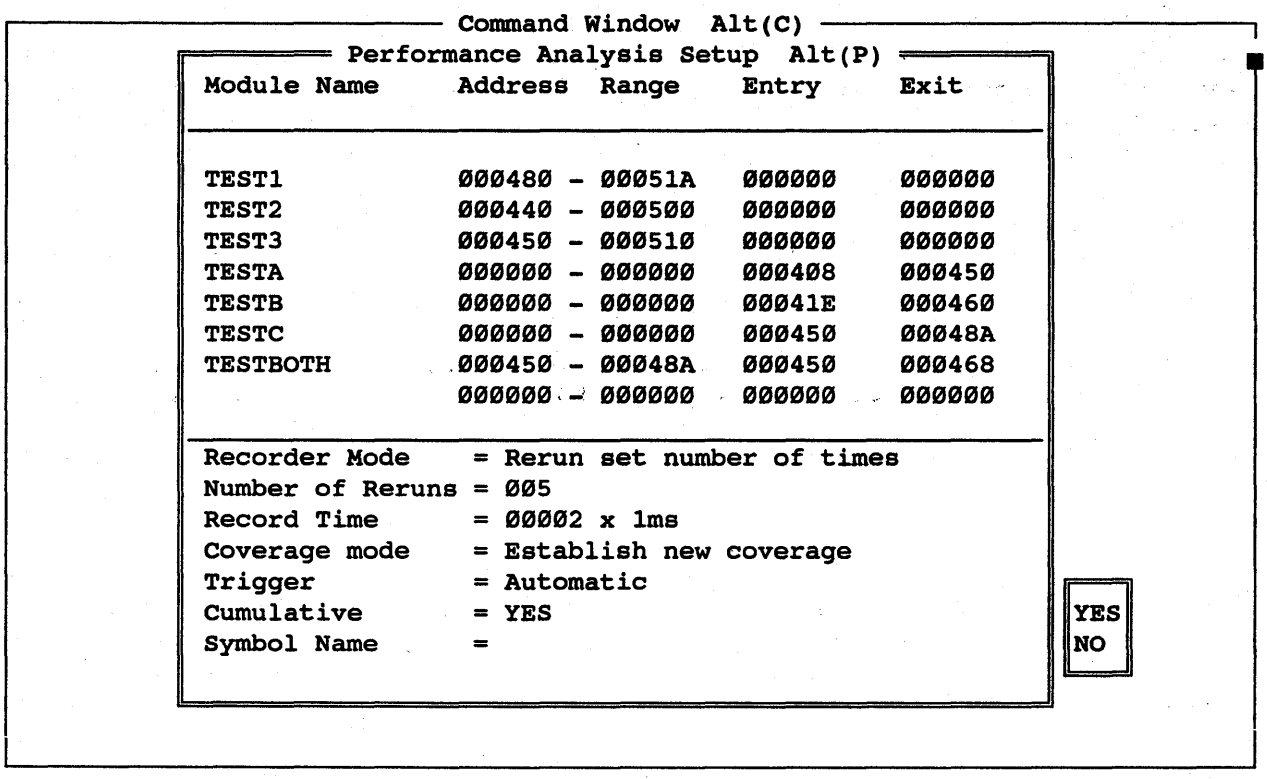
<cr> for available options - View the options



Arrow up to Automatic <cr> - Set this field to be automatic which tells the performance analyzer to begin recording when a defined module is encountered

Arrow down to Cumulative - Arrow to Cummulative Field

<cr> for available options - View the options

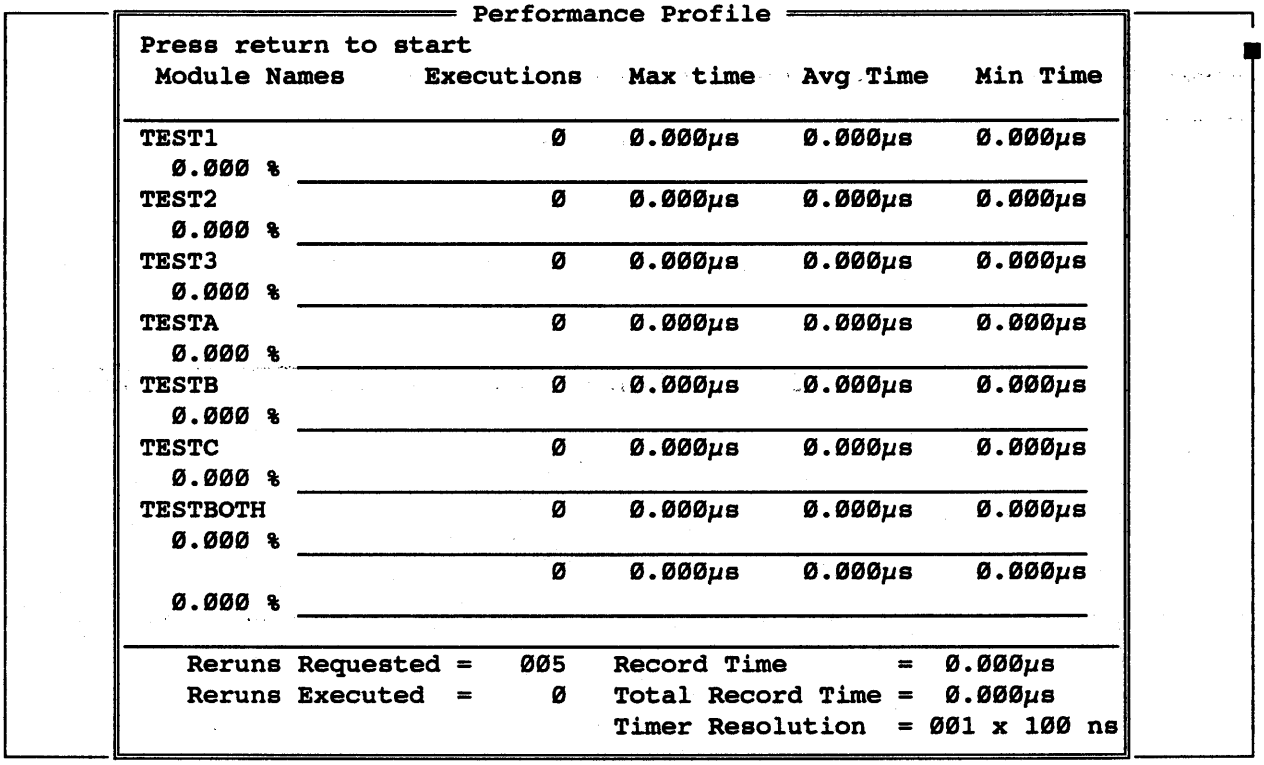


Arrow up to YES <cr> - The module exection times will be added together and the values in the Profile window will reflect this

"HOME" - To view available menus under the Performance Menu

Arrow down to Profile - Arrow to profile option

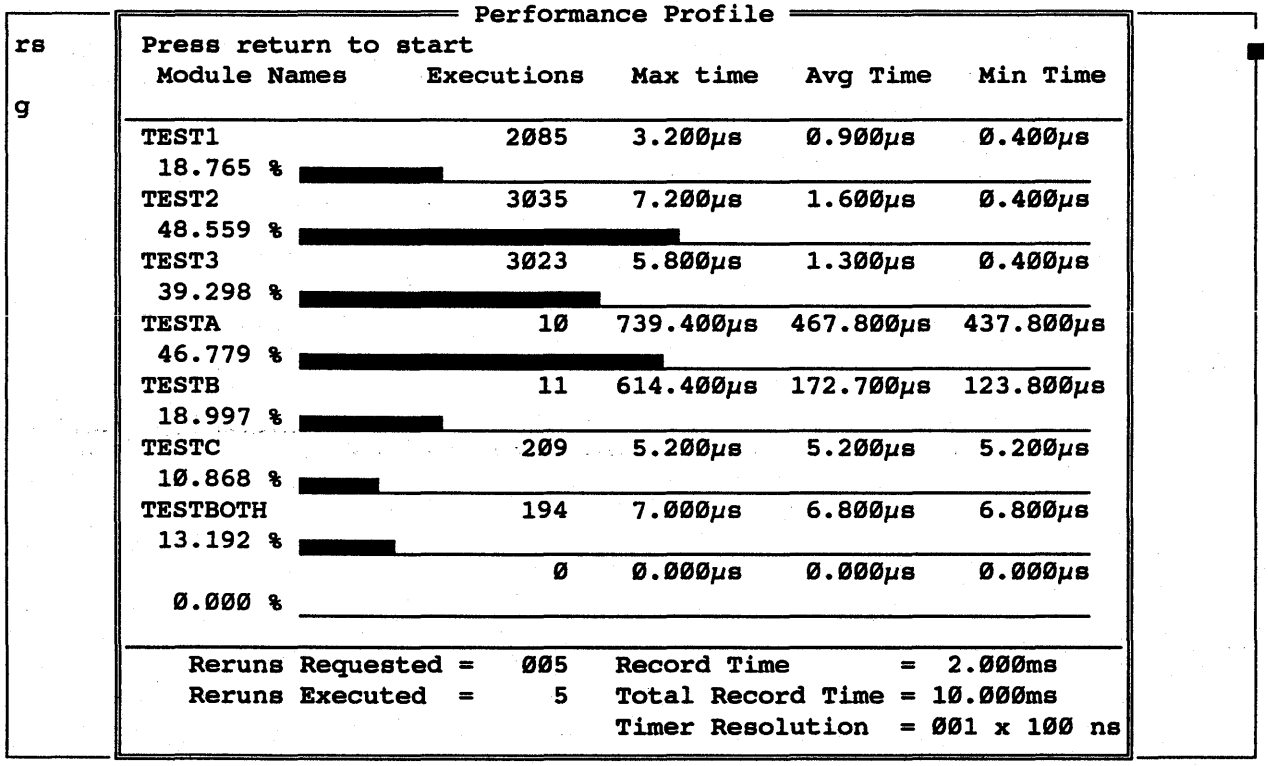
Profile <cr> - Select the Performance Profile



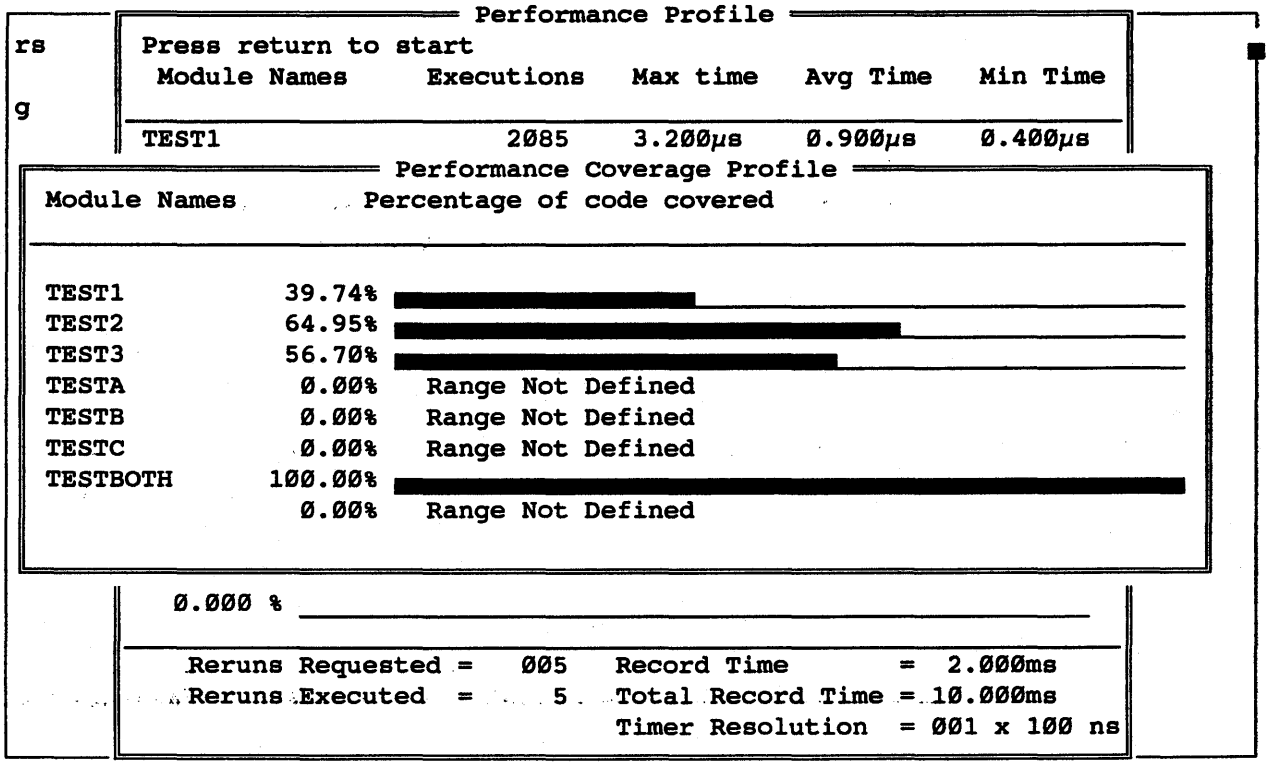
"HOME" - To view available performance windows

Arrow down to where Command is highlighted	- Arrow to Performance Command Window
Command <cr>	- Select the command menu
rs	- Reset the emulator
g <cr>	- Start the emulator
"CNTRL-PGUP"	- To go back to Profile window
<cr>	- Starts recording the Performance Profile

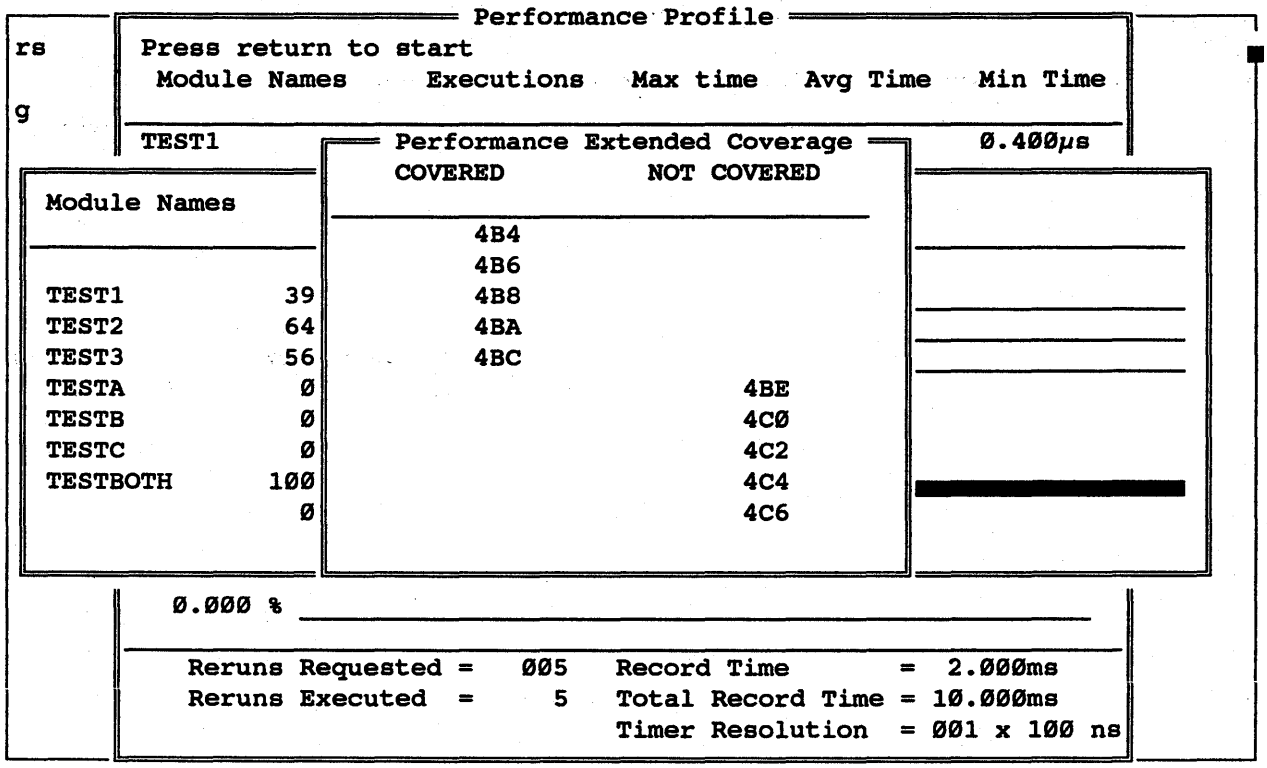
Once the Performance session has been stopped, the other Performance windows can be viewed:



"HOME"	- To view available performance windows
Arrow down to where Coverage is highlighted	- Arrow to coverage window
Coverage <cr>	- To view the % covered in each module



<cr> - The module selected in the coverage mode will be shown in this window with the covered and not-covered addresses



"ESC" - To delete the windows

5.4 Exiting SourceGate

The parameters that have been defined will be saved if an exit key is executed. The parameters will not be saved if quit is used.

"ALT C"	- Go to the command window
exit <cr>	- This will exit SourceGate

APPENDICES

Appendix A Error Messages

A.Ø Introduction

These are the error messages which SourceGate will display in the status window. To clear the error message, press the "ESC" key. The errors have been defined in categories for easy reference.

A.1 Communication Errors

The main port of the emulator should be plugged into the COM1, COM2, or AUX RS232 port of the IBM PC and plugged into a wall socket. The emulator will display a green LED when it is powered on. The following are conditions when the emulator will not completely power up.

A.1.1 Emulator has timed out

Error:

The Communications Software has timed out waiting for a response from the emulator. Please check the serial link or try resetting the emulator. Hit any key to try again.

Response:

The emulator is not responding. Make sure the emulator is plugged to the RS232 connector of the PC correctly and that it is powered on. Press reset on the emulator and hit return.

A.1.2 System not initialized

Error:

System was not initialized on power up

Response:

If the emulator is not responding, hit the emulator reset switch 5 times to re-initialize the system again.

A.2 Debug Problems

These are some common debug problems that will show up when doing a debug session.

A.2.1 No Line Numbers

Problem:

Module: <name> contains no line number information

Response:

Line numbers have not been defined when the source level debugger was initiated. Look at the line number format in the source file and then restart SourceGate with the appropriate -LL_%u format.

A.2.2 Symbols not found

Problem:

Symbol not found in symbol table

Response:

Read the file which contains the symbols into the emulator and establish the symbol information by restarting SourceGate with the appropriate -LL_%u format. If symbols are not part of the file then define the symbols after entering SourceGate.

Example: :test1=0429H.

A.2.3 System Hangs

Problem:

System seems to hang and no key will respond.

Response:

First, try using "CNTRL" Break to abort SourceGate. If there is no response reboot the computer and then hit the reset switch 5 times to re-initialize the system, then restart SourceGate.

A.2.4 Usage Error

Problem:

RD Usage Error

Response:

When doing a RD command, the command needs to have a comma inserted between the read command and the name of the filename.

For example: rd,demo.

A.2.5 Can't Open <filename> for Downloading

Problem:

The current file that is being read with the rd command is not found by the software.

Response:

The filename doesn't exist as the filename typed in or is in a different directory. Re-enter using the correct filename along with the correct drive and path.

A.2.6 Performance Board Not Installed

Problem:

The Performance Analysis software can't find the performance analysis board when trying to do a Performance Profile.

Response:

The Performance Analysis board is an optional card for the emulator. It doesn't come standard with the emulator but can be purchased as an additional item.

Appendix B Using Compilers and Assemblers

SourceGate is compatible with a number of downloadable file formats. SourceGate can load the code, data, symbolic, and source-level debugging information (line numbers) from those formats that supply such information.

SourceGate supports the following formats (those that do not produce symbol information (including line numbers) are so noted):

Motorola S-Record (No symbols)
Intel Hex (No symbols)
Tekhex (No symbols)
HMI Binary (No symbols)
Extended Tekhex
Microtec Research Absolute
Hitachi

Most compiler/assembler manufacturers provide linkers that can produce one of the above formats. Other formats can be converted to Extended Tekhex with the object file convertors supplied with your SourceGate software. These converters are outlined in the following paragraphs (Also see the Read.me file supplied on the converter disk).

Supplied on the converter disk are executable (.EXE or .COM) and documentation (.DOC) files for each converter listed below. Please read the applicable documentation file for the converter you wish to use **before** running the converter. The documentation files have been written to provide insight on how to use the converter programs properly.

INDEX TO THE FILE CONVERTERS DISK

180AD2XT - 2500 A.D. 64180 symbol table file to XTEKHEX file converter.

2500TOXT - 2500 A.D. standard symbol table to Extended TEKHEX file converter.

68KAD2XT - 2500 A.D. MC68000 symbol table file to Extended TEKHEX file converter

AATOXT - American Automation hex file to Extended TEKHEX file converter.

ATOXT - UNIX A.OUT file to Extended TEKHEX file converter.

AVOXTEK - AVOCET plain dump symbol file to Extended TEKHEX file converter.

DBGXTEK - Manx debug (DBG) file to Extended TEKHEX file converter.

HTOXT - HITACHI hex file to Extended TEKHEX file converter.

ITOB - INTEL hex file to BINARY file converter.

ITOM - INTEL hex file to MOTOROLA 'S' record file converter.

MTTOXT - MICROTEK symbol table to Extended TEKHEX file converter.

OBJCNV - INTEL Object file converter.

(The Intel Object file converter operates on an Intel object format file produced by the Intel programs LOCATE, RL51, and/or RL96. It produces as its output a standard Intel format HEX file, and a standard HMI format symbol file.)

QUEXTEK - QUELO linker report file to Extended TEKHEX file converter.

UTILXTEK - Intermetrics (Whitesmiths) Object file to Extended TEKHEX file converter.

INDEX TO THE FILE CONVERTERS DISK (Cont.)

ZTOXT - INTEL hex file/ZILOG symbol file to Extended TEKHEX file converter.

IAR2XT - IAR (Archemedies) UBROF Debug file to Extended TEKHEX file converter.

COFF2XT - COFF file (Introl, Sierra, Intermetrics, SUN386, and PG52x0 COFF) to Extended TEKHEX file converter.

IEEETOXT - IEEE 695 Object format file to Extended TEKHEX file converter.

INDEX

USING COMPILERS AND ASSEMBLERS

HMI emulators are compatible with a number of downloadable file formats. The emulator can load the code, data, symbolic, and source-level debugging information (line numbers) from those formats that supply such information. However, source-level or symbolic communication software (SourceGate or HMI-100/HMI-110) is required for those debugging options.

The emulator supports the following formats (those that do not produce symbol information (including line numbers) are so noted):

Motorola S-Record (No symbols)
Intel Hex (No symbols)
Tekhex (No symbols)
HMI Binary (No symbols)
Extended Tekhex
Microtec Research Absolute
Hitachi

Most compiler/assembler manufacturers provide linkers that can produce one of the above formats. Other formats can be converted to Extended Tekhex with the object file convertors supplied with your emulator. These converters are outlined in the following paragraphs (Also see the Read.me file supplied on the converter disk).

Supplied on the converter disk are executable (.EXE or .COM) and documentation (.DOC) files for each converter listed below. Please read the applicable documentation file for the converter you wish to use before running the converter. The documentation files have been written to provide insight on how to use the converter programs properly.

INDEX TO THE FILE CONVERTERS DISK

180AD2XT - 2500 A.D. 64180 symbol table file to XTEKHEX file converter.

2500TOXT - 2500 A.D. standard symbol table to Extended TEKHEX file converter.

68KAD2XT - 2500 A.D. MC68000 symbol table file to Extended TEKHEX file converter.

AATOXT - American Automation hex file to Extended TEKHEX file converter.

ATOXT - UNIX A.OUT file to Extended TEKHEX file converter.

AVOXTEK - AVOCET plain dump symbol file to Extended TEKHEX file converter.

DBGXTEK - Manx debug (DBG) file to Extended TEKHEX file converter.

HTOXT - HITACHI hex file to Extended TEKHEX file converter.

ITOB - INTEL hex file to BINARY file converter.

ITOM - INTEL hex file to MOTOROLA 'S' record file converter.

MTTOXT - MICROTEK symbol table to Extended TEKHEX file converter.

OBJCNV - INTEL Object file converter.

(The Intel Object file converter operates on an Intel object format file produced by the Intel programs LOCATE, RL51, and/or RL96. It produces as its output a standard Intel format HEX file, and a standard HMI format symbol file.)

QUEXTEK - QUELO linker report file to Extended TEKHEX file converter.

UTILXTEK - Intermetrics (Whitesmiths) Object file to Extended TEKHEX file converter.

INDEX TO THE FILE CONVERTERS DISK (Cont.)

ZTOXT - INTEL hex file/ZILOG symbol file to Extended TEKHEX file converter.

IAR2XT - IAR (Archimedes) UBROF Debug file to Extended TEKHEX file converter.

COFF2XT - COFF file (Introl, Sierra, Intermetrics, SUN386, and PG52xØ COFF) to Extended TEKHEX file converter.

RELEASE NOTES FOR THE SOURCEGATE DEBUGGER

Huntsville Microsystems Inc.

November 28, 1990

Version 1.43

These release notes document the features of SourceGate version 1.43. It was intended for these release notes to be an addendum to the SourceGate version 1.31 manual and they should be stored in the SourceGate manual for future reference.

The organization of these release notes is as follows:

Chapter 1 - New Features Overview:

This chapter gives a brief overview of the new features found in version 1.43

Chapter 2 - Invoking The Software:

This chapter reviews the new command line switches and environment variables that can be used with this release.

Chapter 3 - New Command Line Commands:

This chapter reviews the new command line commands that have been added in this release.

Chapter 4 - New Windows:

This chapter reviews the new windows that have been added in this release.

Application Note 1 - Using The Performance Analysis System:

This section gives helpful hints to users of the HMI Performance Analysis Card (PAC). A sample PAC session is included.

If you have problems with the software or with these release notes, please contact HMI's Technical Support Group at (205) 881-6005.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Chapter 1 New Features Overview

1.Ø Introduction

SourceGate version 1.43 contains many exciting new features that will further aid the user while debugging his/her project. This is part of a continuing effort to upgrade SourceGate with the features requested by our customers. A brief overview of each of these new features is shown below with more detailed explanations given in the applicable chapters that follow.

1.1 Command Line Switches

Several new switches can be defined when invoking SourceGate. These switches allow the user to select a video mode, define a batch file to be executed upon entering SourceGate, specify an extent for source files, define the computer parallel port to be used for file downloading, specify a default filename to be used for read and write operations, and define the number of pages of EMS memory to be used for symbol storage.

1.2 Environment Variables

Several new environment variables used by SourceGate can be defined using the DOS "SET" command. These include variables that determine where temporary files are stored, the baud rate of the communications link, the "COM" port to be used for the communications link, the line number format for source level debugging, the computer parallel port to be used for file downloading, and the batch file to be executed upon entering SourceGate.

1.3 Command Window Commands

Several new commands can be executed while in the Command Window. These new commands include a quicker single-step, a command to pull up an assembly window listing from a specified address, a command to search the help text for a specific keyword, a command to show the computer's memory configuration, a command to download a file from the parallel port, a command to enable "Pass-Thru" mode between the main and auxiliary RS-232 ports, and a command to create a batch file of Command Window commands. There are also commands which affect the display of other windows. These commands will be covered in the sections dealing with those windows.

1.4 Mouse, EMS Memory, and Parallel Port Support

SourceGate now provides both 2-button and 3-button mouse support. In addition, SourceGate will take advantage of EMS memory for symbol storage. This allows very large symbol files to be loaded. This version of SourceGate also allows downloading of files via the computer's parallel port. The downloading of very large files is significantly faster with this feature. Of course, the emulator being used must have a parallel port installed. Older emulators can be upgraded with a parallel port. Please contact HMI for details.

1.5 SourceCode Interface Window

SourceGate now provides a SourceCode Interface Window which enables the user to set breakpoints on certain lines in the source modules. Assembly windows can be popped up at any time. Modules can be searched for a specific string or symbol. This window provides debugging at the true source level without the need to remember line numbers and the module names associated with them. In essence, the emulator is controlled directly from your source file.

1.6 Source Code Interface Options Window

This window allows the user to define such things as the line number format used by the source compiler, the default source file extension, the floating point format used, and the byte order of variable and pointer types.

1.7 Stack Trace Window

This window allows the user to see the functions pushed onto the stack including auto variables and their values being passed to other functions. It should be noted that this feature is heavily compiler dependent. SourceGate will eventually support this feature with all major compilers providing IEEE or COFF formats. However, in this release, this feature is provided for Archimedes and Software Development Systems (SDS) compilers only.

1.8 Watch Window

Watch Window support has been modified to provide structural information for Archimedes and SDS compilers. The emulator can read files from these compilers/linkers directly without going through a converter. This allows auto variable information for specific functions to be displayed in the Watch Window. Watch variables can be defined manually for other compilers. This enables the user to watch certain locations while stepping through his/her code. Watch windows can be saved to disk and reloaded at any time.

1.9 Performance Analysis Enhancements

For those users who have purchased the optional Performance Analysis Card (PAC), the analyzer can now be programmed to start recording when a specific module is reached. Also, the trace buffers can be programmed to display time tags in relative mode (which is the time recorded since the beginning of emulation) or delta mode (which is the time between each cycle in the trace).

Chapter 2 Invoking The Software

2.0 Introduction

The communication software provides the communication between the emulator and computer system. SourceGate will operate only when an HMI-200 series in-circuit emulator is properly connected to the serial port of the host computer and the emulator is powered on.

A number of options can be defined on the command line invoking SourceGate. Invoking SourceGate with the -H switch will list all available options:

```
═══════════════════════════════SourceGate═══════════════════════════════

Usage: sg [-<Baud>] [-<Port>] [-D<DownLoadPort>] [-L<LineSymbol>]
          [-E<SrcExt>]  [-PLM] [-V<Lines>] [-O<OverlayPages>]
          [-S<SymbolPages>] [<HexFile>]

         Baud: 38400,19200,9600(dflt),4800,2400,1200,600,300,150,110,None
         Port:  AUX (default), COM1, COM2
 DownLoadPort:  PRN (default), LPT1, LPT2, LPT3
   LineSymbol: 'scanf()' string for line number symbols (Default: "L_%u")
        Lines: 25, 43, 50 (Sets screen size in lines)
 OverlayPages: EMS pages for code overlay manager (Default: 16)
  SymbolPages: EMS pages for symbol table storage (Default: All available)
       SrcExt: Source File Extension              (Default: ".c")
     HexFile: Establishes default program filename for RD, W commands

═══════════════════════════════SourceGate═══════════════════════════════
```

Note:

SourceGate allocates a number of temporary files when it is running. To ensure that your system has enough memory, it is recommended that all memory resident programs be removed from memory.

The emulator will run up to 38.4K baud. It has been our experience that the 286 machines will support up to 19.2K baud and that the 386 machines will support up to 38.4K baud. Try lowering the baud rate if the emulator appears to have difficulty transferring the data.

If you have been using the emulator with a terminal and want to switch to using a PC computer, the emulator needs to have a CONFIG command done on the command line of the terminal before going to the PC.

2.1 Initializing the Software

To initialize the software and enter the SourceGate program, the standard command line is formatted as follows:

```
SGXXX [-<Baud>] [-<Port>] [-D<DownLoadPort>] [-L<LineSymbol>]
      [-E<SrcExt>] [-PLM] [-V<Lines>] [-O<OverlayPages>]
      [-S<SymbolPages>] [-B<BatchFile>] [<HexFile>]
```

All command line arguments, as indicated by the square brackets ('[' and ']'), are optional. The "XXX" designator should be replaced according to the SourceGate version being run. For example, SG180 and SG11 are for the 64180 and 68HC11 versions of the program respectively.

The command line arguments are as follows:

-<Baud>	Select communications port baud rate. Allowable selections are: N(No serial port initialization performed), 110, 150, 300, 600, 1200, 2400, 4800, 9600 (default), 19200, and 38400.
-<Port>	Select communications port. SourceGate automatically uses COM1 as its default communications port, unless COM1 is not available, in which case it will use COM2. The -port option will override the default SourceGate port selection and cause SourceGate to use only the port named. Allowable options are: COM1, COM2 and AUX.
-D<DownLoadPort>	Select parallel port for downloading. SourceGate will default to PRN. Available options are LPT1, LPT2, and LPT3.
-L<LineSymbol>	In order to enable source-level debugging, SourceGate must know the relationship between the lines in the source file on disk, and the assembly language code in the emulator. This relationship is defined through the use of Line Number Symbols. The default format for the line number symbols recognized by SourceGate is "L_%u". This is a scanf-formatted string indicating that line number symbols are those symbols which consist of one capital 'L' character, followed by an underscore character and an unsigned decimal number. The unsigned decimal number is the

line number. The value associated with the symbol is the starting address of the assembly-language instruction generated by that line of source code.

If, for example, your compiler produced line number symbols of the format LL_%u, then the -LLL_%u option should be used on the command line which initiates SourceGate. In this example, the -L tells SourceGate that the line number format is LL_%u. This command line switch allows source-level debugging using any compiler that produces line number symbols.

-E<SrcExt> This switch defines the source code extent. The default extent is ".C". When source level or mixed mode debugging is enabled, SourceGate will look for source modules with the defined extent.

-PLM This option should be used to support PL/M source level debug.

-V<Lines> This option sets the screen size in lines. Options are 25 (default), 43 (for EGA), and 50 (for VGA).

-O<OverlayPages> This option selects the number of EMS pages for the code overlay manager. The default is 16. This should be sufficient in most cases.

-S<SymbolPages> This option selects the number of EMS pages for symbol table storage. The default is all EMS memory is used for symbols. The -S- option will not use any EMS memory.

-B<BatchFile> This option defines a batch file to be automatically executed upon entering SourceGate.

<HexFile> This option establishes the default program filename for RD and W commands.

Note: To use SourceGate, the HMI 200 series emulator must be set up for the autobaud mode with 8 data bits and no parity (this is how SourceGate will initialize the COM port).

Examples:

```
SGXXX                   = Start SourceGate with default options
SGXXX -19200            = Set 19200 Baud
SGXXX -COM2             = Use COM2
SGXXX -DLPT2            = Use LPT2 for parallel downloads
SGXXX -LLL_%u           = Set Line Number Symbol
                          format to "LL_%u"
SGXXX -EC96             = Set extent default to ".C96"
SGXXX -PLM              = Enable PL/M debugging
SGXXX -V43              = Enable 43 line EGA mode
SGXXX -O32              = Define 32 EMS pages for overlay manager
SGXXX -S16              = Define 16 EMS pages for symbol storage
SGXXX -BTEST            = Define "TEST.BAT" as batch file to be
                          run when SourceGate is invoked
SGXXX DEMO              = Establish "DEMO.HEX" as default
                          filename for RD and W commands


SGXXX -19200 -COM2 -DLPT2 -LLL_%u -EC96 -PLM -V43 -O32 -S16
      -BTEST DEMO       = All Options used
```

2.2 Environment Variables

Several new environment variables used by SourceGate can be defined using the DOS "SET" command. These variables are defined as follows:

TMP - Environment variable for temporary files created by SourceGate.

HMI_TMP - Environment variable for temporary files created by SourceGate.

HMI_BAUD - Environment variable for baud rate. Options are N(None), 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, and 38400.

HMI_LINE - Environment variable for line number format.

HMI_PORT - Environment variable for communications port. Options are COM1, COM2, and AUX.

HMI_LPT - Environment variable for parallel download port. Options are PRN, LPT1, LPT2, and LPT3.

HMI_STARTUP - Environment variable for batch file to be executed upon entering SourceGate.

Chapter 3 New Command Line Commands

3.Ø Introduction

Several new commands can be entered from the command line in the Command Window.

The following new commands have been added to this release:

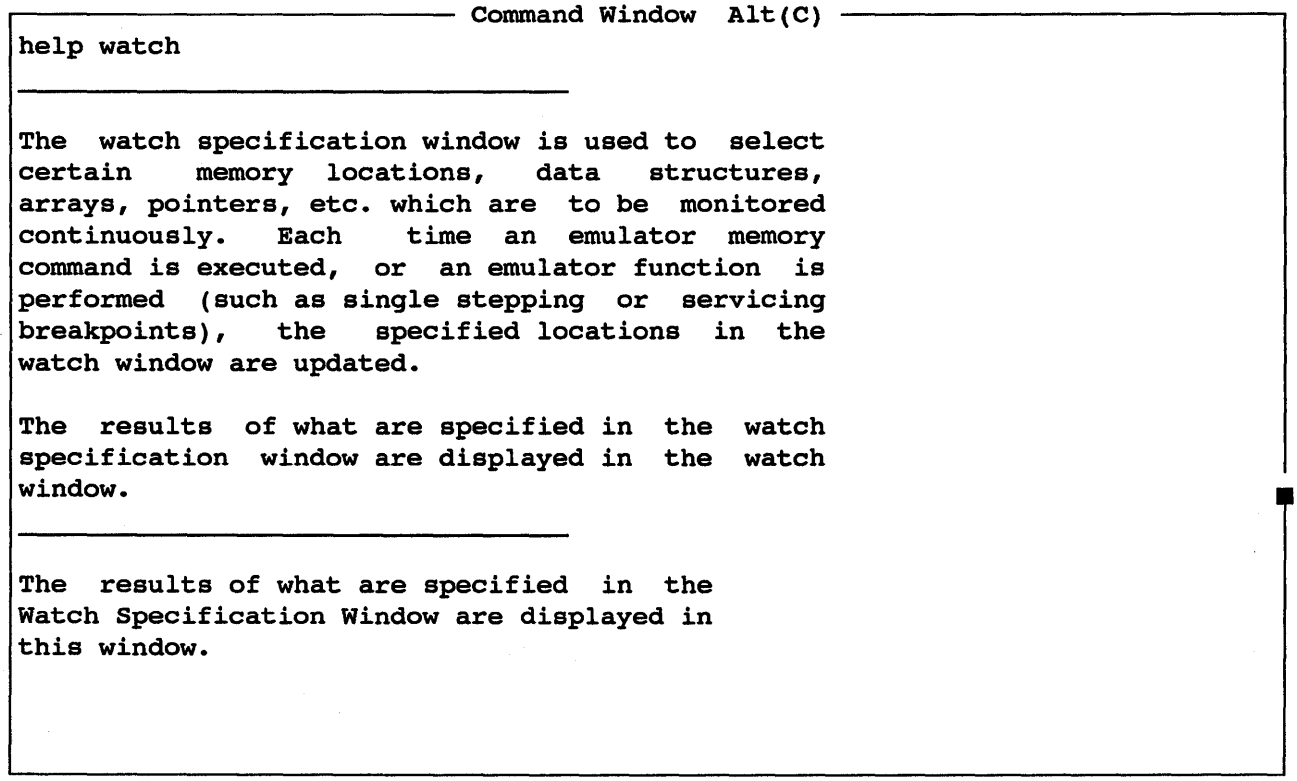
HELP	SEARCH FOR STRING IN HELP TEXT
PT	ENTER PASS-THRU MODE
RDP	READ FILE FROM PARALLEL PORT
SS-Q	SINGLE-STEP QUICK
STM	SHOW MEMORY CONFIGURATION
VASM	PULL UP ASSEMBLY LISTING WINDOW
>	CREATE COMMAND WINDOW BATCH FILE

Some other new commands influence the data displayed in certain windows. Those commands are discussed in the sections dealing with those windows.

3.1 Description of Commands

The commands listed in section 3.Ø are described in detail in the following sections with examples on how to use them. By pressing F1 in the command window and then pressing <cr> on the desired command, additional help information will be displayed in SourceGate.

3.1.1 HELP (Search For String In Help Text) Command



This command will search the help text files for the specified string. The message "Stand-By Searching" will be displayed while the search is being conducted. When a match is found, paragraphs containing the string will be shown with the string highlighted and searching will pause. A <cr> will resume the search. The ESC key will end the search.

The format for the HELP command is:

HELP (String)

where:

String - is the text string to be searched for

3.1.2 PT (Enter Pass-Thru Mode) Command

The Pass-Thru command is used to simulate a wire connecting the emulator's Main and Auxiliary RS-232 ports together. This allows the device (terminal, computer, etc.) connected to the emulator's Main port to control the device connected to the emulator's Auxiliary port.

When the PT command is given, the emulator responds with the following:

```
Please enter a string of characters to be used to terminate Pass-Thru mode
(Use ^W to recall the last string used)
:
```

The emulator will now accept a string of up to eighty displayable ASCII characters. A <cr> terminates the entry of the string. Pass-Thru mode will then commence.

Once Pass-Thru mode is activated, any characters typed at the keyboard of the device connected to the Main port will "pass-thru" the emulator to the Auxiliary port and on to the device connected there. Likewise, any characters received at the emulator's Auxiliary port will be passed through to the Main port. The emulator is doing its best to simulate a wire connecting the two serial ports together.

The emulator is also monitoring all of the characters entered through the Main serial port. When a string of characters is received which exactly matches the previously defined termination string, Pass-Thru mode is canceled and the emulator returns to command mode.

It should be noted that the termination string is also sent out the Auxiliary port as it is entered. When the pass-thru mode terminates, a carriage return character is sent out the Auxiliary port also. Therefore, the chosen termination string should be something which is harmless to the device connected to the Auxiliary port.

3.1.3 RDP (Read File From Parallel Port) Command

```
Command Window  Alt(C)
rdp,multifun.out
Reading multifun.out 104 records read, transfer 100% complete ...Symbols
```

The RDP command is similar to the RD command except that the file is downloaded to the parallel port on the back of the emulator instead of the serial port. To use this command, your computer and the emulator must have a parallel port installed. SourceGate looks for the parallel port to be PRN. If you wish to use a different parallel port, this port must be defined from the command line or set as an environment variable. This procedure is discussed in Chapter 2.

The format of this command is:

RDP(Offset),Filename

where:

Offset - is a hexadecimal value representing the address offset

Filename - is a file name, valid to the host computer, that designates the file to be downloaded to the emulator over the parallel connection

3.1.4 SS-Q (Single-Step Quick) Command

```
                    Command Window  Alt(C)
ss -Q

PC=00000190    LINK.W A6,#FFF8
SR=2700 - T0 S I I7 NX PS NZ NV NC
ISP=00001000   MSP=9FBFFA55   USP=FF6FFFFF   VBR=00000000
D0=00002771    D1=F9DFECAB    D2=CFBFB77F    D3=F77F77F7    CAAR=CE7BCAD5
D4=933F8BF4    D5=44EFB5DF    D6=CF69B7F1    D7=5EFFCF7D    CACR=00000000
A0=FFFFFFEF    A1=4BEF775F    A2=F70FEFF7    A3=DBFFDD74    DFC=7
A4=F8DADFFF    A5=6F7F7D77    A6=CFFFFDDF    A7=00001000    SFC=7

PC=00000194    CLR.L (FFFC,A6)
SR=2700 - T0 S I I7 NX PS NZ NV NC
ISP=00000FF4   MSP=9FBFFA55   USP=FF6FFFFF   VBR=00000000
D0=00002771    D1=F9DFECAB    D2=CFBFB77F    D3=F77F77F7    CAAR=CE7BCAD5
D4=933F8BF4    D5=44EFB5DF    D6=CF69B7F1    D7=5EFFCF7D    CACR=00000000
A0=FFFFFFEF    A1=4BEF775F    A2=F70FEFF7    A3=DBFFDD74    DFC=7
A4=F8DADFFF    A5=6F7F7D77    A6=00000FFC    A7=00000FF4    SFC=7
.
```

The Single-Step Quick command has been added to provide the user with a quicker way to step through code. While in this mode, the registers, flags, and the next mnemonic to be executed will be displayed after each step. The registers and flags will display the results of the last executed instruction. All other windows (such as the Register and Watch windows) will not be changed after each step but will be updated when the single-step mode is terminated. Also, all display of source code will be suppressed while in this mode. Quick single-stepping is allowed in assembly language only. The regular single-step commands should be used if the user wishes to step through source code and have all windows updated after each step.

There are three options for operating the Single-Step Quick Command. The first option is to single step every time a carriage return is entered. To terminate this mode, type a period. The second option is to single step a specified number of times. The third option is to single step continuously. To terminate the second and third options, a carriage return should be entered.

The format for single stepping once for each carriage return entered is:

SS-Q

The format for single stepping continuously until a carriage return is entered is:

SSX-Q

The format for single stepping a specified number of times is:

SScount-Q

Where:

count - is the number of iterations to single step

3.1.5 STM (Show Memory Configuration) Command

```
Command Window  Alt(C)
stm
System Memory Used : 132784
System Memory Left : 195760
Overlay Manager Memory : 65536
EMS Total pages 128 Memory 2048K
EMS Used pages 0 Memory 0K
EMS left pages 128 Memory 2048K
```

This command shows the user how the memory is being used in the computer. The following is an explanation of each of these fields:

System Memory Used: 132784

This shows how much of the emulator's system memory is being used (not the computer's memory).

System Memory Left: 195760

This shows how much of the computer's memory is available.

Overlay Manager Memory: 65536

This shows how much of the computer's memory is being used by the Overlay Manager.

EMS Total pages 128 Memory 2048K

This shows how much EMS memory is installed in the host computer in both pages (16K per page) and bytes (pages * 16K).

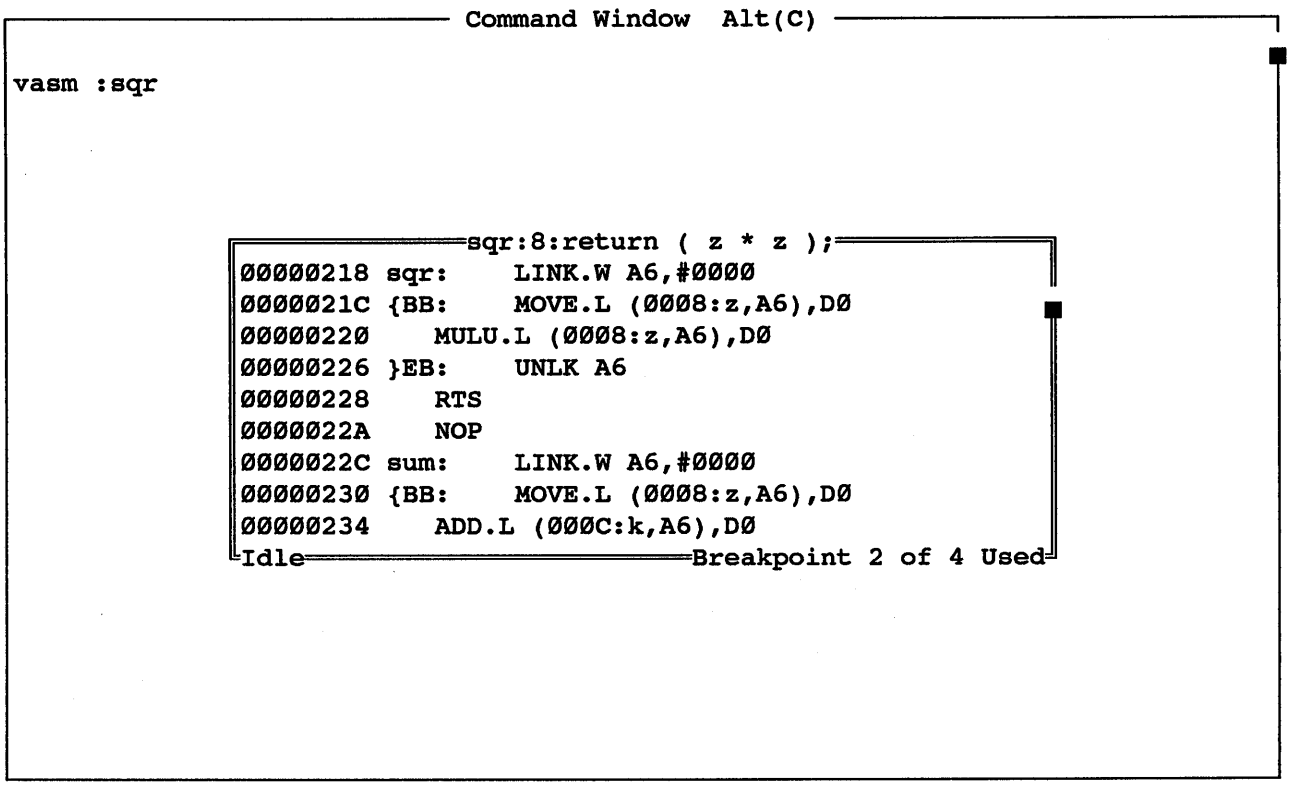
EMS Used pages 0 Memory 0K

This shows how much EMS memory is being used for symbol storage in both pages (16K per page) and bytes (pages * 16K).

EMS left pages 128 Memory 2048K

This shows how much EMS memory is left for symbol storage in both pages (16K per page) and bytes (pages * 16K).

3.1.6 VASM (Pull Up Assembly Listing Window) Command



This command will pull up an assembly window starting at the defined address or symbol. If no address or symbol is given, then the assembly window will start at the current program counter. The current program counter is shown as a line with a different color than those around it.

While in the assembly window, the user may scroll down from the address or symbol displayed at the top of the window. Also, the bottom border of the window will show how many breakpoints have been set. This number will increment each time a new breakpoint is entered. The current status of the emulator is shown in the bottom left-hand corner of this window also.

There are several commands that can be executed from this window. A summary of these commands is listed below:

B - This command will set a breakpoint on the highlighted line. When moving the cursor off this line, the user will notice that the line is shown on a dark background. Notice also that the bottom border of the window reflects how many breakpoints have been used.

C - This command will clear all breakpoints that have been set in this window.

F - This command will prompt the user to enter a symbol name. The software will then locate the symbol name in the symbol table and will update the assembly window to show code around this symbol.

G - This command will start emulation from the current program counter location. The emulator will start looking for any breakpoints defined with the "B" command.

P - This command will update the assembly window to show code around the current program counter value. If the command "Source On" is entered in the command window, then entering a "P" command will display a SourceCode window, provided the program counter points to a line of source code. If the command "Source Off" is entered in the command window, then entering a "P" command will display an assembly window at all times. Whether or not source is on or off can be determined by entering the "Source" command from the command window.

R - This command will reset the processor to the defined reset vector.

S - This command will stop emulation and all registers and Watch window values will be updated.

/ - This command will prompt the user to enter a string to be searched for. The window will then be searched and the bar will highlight the assembly line containing the string. To search for the next occurrence of the string, "/" should be entered again. The last string entered will be displayed and a <cr> will search for that string again. This search command is case sensitive.

- This command allows the user to move up and down a given number of lines or to go to a specific line in the code. To go up 3 lines, the user should enter "3" and then hit the up arrow key. To go down 3 lines, "3" should be entered followed by the down arrow key. To go to line 3 of the window, "3<cr>" should be entered.

3.1.7 > (Create Command Window Batch File) Command

```
Command Window  Alt(C)
>test
dØ 1f
00000000 0000 2700 0000 0190 0000 000F 0000 0038  ..'.............8
00000010 CA88 4200 0000 002C 0002 E000 1025 240D  ..B....,.....%$.
bm
 Byte mode
dØ 1f
00000000 00 00 27 00 00 00 01 90 00 00 00 0F 00 00 00 38  ..'.............8
00000010 CA 88 42 00 00 00 00 2C 00 02 E0 00 10 25 24 0D  ..B....,.....%$.
wm
 Word mode
>
<test
dØ 1f
00000000 0000 2700 0000 0190 0000 000F 0000 0038  ..'.............8
00000010 CA88 4200 0000 002C 0002 E000 1025 240D  ..B....,.....%$.
bm
 Byte mode
dØ 1f
00000000 00 00 27 00 00 00 01 90 00 00 00 0F 00 00 00 38  ..'.............8
00000010 CA 88 42 00 00 00 00 2C 00 02 E0 00 10 25 24 0D  ..B....,.....%$.
wm
 Word mode
```

This command will create a batch file under the specified name. All commands typed while in the command window will be entered into the batch file. Commands such as dumping memory, filling memory, reading a file, etc. can be automated as part of a batch file. A ">" by itself on a line closes the batch file. The batch file can be executed by entering the command "<filename".

NOTE: Only command line operations are allowed when creating batch files. Other windows should not be entered while in this mode.

4.0 Introduction

This release of SourceGate supports several new windows that make source level debugging even easier. An explanation of how to use these windows is provided in the sections that follow.

4.1 Source Code Interface Options Window

```
──────────────────── Source Code Interface Options ────────────────────
Line Symbol Format ..... L_%u
Default Source Ext ..... .C
Program Counter ........ PC
Stack Pointer .......... SSP
Stack Frame Pointer .... A6
Floating Point Format .. IEEE - 1985

Type            Byte Order

Character .... Ø
Short ........ 1Ø
Integer ...... 321Ø
Long ......... 321Ø              ______Floating Point Format______
Float ........ 321Ø              es[31]e[3Ø,8<127]M[23,22]
Double ....... 76543210          es[63]e[62,11<1Ø23]m[51,52]
Long Double .. 9876543210        es[79]e[78,15<1638]3m[63,64]

Pointers        Near             Far
Address ...... 321Ø              321Ø
Function ..... 321Ø              321Ø
Data ......... 321Ø              321Ø
```

The Source Code Interface Options window allows the user to define compiler and processor specific parameters such as the line number symbols generated by the compiler, the default source extent used by SourceGate, the floating point format, and the byte order for all variable types and pointers. These parameters can be changed by highlighting the desired section and entering changes. The Floating Point Format field will display a selection window that will allow IEEE-1983, IEEE-1985, or the defined format to be used when dealing with floating point numbers. The parameters defined in this window will become the default values used by SourceGate until they are changed again by the user.

To pull up the Source Code Interface Options window, the "HOME" key should be entered and the "Source" option should be selected from the Main Emulation Control Menu.

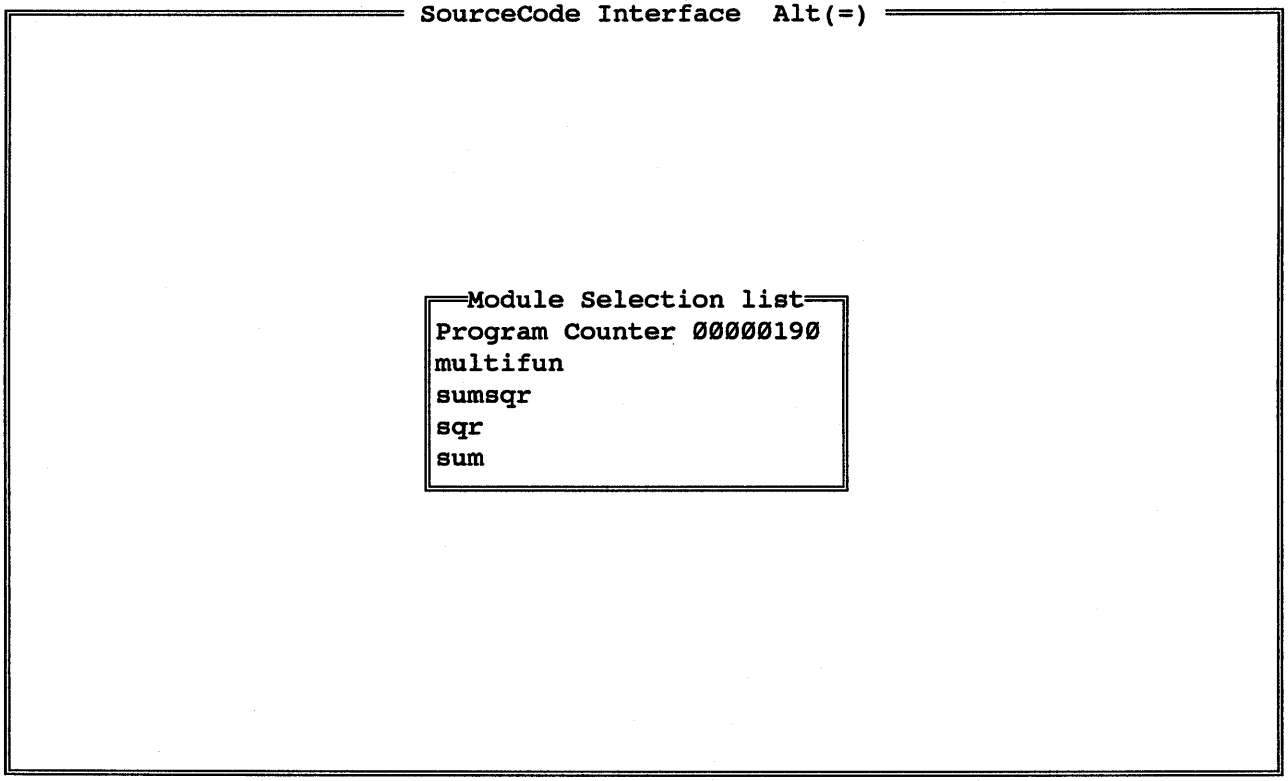
4.2 SourceCode Interface Window

The SourceCode Interface window allows the user to set breakpoints in his/her source code, pull up assembly windows, search the module for a text string or symbol, start and stop emulation, and reset the emulator. In essence, the emulator is controlled directly from your source file.

To pull up the SourceCode window, the "HOME" key should be entered and the "Code" option should be selected from the Main Emulation Control Menu. This window can also be chosen by entering "ALT =". The operation of the SourceCode Interface window is described in the sections that follow.

4.2.1 Source Module Selection

When the user enters the SourceCode window, a Module Selection List will be displayed showing the available source modules along with the current program counter value:

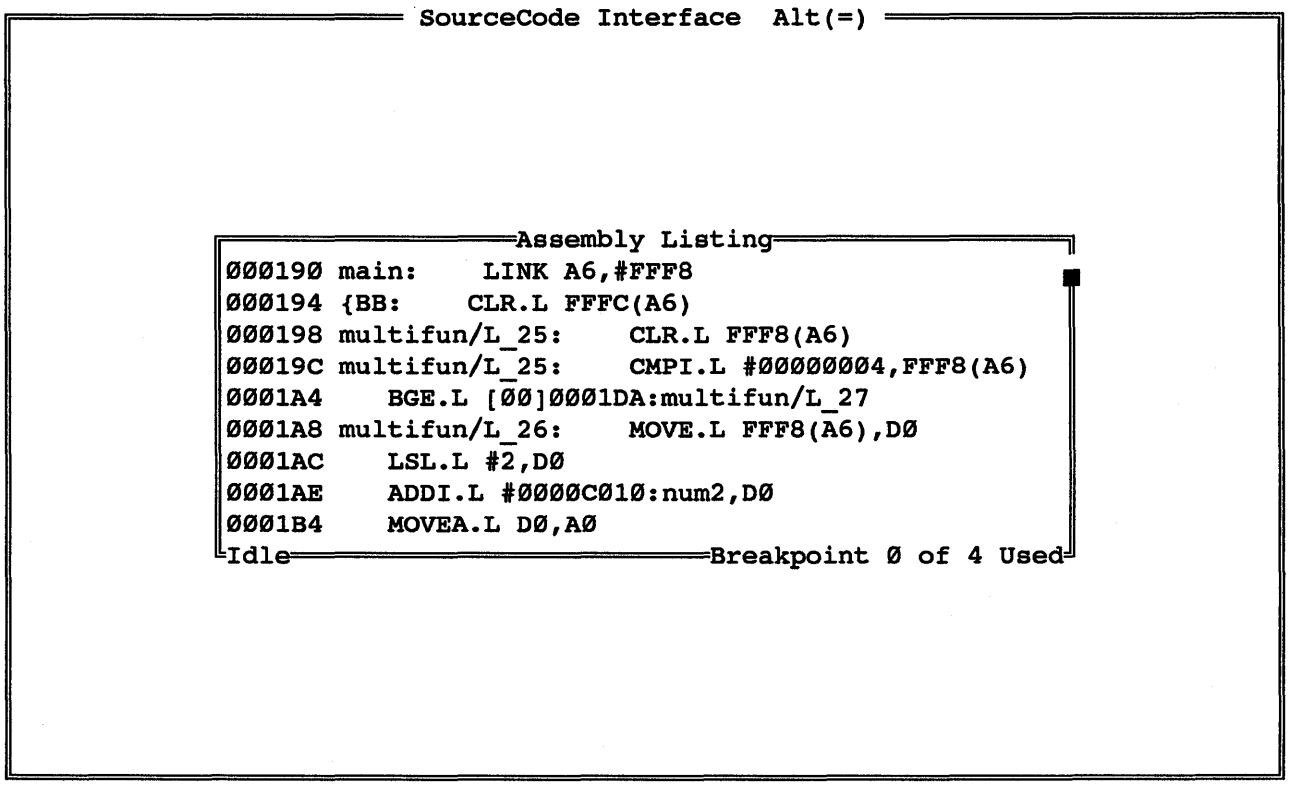


The user can scroll through the Module Selection List by using the up/down arrow keys. The module can also be entered by directly typing its name. SourceGate will move the highlighted bar to the module name entered. The module can then be loaded by entering a <cr>. The message "Stand By-Loading Source Code" will then be displayed. SourceGate will then proceed to load the requested module into the SourceCode window:

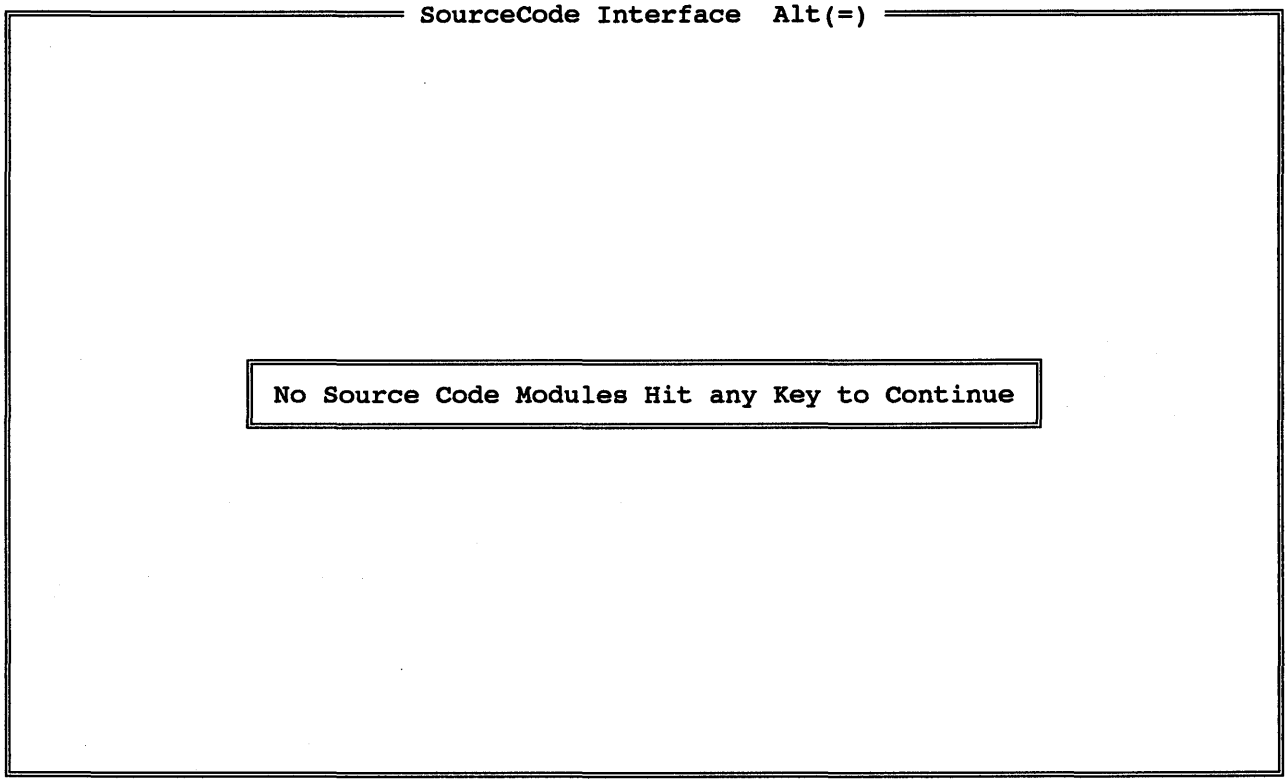
```
SourceCode Module[multifun]=================================lines 1 of 13  Alt(=)
/* multifun.c */
/* tests sumsqr function */
main()
{
   static int num1[4] = { 1, 2, 3, 4};
   static int num2[4] = { 5, 6, 7, 8};
   int sqrofsum, count;

   loop:  sqrofsum = 0;
   for (count=0; count<4; count++)
       sqrofsum = sumsqr(num1[count],num2[count]);
   goto loop;
}









Emulator[Idle] PC = 00000190================================= Breakpoint 0 of 4 Used
```

It should be noted that the borders of this window display information such as the module name, the line of the module that the cursor is on, the status of the emulator (running or idle), the current PC value, and the number of breakpoints used. If the program counter module is selected, then the source module where the PC is located will be loaded. Also, if the program counter is located in the currently displayed module, the line of source code that contains the PC value is shown in a different color.

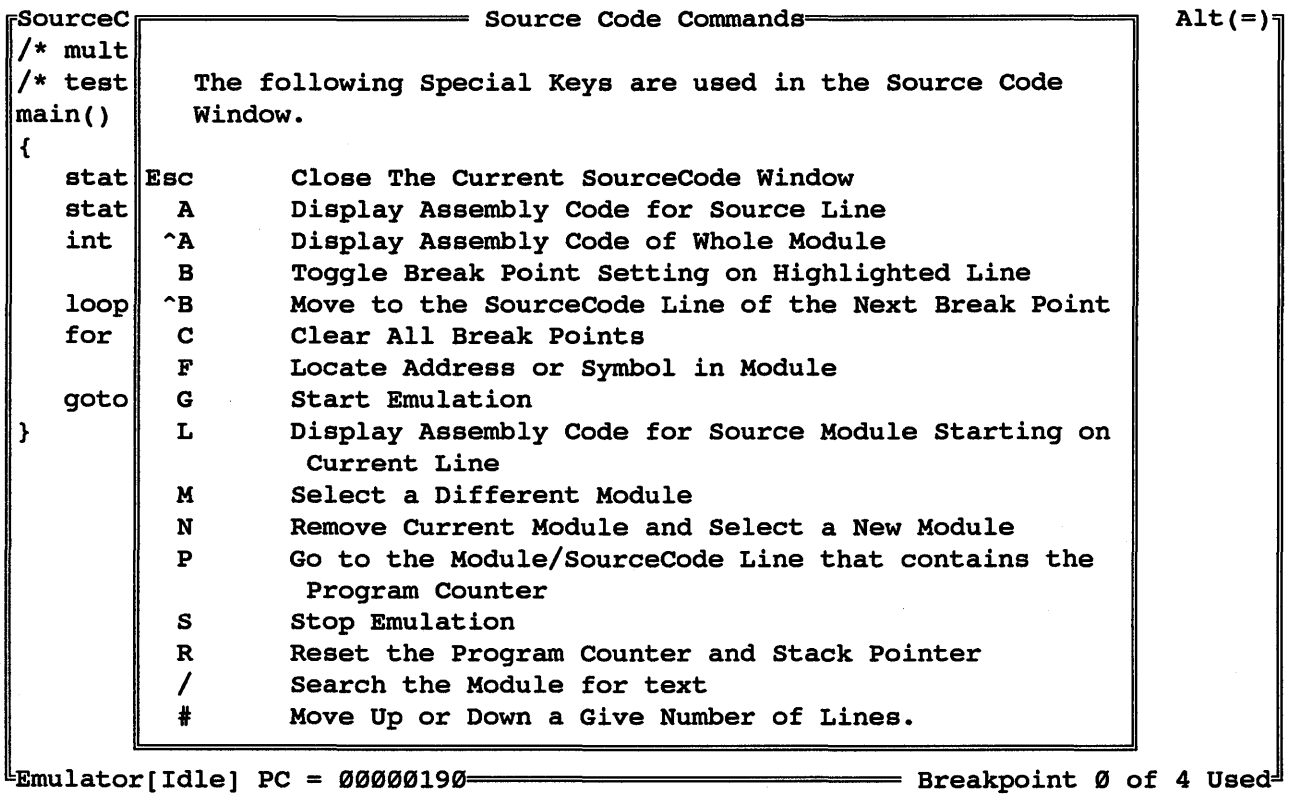
If, however, the PC value is not contained in a source module, an assembly window starting at the PC value will be shown instead:



If source level debugging has not been enabled (using the :=S or :=M commands), or if the file that was loaded did not contain any line number information, then the following message will be displayed when entering the SourceCode Interface window:



Help information can be displayed in the SourceCode window by entering the F1 key:



Additional information can be obtained by highlighting the desired command and entering the F1 key again.

4.2.2 SourceCode Interface Commands

There are several commands that can be executed from the SourceCode Interface window. A summary of these commands follows:

ESC Closes the current SourceCode window. Control will be passed to the window that was opened prior to the one just exited.

A Displays assembly code for the highlighted source line. The source line will be displayed in the top border of the assembly window.

^A Displays assembly code for the whole module. The source line that generated the highlighted assembly code is shown in the top border of the assembly window. The source line shown will be updated as assembly code from other source lines is highlighted.

B Toggles breakpoint setting on the highlighted line. When a line has been selected as a breakpoint, the line will display a dark background and the line itself will take on a different color. Also, the bottom border of the window will be updated to show how many breakpoints have been used. When a breakpoint has been selected on a line, emulation will stop when code related to that line is executed. Setting breakpoints from this window overrides any breakpoints defined in the Sequence window.

^B Moves the highlighted bar to the next defined breakpoint.

C Clears all breakpoints.

F Locates a symbol in the module. A separate window will appear in which the user types the name of the symbol to be found followed by a <cr>. SourceGate will search the current module for the symbol. If it is not found in the current module, other modules will be searched. When a match is found, that module will be loaded and the highlighted bar will be on the specified symbol.

G Starts emulation from the current program counter value and "Running" will be displayed in the bottom border of the window.

L Displays assembly code for the current module starting on the current line.

M Will pull up the Module Selection List. Other modules can be selected at this time. The current module will still be loaded, but its window will be inactive in the background.

N Will remove the current module and pull up the Module Selection List. The current module will be unloaded, and its window will be deleted.

P Will take the highlighted bar to the module and source line that contains the program counter. If the current PC is not contained in a source module, an assembly window starting at the PC value will be shown instead.

R Resets the program counter and loads the module in which the program counter resides.

S Stops emulation and "Idle" will be displayed in the bottom border of the window.

/ Searches the current module for a text string. When a match is found, the highlighted bar will appear on the line of code containing the string. If the defined string is not found, the message "String Not Found" will be displayed in a status window.

Allows the user to move the highlighted bar up or down a specified number of lines or to a specified line. To go up a certain number of lines, the user should enter the desired number followed by the up arrow key. To go down a certain number of lines, the user should enter the desired number followed by the down arrow key. To go to a specific line, the user should enter the desired line number followed by a <cr>.

It should be noted that the above commands can be used while in an assembly window except for A, ^A, L, M, and N.

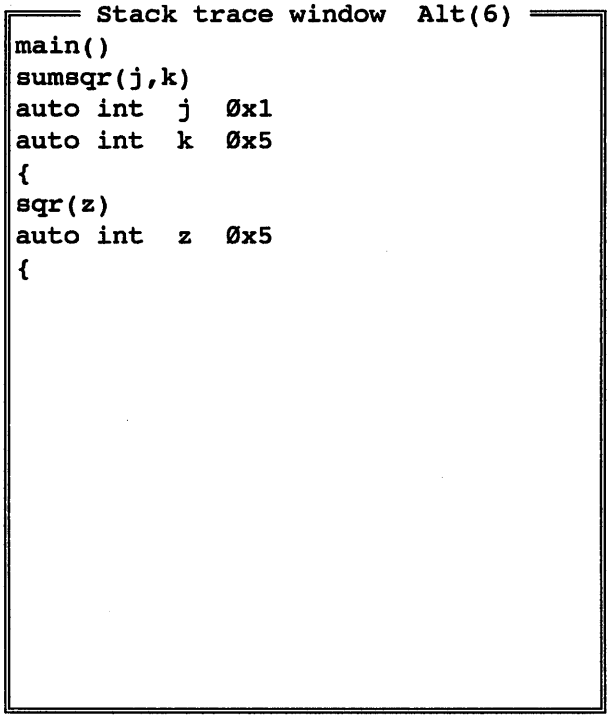
There are two command line commands executed from the Command window that may have an impact on how SourceCode windows are shown.

If the "Source On" command is used (which is the default), any commands executed from an assembly window will cause control to be passed to the applicable source window upon completion of the command provided the program counter points to a line of source code.

For example, if the user set and reached a breakpoint on a line of assembly code in an assembly window, then, if that line of assembly code relates to a source line in a source module, a new SourceCode window would be generated upon reaching this breakpoint and the assembly window would be suppressed.

If the "Source Off" command is used, any commands executed from an assembly window will cause control to be returned to that window upon completion of the command. An assembly window is always generated in this case and SourceCode windows are suppressed, even if the program counter points to a line of source code.

4.3 Stack Trace Window

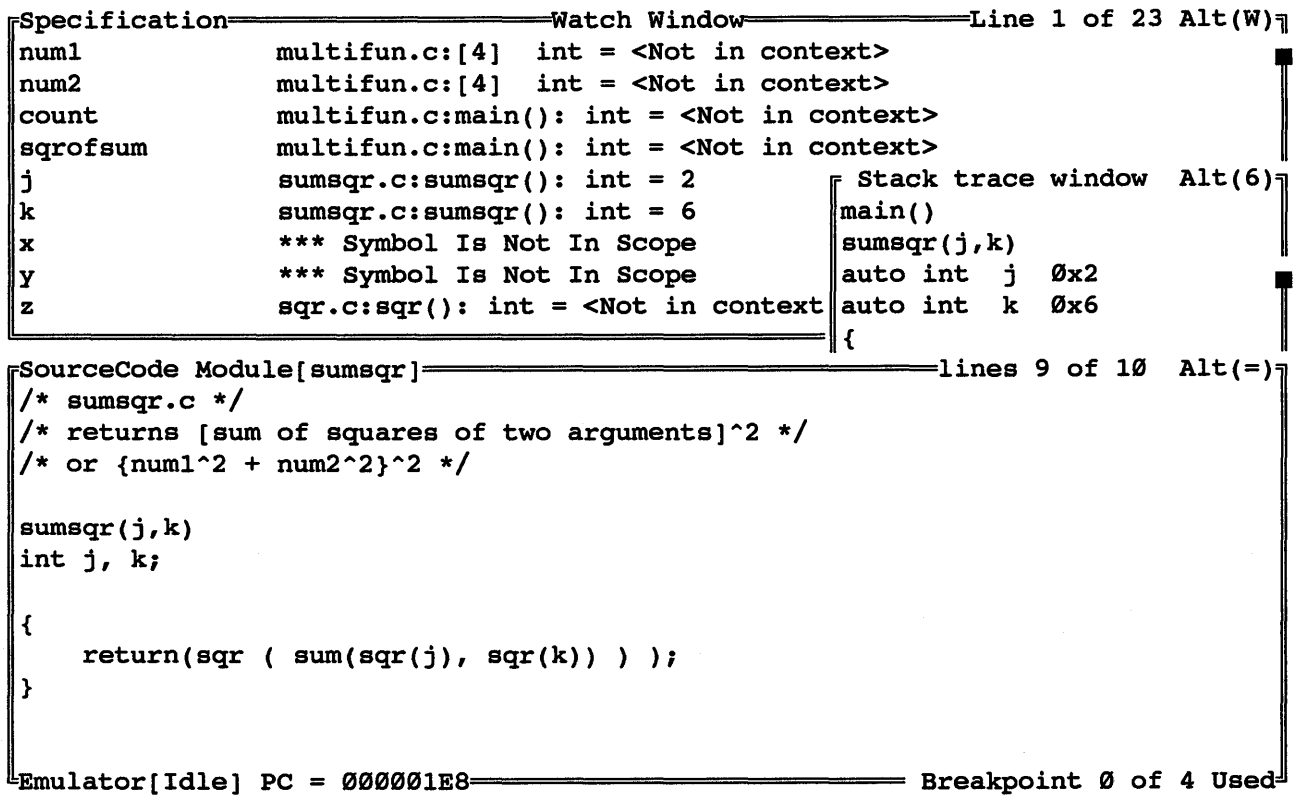


The Stack Trace window shows the functions and auto variables that are pushed onto the stack. The auto variables passed and their values are displayed. The display of auto variable values is in hexadecimal.

The Stack Trace window is updated each time emulation is stopped by hitting a breakpoint or using the SP command.

This version of SourceGate will display auto variable values in this window only for Archimedes and Software Development System (SDS) compilers. Future releases will provide this feature for other compilers.

4.4 Watch Window



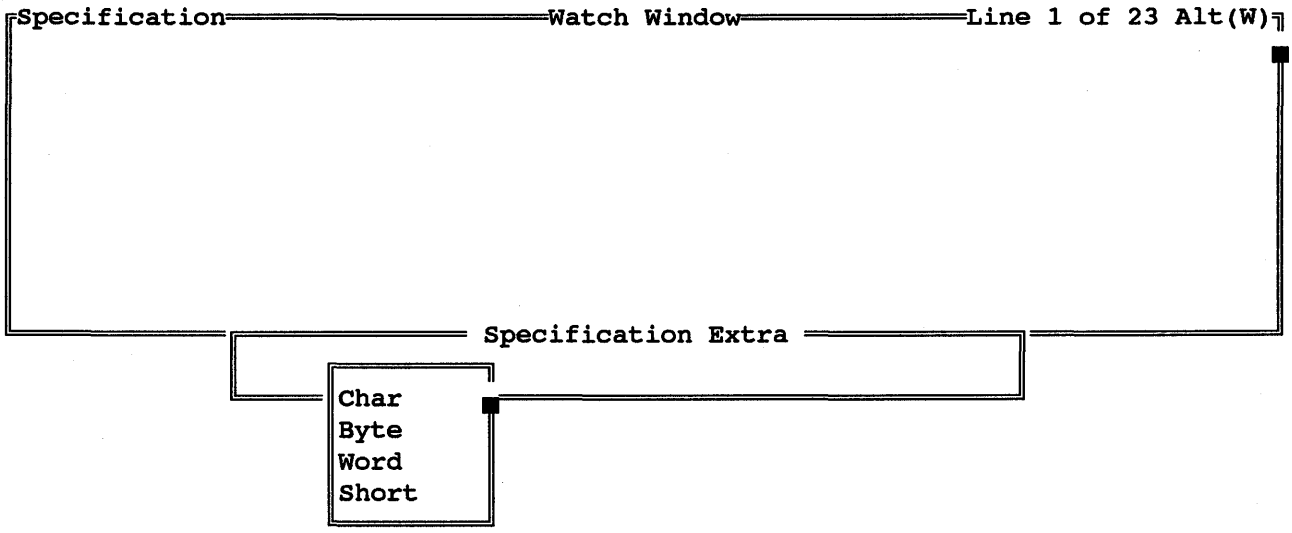
The Watch window has undergone significant changes since the last release of SourceGate. Data structure information along with the display of auto variables is supported when Archimedes and Software Development Systems (SDS) compilers are used. The Watch window can be saved and loaded again at a later date. All variable types are now supported and can be displayed. An explanation of each of these new features is shown in the sections that follow.

4.4.1 Entering Watch Window Variables

Watch window variables can be entered in several ways. While in the Watch window, variables can be defined directly by typing in the type, address (or symbol), and display type under the "Specification" column. Variables can also be entered by using the ^W (CTRL W) command. This command will display a "Specification Extra" window from which variables can be defined for use in the Watch window.

4.4.1.1 Entering Variables Using The ^W Command

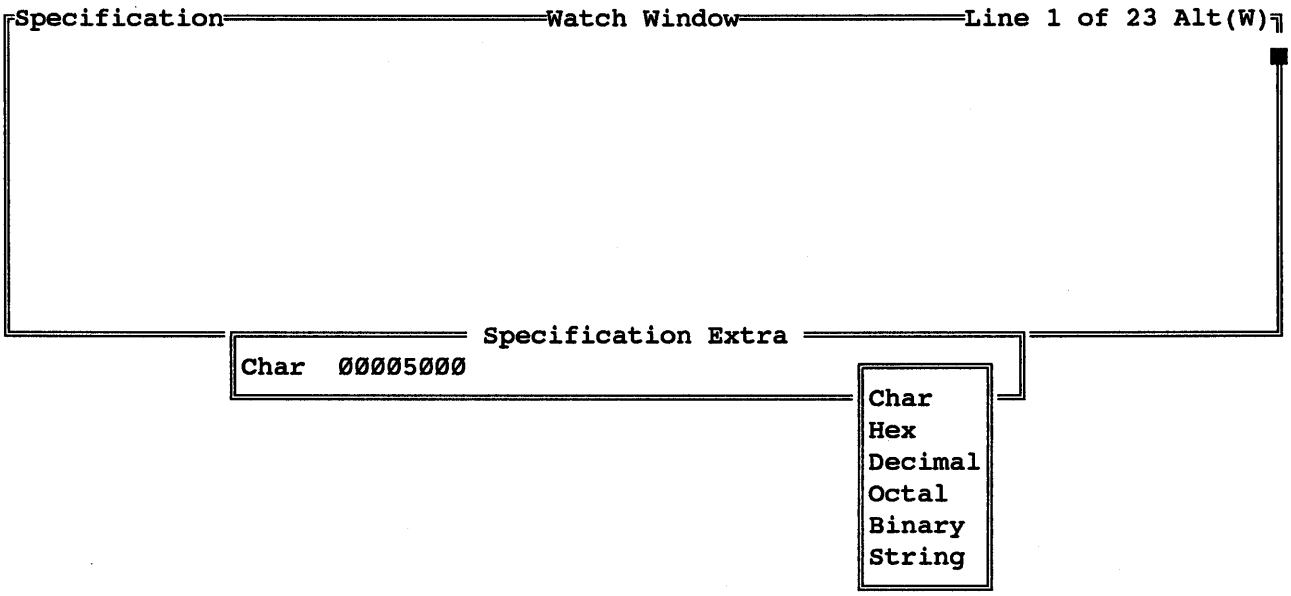
Entering a ^W will display a "Specification Extra" window from which variables can be defined for use in the Watch window. Entering ^W will produce the following screen:



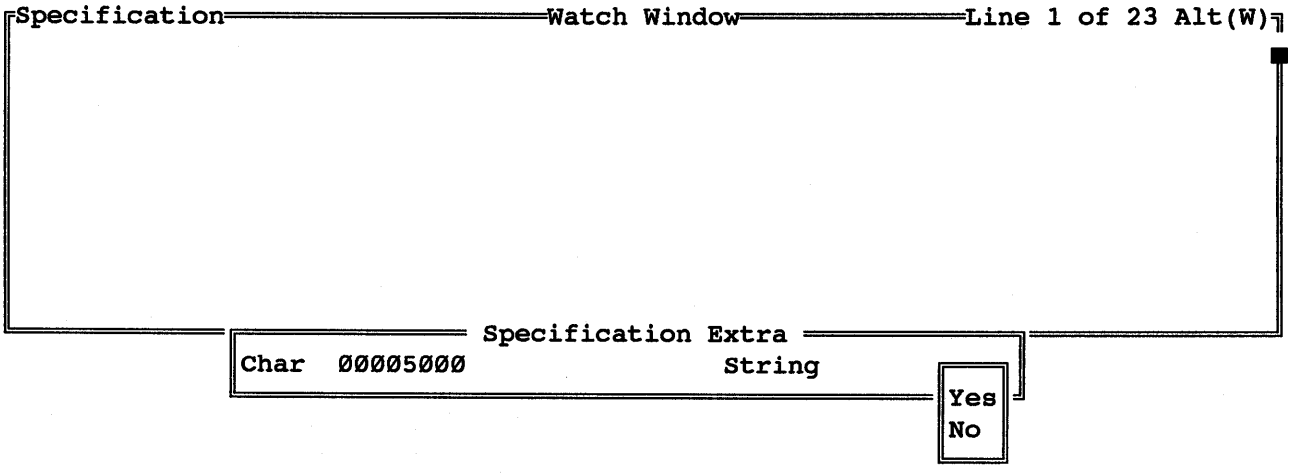
The variable type is defined by placing the highlighted bar on the desired selection and entering a <cr>. Additional options can be scrolled by using the up or down arrow keys.

Once a variable type has been chosen, the user needs to enter an address or symbol name of the variable to be monitored. To enter an address directly, simply type it in the provided space. To enter a symbol, enter a colon (:) and then the symbol name. The address will be entered automatically. The address values are always displayed in hexadecimal.

Once the address has been entered, the display type must be defined. To select a display type, simply move the highlighted bar to the desired selection and enter a <cr>.



After the type, address, and display type have been chosen, another field will be shown with "YES" and "NO" displayed. If the user wishes to add the defined variable to the Watch window, a "YES" should be entered. If "NO" is entered, the Specification Extra window is cleared and a new definition can begin.



If "YES" is chosen, then the defined variable is loaded into the Watch window and the current value of the variable is displayed in the specified output format:

```
┌Specification═══════════════════Watch Window═══════════════Line 1 of 23 Alt(W)┐
Char 00005000 S This is a test of the watch window.
```

In this example, the string "This is a test of the watch window." was stored in memory beginning at location 5000H and was terminated by a null byte. It is important to note that the Watch window will only display that part of the string that will fit in the window.

Some other examples of variables entered from the Specification Extra window are shown below:

```
┌Specification═══════════════════Watch Window═══════════════Line 1 of 23 Alt(W)┐
Char 00005000 S This is a test of the watch window.
Char 00005000 C T
Char 00005000 H 54
Char 00005000 D 84
Char 00005000 O 124
Char 00005000 B 01010100
```

4.4.1.2 Entering Variables From The Command Line

There is also a way to enter variables directly from the command line in the Command window. To enter variables in this fashion, the command "Watch <type> <value> <display type>" should be used. This will automatically create variables in the Watch window in the specified format.

Some examples of command line entered variables is shown below:

```
Specification=========================Watch Window=================Line 3 of 23 Alt(W)
char 5000 s      This is a test of the watch window.
c 5000 c         T
c 5000 h         54
c 5000 d         84
c 5000 o         124
c 5000 b         01010100
```

```
------------------------------ Command Window  Alt(C) ------------------------------
watch char 5000 s
watch c 5000 c
watch c 5000 h
watch c 5000 d
watch c 5000 o
watch c 5000 b
```

The following is a definition of valid types, values, and display types that can be entered from the command line in the format:

watch <type> <value> <display type>

Type:

The type field is used to tell the Watch window what type of field the value is. Any type can include an '*' after the type to denote a pointer. Some examples follow:

type	Value is the address of a type
type*	Value is a pointer to address of a type
type**	Value is a pointer to a pointer of a type

Types used:

char	(1)byte in length
int	(2)bytes in length
short	(2)bytes in length
word	(2)bytes in length
long	(4)bytes in length
addr	(4)bytes in length
float	(4)bytes in length
double	(8)bytes in length
long double	(10)bytes in length
extended float	(12)bytes in length
packed float	(12)bytes in length

Note: The length of each type may vary depending on the emulator used. The actual length is loaded from the configuration file except for Extended float and Packed float which are 68020 math processor dependent.

The abbreviation of 'LD' must be used for long double.

Value:

The value is used to tell the Watch window the address of the data inside of the emulator. This field can be a hex value or any symbol by using a ':' as the first character of the field (:name). An offset can be added to the symbol value by placing a '+' or '-' value after the symbol.

Arrays of values can be defined by placing a [number] after the value. Two dimensional arrays can be defined by placing a [number][number] after the value. It should be noted that on two dimensional arrays, the first set of brackets provides the display offset and the second set provides the number of entries to display.

Display Type:

The display type is used by the Watch window to determine the way in which the data will be displayed. The default display type is hexadecimal.

An example using each display type follows:

```
Hex       54
Decimal   84
Octal     124
Binary    01010100
Char      T
String    This is a test of the watch window.
```

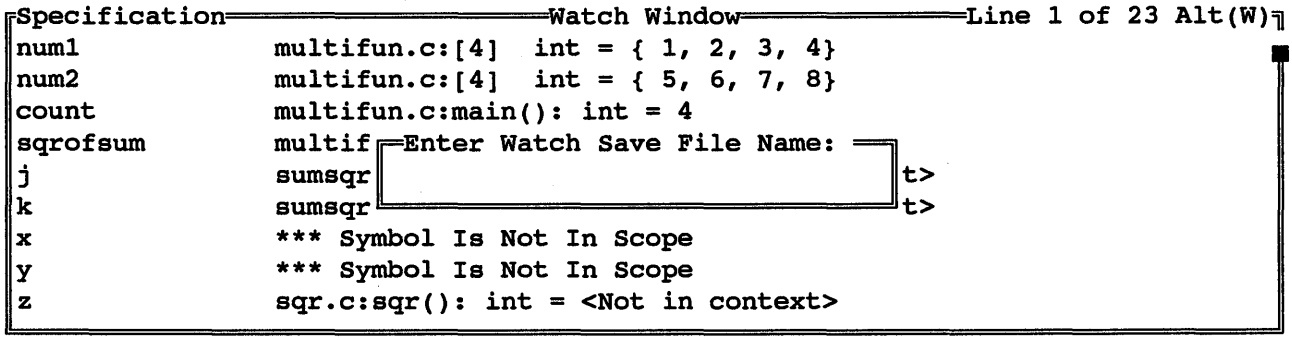
Note: The Watch window will only display what can be seen on a single line.

Only the first character of the display type is needed to define the type.

On float, double, extended double, and packed double, the display type can be any standard 'C' definition (%f, %g, %e, %F, %G, %E, etc.).

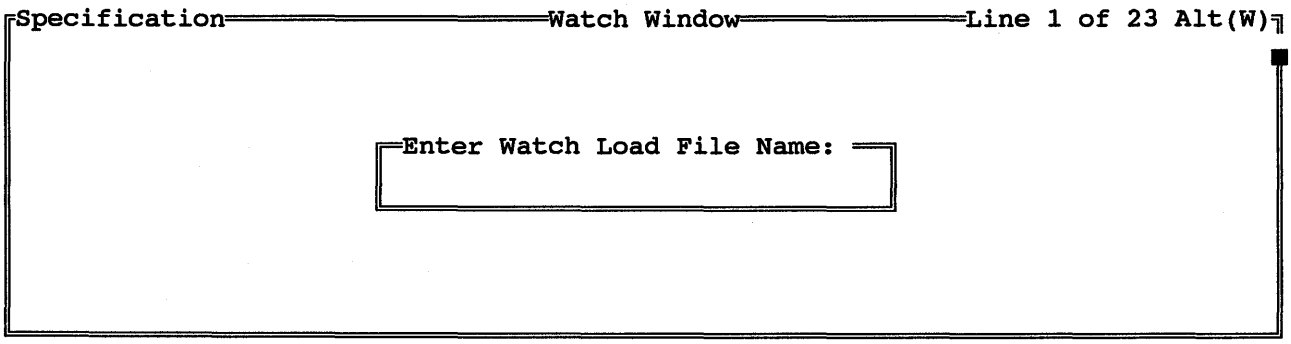
4.4.2 Saving The Watch Window

It is possible to save the current Watch window to disk by entering a ^S (CTRL S) while in the Watch window. After a ^S is entered, SourceGate will ask the user for a filename under which the Watch window data will be saved:



4.4.3 Restoring A Saved Watch Window

Once a Watch window has been saved, it can be restored by entering a ^L (CTRL L) or ^R (CTRL R) while in the Watch window. When one of these commands is entered, SourceGate will prompt the user to enter the name of the Watch window file to load:



After the filename is entered followed by a <cr>, the Watch window data contained in the load file will be put into the Watch window and all variables will be updated to their current values.

4.4.4 SDS and Archimedies Support

For those using Software Development Systems (SDS) and Archimedes compilers, this version of SourceGate will read linker output files from these compilers directly, without having to run the object file through a converter.

Being able to read the file directly from the linker has distinct advantages, the most notable being that "auto" variables and other data structure information can be supported. This support allows auto variable information to be displayed in the Watch window with the values of these variables shown within their functions as the program is stopped or stepped. Also, since the file comes directly from the linker, the format of each variable defined in the watch window is known without the user having to define it. SourceGate can track when a defined variable is active (when the program is operating inside the variable's function) and display its value. If the symbol is not in scope, "*** Symbol Is Not In Scope" is displayed for that variable:

```
Specification=========================Watch Window=================Line 1 of 23 Alt(W)
num1            multifun.c:[4]  int = { 1, 2, 3, 4}
num2            multifun.c:[4]  int = { 5, 6, 7, 8}
count           multifun.c:main(): int = 4
sqrofsum        multifun.c:main(): int = 6400
j               *** Symbol Is Not In Scope
k               *** Symbol Is Not In Scope
x               *** Symbol Is Not In Scope
y               *** Symbol Is Not In Scope
z               *** Symbol Is Not In Scope
```

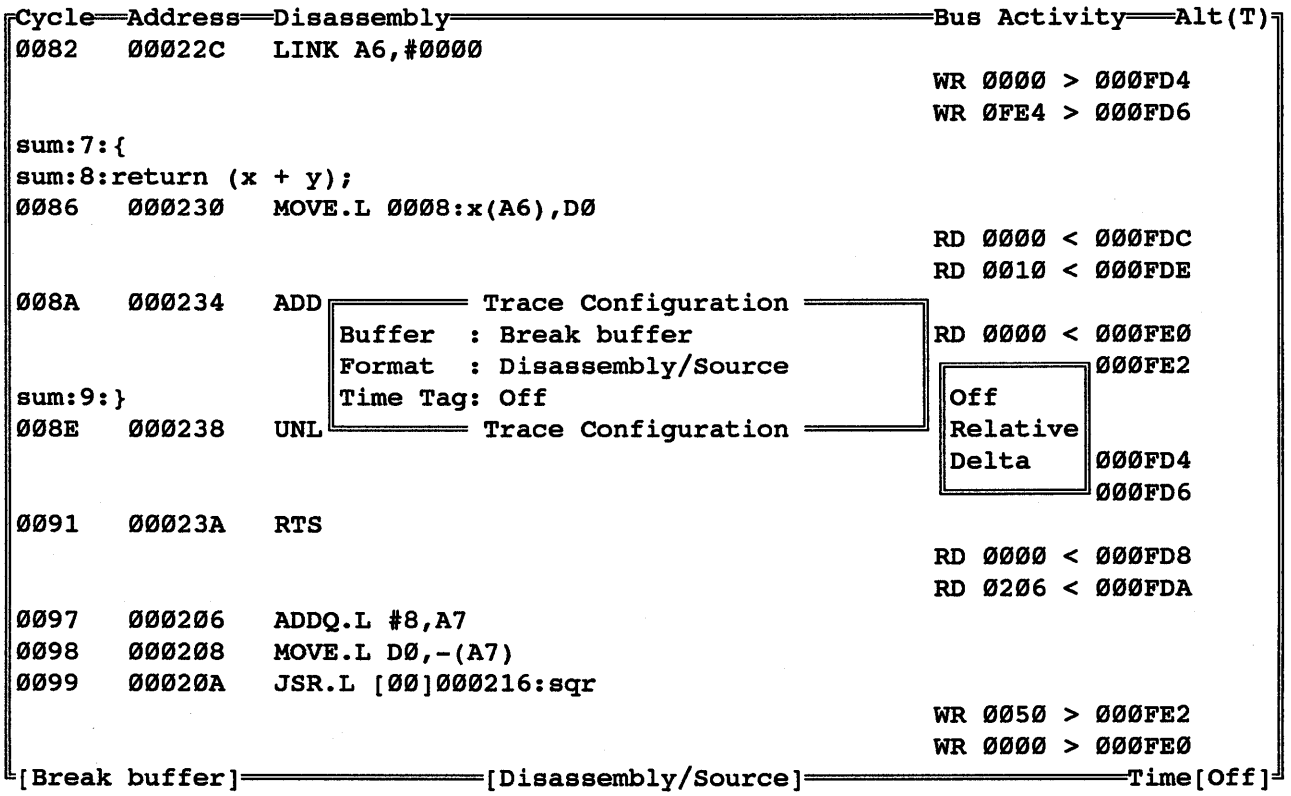
```
SourceCode Module[multifun]=================================lines 9 of 13  Alt(=)
/* multifun.c */
/* tests sumsqr function */
main()
{
   static int num1[4] = { 1, 2, 3, 4};
   static int num2[4] = { 5, 6, 7, 8};
   int sqrofsum, count;

   loop:  sqrofsum = 0;
   for (count=0; count<4; count++)
       sqrofsum = sumsqr(num1[count],num2[count]);
   goto loop;
Emulator[Idle] PC = 00000194============================ Breakpoint 0 of 4 Used
```

This capability will be extended to support those compilers producing COFF and IEEE-695 formats in future releases of SourceGate.

4.5 Trace Window Displaying Time Stamps

This release of SourceGate allows the user to set up the Trace window where the display of time stamps from the Performance Analysis Card (PAC) is in relative or delta mode. This option is available only in those units with PAC installed.

When the user moves the cursor into the "Time Tag:" field of the Trace Configuration window, the different choices for the trace's time stamp format are displayed:



The user can then decide which format is to be displayed. The choices are Off (no time stamp information displayed), Relative (time stamps shown are relative to the beginning of emulation), and Delta (time stamps show the time between each instruction or cycle depending on the format of the Trace window display).

If Relative is chosen for the time stamp format, then the Trace window will show a time stamp for each instruction or cycle (depending on the trace display format) relative to the beginning of emulation:

```
╔═Cyc.═Bnk═Addr.══Time Stamp══Disassembly═══════════════════Bus Activity══Alt(T)╗
multifun:10:for (count=0; count<4; count++)
 00FB      6013       125.5µs  CLRB
 00FD      6014       126.5µs  CLRA
 00FF      6015       127.5µs  STD 02,X
                                                          WR 00 > 220C
                                                          WR 00 > 220D
 0104      6017       130.0µs  TSX
 0107      6018       131.5µs  LDD 02,X
                                                          RD 00 < 220C
                                                          RD 00 < 220D
 010C      601A       134.0µs  SUBD #0004
 0110      601D       136.0µs  BGE 6043:?0003
multifun:11:sqrofsum = sumsqr(num1[count],num2[count]);
 0113      601F       137.5µs  TSX
 0116      6020       139.0µs  LDD 02,X
                                                          RD 00 < 220C
                                                          RD 00 < 220D
 011B      6022       141.5µs  ASLD
 011E      6023       143.0µs  ADDD #2008:?0001
 0122      6026       145.0µs  XGDX
 0125      6027       146.5µs  LDX 00,X
                                                          RD 00 < 2008
                                                          RD 05 < 2009
╚[Disassembly/Source]═[Trigger buffer]═Ports[Off]=[Unfiltered]=Time[Relative]═╝
```

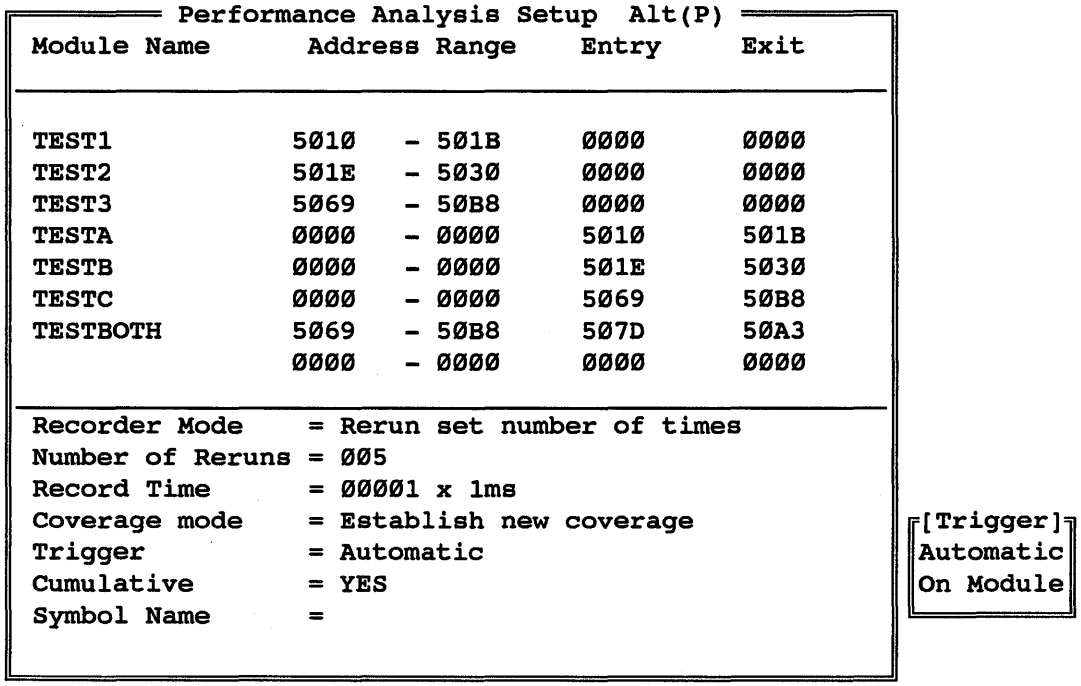
In the example above, these instructions were executed between 125.5µs and 146.5µs after emulation was started.

If Delta is chosen for the time stamp format, then the Trace window will show the elapsed time between each instruction or cycle (depending on the trace display format):

```
╔═Cyc.═Bnk═Addr.══Time Stamp══Disassembly═══════════════════Bus Activity══Alt(T)╗
multifun:10:for (count=0; count<4; count++)
 00FB      6013         0.5µs  CLRB
 00FD      6014         1.0µs  CLRA
 00FF      6015         1.0µs  STD 02,X
                                                          WR 00 > 220C
                                                          WR 00 > 220D
 0104      6017         2.5µs  TSX
 0107      6018         1.5µs  LDD 02,X
                                                          RD 00 < 220C
                                                          RD 00 < 220D
 010C      601A         2.5µs  SUBD #0004
 0110      601D         2.0µs  BGE 6043:?0003
multifun:11:sqrofsum = sumsqr(num1[count],num2[count]);
 0113      601F         1.5µs  TSX
 0116      6020         1.5µs  LDD 02,X
                                                          RD 00 < 220C
                                                          RD 00 < 220D
 011B      6022         2.5µs  ASLD
 011E      6023         1.5µs  ADDD #2008:?0001
 0122      6026         2.0µs  XGDX
 0125      6027         1.5µs  LDX 00,X
                                                          RD 00 < 2008
                                                          RD 05 < 2009
╚[Disassembly/Source]═[Trigger buffer]═Ports[Off]═[Unfiltered]══Time[Delta]═══╝
```

In the example above, 2.5µs elapsed between the "STD 02,X" and the "TSX" instruction that followed. This fairly large value was no doubt due to the memory writes associated with the "STD 02,X" instruction. These memory writes are shown in the bus activity column under the instruction that generated them.

4.6 Performance Analysis Trigger Window



This version of SourceGate adds the capability for the PAC (Performance Analysis Card) to trigger (or begin recording information) either automatically or on a module.

In the "Automatic" mode, the analyzer will begin to collect data immediately after pressing a <cr> (to start recording) in the profile window. The analyzer will continue to run for the length of time specified in the 'Record Time Field.'

In the "On Module" mode, the analyzer will not start until the setup conditions are met for the module selected as the trigger module. For example, if a module is selected as the trigger module and it is set up for an Address Range of 1000H to 1100H, then the analyzer will trigger (or start) on the first execution of an address in this range. If the trigger module is set for an Entry-Exit point, then the analyzer will trigger when either the

Entry or Exit point on the trigger module is executed. The trigger module is selected through the Profile window by highlighting the desired module and entering a <cr> to start recording.

NOTE: A good way to insure that the analyzer will trigger at the desired point is to define a module with both the Entry and Exit points set to the same address and then use this module as the trigger. For example, if a module has been defined with an Entry point of 1100H and an Exit point of 1250H and it is necessary to know exactly how long this routine will take to execute, the analyzer must trigger when it executes address 1100H and stop when it executes address 1250H. If a second module is defined with an Entry point of 1100H and an Exit point of 1100H, and this module is used as the trigger, then the analyzer will always start when address 1100H is reached (not 1250H which is the Exit point).

APPLICATION NOTE 1

USING THE PERFORMANCE ANALYSIS SYSTEM

1.Ø INTRODUCTION

The HMI Performance Analysis System allows the software engineer to optimize code and improve software performance by revealing the real-time execution activity of his system. Unlike some software based analysis tools, the HMI system is hardware based and therefore provides 1ØØ% collection of execution data. This information then allows the designer to fine tune his code eliminating bottlenecks and improving performance.

2.Ø OVERVIEW

The PAC (Performance Analysis Card) system runs within the SourceGate environment and is accessed through the PERFORMANCE WINDOW. Once the performance window is entered, a submenu will appear with the following window choices:

The SETUP window allows the user to establish the criteria used by PAC when collecting execution data. The user is allowed to configure 8 different test modules for collection of information. Modules may be defined on the basis of address ranges, address entry and exit points, or both. The way in which data is collected under these different setup conditions is explained in a later section. Other options in the setup window determine the number of collection periods and the duration of each period as well as the trigger method.

The PROFILE window displays a histogram along with a percentage value indicating the % of total execution time spent within each of the defined modules. It also shows the number of times each module was accessed (executions) along with Minimum, Maximum, and Average times spent in each module.

The COVERAGE window shows the user a histogram indicating the percentage of code covered in each module. The EXTENDED COVERAGE window may then be accessed to view a listing of all addresses that were covered and all that were not covered in each of the 8 modules. This feature can quickly reveal "holes" in code or a subroutine that never gets executed.

The COMMAND window is used to issue emulator commands while in the Performance Analysis mode.

The EXIT window is used to properly exit the Performance Analysis mode.

3.Ø SETUP WINDOW

Performance Analysis Setup Alt(P)

Module Name	Bank	Address Range	Entry	Exit
Trigger	X	0000 - 0000	1000	1000
Loop 1	X	0000 - 0000	1000	1014
Loop 2	X	0000 - 0000	1014	1000
Loop 3	X	0000 - 0000	100A	1016
Range All	X	1000 - 101D	0000	0000
Module 6	X	0000 - 0000	0000	0000
Module 7	X	0000 - 0000	0000	0000
Module 8	X	0000 - 0000	0000	0000

Recorder Mode = Rerun set number of times
Number of Reruns = ØØ1
Record Time = ØØØ1Ø x 1ms
Coverage mode = Establish new coverage
Trigger = On Module
Cumulative = Yes
Symbol Name =

The SETUP window is used to define the parameters used in the collection of data by the PAC hardware. There are several fields that must be defined before the PAC system will function properly. Each of those fields is explained in the following sections.

3.1 Module Name Fields

This field is used to define a name for the address range or entry and exit points on which the analysis will be performed. This name can be a symbolic name referring to a particular block of code such as "GET_INPUT" or any other name that is descriptive of the code in that particular module. Up to 8 modules can be named.

3.2 Address-Range Fields

The address-range fields are used to define the range of addresses whereby if the processor accesses an address within this range it will be recorded for processing in the Profile and Coverage windows. The first address field in the range is always smaller than the second field. The range specified can be any length for timing analysis in the profile window, but must be limited to 64K for code coverage analysis. Any hexadecimal digits (Ø-F) are allowable characters for these fields.

NOTE: If the code makes a large number of accesses to a particular range, the analyzer buffer may fill up rapidly, causing an overflow to occur. If this occurs it will be necessary to reduce the Record Time, reduce the Number of Reruns, or change the Range definition to correct this.

3.3 Entry-Exit Fields

The entry-exit fields are used to define address entry and exit points for analysis. Unlike the address-range, the entry-exit definition requires the code to execute both the entry and the exit address to qualify as an valid execution. For example, if the entry address is set to 11ØØH and the exit address is set to 12ØEH, the code MUST execute address 11ØØH and then, at some point later, it MUST execute address 12ØEH to qualify as an execution in this module. If no entries are made in the address-range fields then the code may execute addresses anywhere in the current 64K block after executing the entry point and before executing the exit point and the entire path of execution will be timed and profiled.

3.4 Combined Address-Range and Entry-Exit Fields

A module may be defined with both an address Range and an Entry and Exit point. Under these conditions, executions will be counted only if the code enters the specified Range at the specified Entry point and exits the range at the specified Exit point. Likewise the timer will start when the Entry point is executed and run only during the time that code is executing within the specified range and stop when the Exit

point is executed. For example, if the Range is set to 1000H to 1300H, the Entry point is set to 1100H, and the Exit point is set to 120EH, then, for an execution to be counted, the code must enter the Range (1000H to 1300H) at address 1100H and exit at address 120EH. If the code goes outside the defined Range after executing the Entry address, the timer will stop until the code re-enters the range and will continue until the Exit address is executed unless the code goes outside the Range again.

3.5 Recorder Mode Field

The Recorder Mode determines how the Performance Analysis will be run. The available options are:

Rerun Set Number of Times: This option runs, updates the screen, then reruns again and continues to do this for the number of times specified by the user.

Free Run (continuously): The free run option will run, update the screen, and run again continuously until stopped by the user.

3.6 Number of Reruns Field

This field determines the number of reruns executed if the "Rerun Set Number of Times" option is selected above.

3.7 Record Time Field

The Record Time determines how long the Performance Analysis will be run for each pass and is specified in milliseconds. The maximum record time is 99.99 seconds. The user should be careful not to specify too long of a record time because this can cause a buffer overflow condition as described in Section 3.2.

3.8 Coverage Mode Field

This field determines how the analysis will be displayed. The options are:

Establish New Coverage - All existing analysis will be written over by the new analysis being run.

Append Current Coverage - The existing analysis will be saved and the new coverage will be added to it.

3.9 Trigger Field

The Trigger field can be set to Automatic or On Module. In the automatic mode, the analyzer will begin to collect data immediately after pressing the <cr> in the profile window. The analyzer will continue to run for the length of time specified in the 'Record Time Field.'

In the On Module mode, the analyzer will not start until the setup conditions are met for the module selected as the trigger module. For example if the 2nd module is selected as the trigger module and it is setup for an Address Range of 1000H to 1100H, then the analyzer will trigger or start on the first execution of an address in this range. If the trigger module is set for an Entry-Exit point, then the analyzer will trigger when either the Entry or Exit point is executed on the trigger module. The trigger module is selected through the Profile window as explained in Section 4.0.

NOTE: A good way to insure that the analyzer will trigger at the desired point is to define a module with both the Entry and Exit points set to the same address and then use this module as the trigger. For example, if a module has been defined with an Entry point of 1100H and an Exit point of 1250H, and it is necessary to know exactly how long this routine will take to execute, the analyzer must trigger when it executes address 1100H and stop when it executes address 1250H. If a second module is defined with an Entry point of 1100H and an Exit point of 1100H, and this module is used as the trigger, then the analyzer will always start when address 1100H is executed.

3.10 Cumulative Field

The Cumulative field works in conjunction with the Number of Reruns field. If the Cumulative field is set to YES, the number of executions shown in the Profile window will accumulate for all the reruns. If this field is set to NO, the analyzer will reset and collect new data on each of the reruns. For example, assume the number of reruns is set to 3 and the Cumulative option is set to YES. If a module has 20 executions on the first run, 19 on the second, and 21 on the third, then the number in the execution column will be 60. If the Cumulative option is set to NO, the execution column will show 20 on the first run, reset, show 19 on the second, reset, and show 21 on

the third. In the same manner the percentage and time information will either be based on the accumulated data for all reruns or just the current run. The Total Record Time field will also depend on the Cumulative field setting. If it is set to YES then the Total Record Time will be the Record Time multiplied by the Number of Reruns (both set in the Setuup Window). If Cumulative is set to NO then the Total Record time will also reset for each rerun and show only the time for that particular run.

3.11 Symbol Name Field

The symbol name field allows the use of symbol names instead of hex values for the Address-Range and Entry-Exit fields. When the cursor is positioned in any of the Address Range or Entry-Exit fields the user can enter a ':' and the cursor will move to the Symbol Name field where a symbol name is entered. When the symbol is entered, the address of that symbol will then appear in the address field where the ':' was entered.

4.Ø PROFILE WINDOW

Performance Profile

Press return to start

Module Names	Executions	Max time	Avg Time	Min Time
Module 1	Ø	Ø.ØØØµs	Ø.ØØØµs	Ø.ØØØµs
Ø.ØØØ %				
Module 2	Ø	Ø.ØØØµs	Ø.ØØØµs	Ø.ØØØµs
Ø.ØØØ %				
Module 3	Ø	Ø.ØØØµs	Ø.ØØØµs	Ø.ØØØµs
Ø.ØØØ %				
Module 4	Ø	Ø.ØØØµs	Ø.ØØØµs	Ø.ØØØµs
Ø.ØØØ %				
Module 5	Ø	Ø.ØØØµs	Ø.ØØØµs	Ø.ØØØµs
Ø.ØØØ %				
Module 6	Ø	Ø.ØØØµs	Ø.ØØØµs	Ø.ØØØµs
Ø.ØØØ %				
Module 7	Ø	Ø.ØØØµs	Ø.ØØØµs	Ø.ØØØµs
Ø.ØØØ %				
Module 8	Ø	Ø.ØØØµs	Ø.ØØØµs	Ø.ØØØµs
Ø.ØØØ %				

Reruns Requested = ØØ5 Record Time = Ø.ØØØµs
Reruns Executed = Ø Total Record Time = Ø.ØØØµs
Timer Resolution = ØØ1 x 1ØØ ns

The Profile window displays the module name, percent of time that was spent in each module, number of executions for each module, along with max, min, and average time spent in each module. The user may move the highlight bar to any of the 8 modules to select the one that will be used as the trigger if the 'On Module' trigger mode is selected. The various fields that are displayed in the profile window are defined in this section.

4.1 Reruns Requested and Record Time Fields

These values were defined in the Setup window.

4.2 Reruns Executed Field

The is the actual number of runs that have been executed.

4.3 Total Record Time Field

This field shows the cummulative record time for all reruns executed. The user should pay close attention to this field. If the Total Record Time does not equal the Number of Reruns multiplied by the Record Time, then there has been a buffer overflow and the displayed data MAY be in error.

4.4 Timer Resolution Field

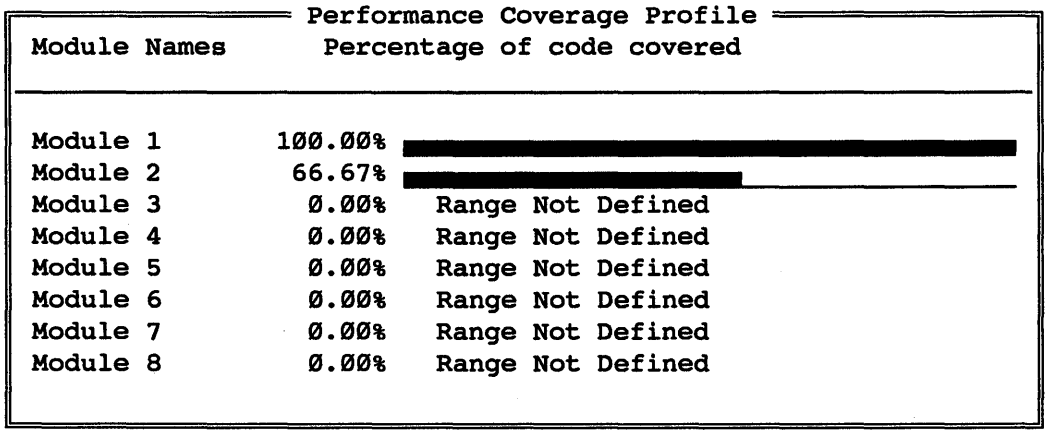
This field shows the resolution of the timer used by the Performance Analyzer and the time tag field of the trace buffer.

NOTE: The percentage fields and histograms shown in the profile window indicate the percent of total execution time spent within each module. These percentages will depend on the way the module is defined and the code execution path. The percentages displayed will rarely, if ever, add up to 100% for several reasons. Some modules may overlap in that code may begin in one module and jump to another one and depending on how the modules are defined, timing and execution information may be collected for both modules. For instance, assume a module is defined with an Entry point of 1100H and an Exit point of 1300H with no Range specified. If the code enters this module at 1100H and runs for 25 microseconds, then jumps to location 1420H in another module and runs for 10 microseconds, then comes back into the original module for 5 microseconds before hitting the exit address, the timer will record 40 microseconds for this module. The 10 microseconds spent in the other module will also be recorded as time spent in that module. While the correct information is collected and displayed for each module, it may not be what the user is expecting. The use of a Range definition along with the Entry-Exit points on this module would have prevented the 10 microseconds spent in the other module from being recorded. The user should give careful thought to module definition and examine the results very closely to avoid drawing the wrong conclusions.

The Min, Max, and Avg times spent in each module should also be carefully interpreted. The numbers simply mean that the code made at least one pass through this module at the minimum time, at least one pass at the maximum time, (obviously a longer path) and the Avg number indicates the average time spent in the module. As pointed out in Sections 3.2 and 3.3, the way in which the time data is collected depends on how the modules are defined.

For more information on module definition and interpretation of results, see section 9.0 "A Sample Session".

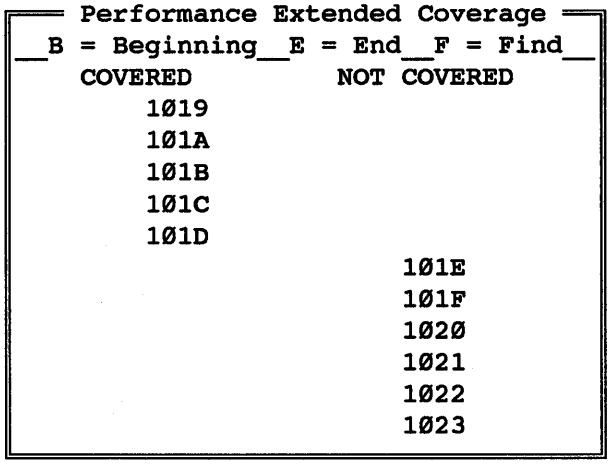
5.Ø COVERAGE WINDOW



The Coverage Window displays the module names and the histograms which represent the percentage of code that was covered in each module that has an Address-Range defined. A module that does not have an Address-Range defined will display the message "Range Not Defined".

NOTE: Coverage analysis is performed only for those modules that have an Address-Range defined and are 64K or less in size.

6.Ø EXTENDED COVERAGE WINDOW



The Extended Coverage window can be displayed for each module by moving the cursor to highlight the desired module and pressing <cr>. The Extended Coverage window will then display the covered and not covered addresses for the selected module.

The following commands may be used while in the Extended Coverage window:

C - Use this command to locate covered addresses. If only noncovered addresses appear when the Extended coverage window is opened, this command will quickly locate any covered addresses not currently visible.

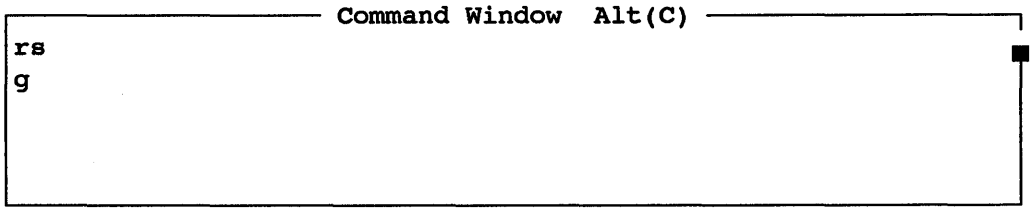
N - This command is used in the same way as the 'C' command to locate Noncovered addresses.

B - Used to move to the beginning of the listing.

E - Used to move to the end of the listing.

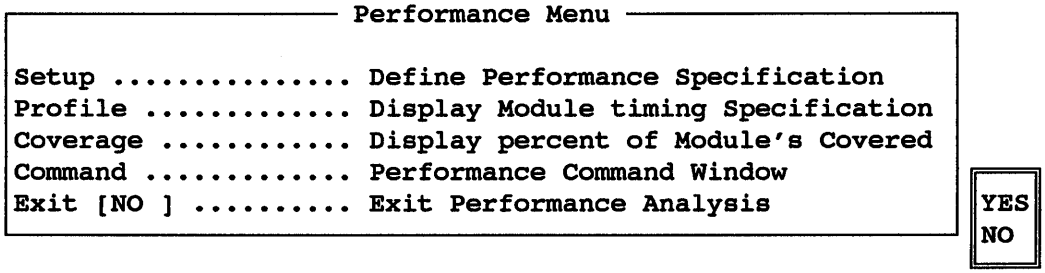
F - To find a particular address enter 'F' then press <cr>, you will then be asked to enter the hex address you wish to find.

7.Ø COMMAND WINDOW



Selecting the Command window will allow the user to issue standard emulator commands such as reset, go, stop, etc. The Command window option is available here as a convenience to avoid having to exit the Performance window to issue commands then re-enter to run the analysis.

8.Ø EXIT WINDOW



The Exit window is used to exit the Performance Analysis mode. To exit properly, move the cursor to highlight the Exit option and press <cr>. A small window will appear with the 'YES-NO' option. Press <cr> again and then use the up and down arrows to select YES and press <cr> again. At this point SourceGate will exit to the "Home" window list.

NOTE: When the performance analyzer is running, the display in the profile window may appear to update slowly which may seem to be inconsistent with an analysis time of a few milliseconds. However, while the data is actually collected in real time by the analyzer, the software must make several calculations on the collected data before displaying it on the screen. It is these calculations and display updating that make the analysis appear to be slower than real time.

9.0 A SAMPLE SESSION

The best way to illustrate the powerful capabilities of the HMI Performance Analysis System is to program the analyzer for a typical application and then examine the results. This sample session is based on a program written for the Motorola MC-6809 microprocessor. The code was purposely made very simple so that the reader can easily see the actual path the code will take, which will make it easier to understand the results the analyzer produces. The analyzer will then be programed with various setup configurations and the results examined. The purpose of this exercise will be to illustrate not only how to setup the analyzer, but how to obtain the most useful results from the system.

9.1 THE PROGRAM

The following is a listing of the code that will be used in this sample session:

	ADDRESS	CODE	COMMENTS
┌──	1000	LDA #1300	Load hardware stack pointer.
|	1004	LDU #1250	Load user stack pointer.
|	1007	LDA #12	Load A Register with 12H.
|	1009	STA 1201	Store A to address 1201.
	100C	LDA #05	Load A Register with 5H.
MAIN-1	100E	ADDA #10	Add 10H to A Register.
	1010	STA 1200	Store A to address 1200.
|	1013	LDA 1201	Load A from address 1201.
|	1016	EORA 1200	Exclusive OR A & address 1200.
|	1019	PSHU A	PUSH A Register on stack.
└──	101B	BSR 1026	BRANCH TO SUBROUTINE @ 1026.
┌──	101D	PULU A	PULL A Register from stack.
MAIN-2	101F	DECA	Decrement A Register.
	1020	STA 1200	Store A to address 1200.
└──	1023	JMP 1000	Jump back to beginning.
┌──	1026	LDB #10	Load B Register with 10H.
|	1028	DECB	Decrement B Register.
|	1029	CMPB #00	Compare B Register with 0.
	102B	PUSU B	PUSH B Register on stack.
LOOP	102D	BEQ 1034	Branch if Equal to 1034.
	102F	PULU B	PULL B Register from stack.
|	1031	JMP 1028	If not Equal, Decrement again.
└──	1034	RTS	Return from Subroutine to 101D.

NOTE: The RESET VECTOR for the MC6809 must be set for 1000H for this program to run properly.

The program has been divided into three sections or modules called MAIN 1, MAIN 2, and LOOP. In an actual application these modules might represent sections of code that the user might need to have specific timing or execution information about. In our example, MAIN-1 refers to a section of straight in-line code which Branches to a subroutine at the end. The module called LOOP refers to the subroutine and MAIN-2 represents the remainder of the program.

An examination of the listing and comments reveals that the code executes from address 1000 to address 101B then branches to a subroutine at address 1026. This subroutine is a simple loop that executes 16 (10H) times then returns to address 101D. The code then runs through address 1023 where it jumps back to the beginning and starts over.

We will assume the user has a critical timing problem and needs to know how long it takes to execute each of the modules. There are a number of ways that the Performance Analyzer may be set up to record data and it is very important that the user understand the differences in order to properly interpret the results. In the following sections we will look at the 3 basic ways to set up this problem and examine the results for each case.

9.2 SETUP NUMBER 1

RANGE ONLY

Performance Analysis Setup Alt(P)

Module Name	Bank	Address Range	Entry	Exit
Trigger	X	0000 - 0000	1000	1000
Main-1	X	1000 - 101C	0000	0000
Loop	X	1026 - 1034	0000	0000
Main-2	X	101D - 1025	0000	0000
Stack Area	X	1200 - 1300	0000	0000
All Memory	X	1000 - 1300	0000	0000
	X	0000 - 0000	0000	0000
	X	0000 - 0000	0000	0000

Recorder Mode	= Rerun set number of times
Number of Reruns	= 001
Record Time	= 00008 x 1ms
Coverage mode	= Establish new coverage
Trigger	= On Module
Cumulative	= Yes
Symbol Name	=

FIG. 1

In the first set up we will use address ranges only to define the modules to be tested. Figure 1 shows the setup window and the entries made for a range only analysis. Notice that the beginning and ending addresses for each module have been entered. Since the last instructions in modules MAIN-1 and MAIN-2 are multiple byte instructions, the ending address entered is the address of the last byte of those instructions. The user may also notice that several other modules have been defined. These modules are explained below.

TRIGGER MODULE - Because the emulator must be running when the Perfomance Analyzer is started, it is impossible to know where in memory the processor is running when the analysis is begun. If the analyzer is set for automatic triggering the PAC system will begin gathering data as soon as it is started. Since this could be in the middle of a module that must be timed accurately, it is possible to get incorrect data. Ideally we would like to control exactly where the analyzer starts in order to get the most useful data. We can accomplish this by using the "On Module" trigger mode and setting up a module with both the Entry and Exit point set to the same address. (This address should be the address we want to start the analyzer on.) In the example used here we have set the "Trigger Module" to address 1000 which is the beginning of the program. With this setup it does not matter where the processor is running, the analyzer will not start collecting data until address 1000 is accessed.

STACK AREA - This module refers to the area of memory used by the the user and hardware stacks. This will show how much time is spent in stack operations.

ALL MEMORY - This module includes the complete range of addresses that the sample program should execute if it runs properly. If this module ever indicates less than 100 percent it means that there has been an address outside the expected range accessed. (The program is in the weeds.)

With the setup complete the user then opens the Profile Window by pressing "HOME" then highlighting "PROFILE" and pressing <return>. At this point the emulator must be started. To do this press "HOME" again and highlight "COMMAND". Type "RS" to reset the emulator and then type "G" to start emulation. Now press "Cntrl-PageUp" to return to the Profile Window and press <return> again to start the analyzer.

When the analyzer stops the Profile Window display will look like Fig. 2:

RANGE ONLY

Performance Profile

Press return to start Use arrows for trigger on/off

Module Names	Executions	Max time	Avg Time	Min Time
Trigger	37	0.000µs	0.000µs	0.000µs
0.000 %				
Main-1	228	6.500µs	2.900µs	1.500µs
8.265 %				
Loop	1233	6.000µs	4.500µs	1.000µs
69.355 %				
Main-2	111	3.000µs	2.300µs	2.000µs
3.191 %				
Stack Area	1496	1.500µs	0.900µs	0.500µs
16.830 %				
All Memory	1	8.000µs	8.000µs	8.000µs
100.000 %				
	0	0.000ms	0.000ms	0.000ms
0.000 %				
	0	0.000µs	0.000µs	0.000µs
0.000 %				

Reruns Requested = 001 Record Time = 8.000ms
Reruns Executed = 1 Total Record Time = 8.000ms
Timer Resolution = 001 x 100 ns

FIG. 2

If we examine Fig. 2 we see that the trigger point occured 38 times. (The count shows 37, because the first trigger is not counted.) This means that the program started from address 1000 38 times. We also see that module MAIN-1 was entered 228 times and the average time spent in that module or range was 2.9 micro seconds. Likewise the LOOP module was entered 1233 times with an average time of 4.5 microseconds and MAIN-2 was entered 111 times and had an average time of 2.3 microseconds per entry. If the user stops here he might assume that these times represent the average execution times for these modules, however, this is not the case.

It is obvious that since all three of these modules run exactly the same each time, the execution time should always be very close to the same. Since the Min, Max and Avg times displayed in Fig. 2 vary widely, it is evident that they do not represent the full execution time of the module. In fact, by definition, the Range only setup will start timing when the processor enters the specified range and stop when it leaves. Therefore, if we examine the MAIN 1 module we see that it enters the range a total of 6 times during execution and the times shown in Fig. 2 represent the Min, Max, and Avg times spent in the range and NOT the Min, Max, and Avg times required to execute the entire module. All three of the defined modules have instructions that access memory locations outside their range. The STA, LDA 1201, PSHU, PULU and EORA instructions all do memory accesses outside the module range. Since these accesses take processor time, they should be counted as part of the execution time for that particular module. We need a set up that will measure not only the time spent in the range, but any time spent outside the range during execution of that particular module.

Performance Coverage Profile

Module Names	Percentage of code covered	
Trigger	0.00%	Range Not Defined
Main-1	100.00%	
Loop	100.00%	
Main-2	100.00%	
Stack Area	2.72%	
All Memory	7.93%	
Module 7	0.00%	Range Not Defined
Module 8	0.00%	Range Not Defined

FIG. 3

Fig. 3 shows the code coverage window for the Range only setup. As expected, the 3 modules we are interested in show 100% coverage. However, if this were not the case, we could highlight a particular module and press <return> to get the extended coverage window. This would show the actual addresses that were covered and the ones not covered. This would be useful in finding sections of code that never get executed.

9.3 SETUP NUMBER 2

BOTH RANGE AND ENTRY-EXIT

Performance Analysis Setup Alt(P)

Module Name	Bank	Address Range	Entry	Exit
Trigger	X	0000 - 0000	1000	1000
Main-1		1000 - 101C	1000	101C
Loop	X	1026 - 1034	1026	1034
Main-2	X	101D - 1025	101D	1025
Stack Area	X	1200 - 1300	0000	0000
All Memory	X	1000 - 1300	0000	0000
	X	0000 - 0000	0000	0000
	X	0000 - 0000	0000	0000

Recorder Mode	= Rerun set number of times
Number of Reruns	= 001
Record Time	= 00010 x 1ms
Coverage mode	= Establish new coverage
Trigger	= On Module
Cumulative	= Yes
Symbol Name	=

FIG. 4

In the second setup we have added a definition for the Entry and Exit points for each of the 3 modules of interest as shown in Fig. 4. This type of setup will cause the timer to start when the Entry address is accessed and stop when the Exit address is accessed. Referring to the procedure used with setup 1, the user should now reset and restart the emulator. After returning to the Profile Window, the analyzer should be started by pressing

<return>. When the analyzer stops the display should look like Fig. 5:

BOTH RANGE AND ENTRY-EXIT

Performance Profile

Press return to start Use arrows for trigger on/off

Module Names	Executions	Max time	Avg Time	Min Time
Trigger	37	0.000µs	0.000µs	0.000µs
0.000 %				
Main-1	38	17.500µs	17.400µs	17.000µs
8.265 %				
Loop	37	149.000µs	148.900µs	148.600µs
68.865 %				
Main-2	37	6.500µs	6.500µs	6.500µs
3.006 %				
Stack Area	1496	1.500µs	0.900µs	0.500µs
16.830 %				
All Memory	1	8.000µs	8.000µs	8.000µs
100.000 %				
	0	0.000ms	0.000ms	0.000ms
0.000 %				
	0	0.000µs	0.000µs	0.000µs
0.000 %				

Reruns Requested = 001 Record Time = 8.000ms
Reruns Executed = 1 Total Record Time = 8.000ms
Timer Resolution = 001 x 100 ns

FIG. 5

The user will immediately notice that this display is vastly different than the one shown in Fig. 2. The number of executions for MAIN 1,MAIN 2, and LOOP are considerably less than for the previous setup. Also the times shown are much greater than for the Range only setup. Since the Trigger number tells us that the program starts 38 times (remember the first trigger is not counted), the user can see that the program ran through MAIN-1 38 times, MAIN-2 37 times and LOOP 37 times before the 8 ms record time ran out.

Looking at the execution times we can see that the Min, Max and Avg times are very close to each other which could lead us to believe that they now represent the true execution times of the modules. However, referring to Section 3.4 of the Performance Analysis Description, we see that with this setup, the timer does start when the entry address is accessed and stops when the exit address is accessed. However, the timer also stops when the processor accesses addresses outside the range specified and restarts when the range is reentered. Therefore since we have already determined that each of the modules accesses addresses outside their code range, we can see that these times are still not representative of the total execution time for the modules.

9.4 SETUP NUMBER 3

ENTRY-EXIT ONLY

Performance Analysis Setup Alt(P)

Module Name	Bank	Address Range	Entry	Exit
Trigger	X	0000 - 0000	1000	1000
Main-1	X	0000 - 0000	1000	101C
Loop	X	0000 - 0000	1026	1034
Main-2	X	0000 - 0000	101D	1025
Stack Area	X	1200 - 1300	0000	0000
All Memory	X	1000 - 1300	0000	0000
	X	0000 - 0000	0000	0000
	X	0000 - 0000	0000	0000

Recorder Mode	= Rerun set number of times
Number of Reruns	= 001
Record Time	= 00010 x 1ms
Coverage mode	= Establish new coverage
Trigger	= On Module
Cumulative	= Yes
Symbol Name	=

FIG. 6

In setup number 3 we have specified only the Entry-Exit points for each of the 3 modules of interest. Referring to Section 3.3 of the PAC Description, we see that this type of setup will give us timing information that starts when the Entry address is accessed and stops ONLY when the Exit address is accessed, regardless of where in memory the processor runs. This should give us the kind of timing information we need for these modules. We now rerun the analyzer as before which produces the display shown in Fig. 7.

ENTRY-EXIT

Performance Profile

Press return to start — Use arrows for trigger on/off

Module Names	Executions	Max time	Avg Time	Min Time
Trigger	37	0.000µs	0.000µs	0.000µs
0.000 %				
Main-1	38	20.000µs	19.900µs	19.400µs
9.452 %				
Loop	37	180.000µs	179.900µs	179.900µs
83.203 %				
Main-2	37	7.500µs	7.500µs	7.500µs
3.469 %				
Stack Area	1496	1.500µs	0.900µs	0.500µs
16.830 %				
All Memory	1	8.000µs	8.000µs	8.000µs
100.000 %				
	0	0.000ms	0.000ms	0.000ms
0.000 %				
	0	0.000µs	0.000µs	0.000µs
0.000 %				

Reruns Requested = 001 Record Time = 8.000ms
Reruns Executed = 1 Total Record Time = 8.000ms
Timer Resolution = 001 x 100 ns

FIG. 7

From this display we can see that, as expected, the number of executions for each module has stayed the same as in Setup 2, however the execution times have again changed. The longer times are evidence that the analyzer is now timing the total execution time for each module including the time spent accessing other memory locations outside the code range.

We can now see that the LOOP module, which appeared to take only about 4.5 microseconds to execute in Setup 1, actually takes 180 microseconds and over 83% of the processor time. In a piece of code as simple as this example, the user can easily see that the LOOP module would take the most execution time because it is running 16 times for every single pass of the program. Keep in mind that the purpose of making this example simple was to make it easier to interpret the results of the analyzer. In real applications the PAC system can give the user information about very complex programs that cannot be analyzed by simple inspection.

9.5 CONCLUSION

In order to obtain useful information from the PAC system the user MUST understand how it gathers and processes execution data. As we have seen in this sample session, it depends on the type of information that the user needs as to how the system should be set up. In our example, we were interested in module execution times and the Entry-Exit Only setup was the method needed to produce accurate results. However, this does not mean that this is the best method to use in all cases. The user should study the PAC Description and FULLY understand its operation to determine the proper method to use in obtaining the kind of information needed. The user is incouraged to experiment with the PAC system using different combinations of setup parameters to see how they affect the results.