ED KIRKPATRICK
AD ELECTRONICS
603 964 5924

EMUL68™-PC Windows

User Guide

We would appreciate any feedback about the product (including the manual) ranging from simple software defects to suggestions on how to improve the examples. Thank you.

NOHAU
CORPORATION

EMUL68™ - PC Windows
User Guide

Copyright © 1997 by

nohau
CORPORATION

51 E. Campbell Avenue
Campbell, CA. 95008
Tel: (408) 866-1820
Fax: (408) 378-7869
BBS: (408) 378-0940
URL: http://www.nohau.com
E-Mail: support@nohau.com
All rights reserved worldwide.

Edition 1: Feb 10, 1997

Development Team:	Documentation:
Chris Boehr Peter Zou Nils Johansson Per Nothagen	Chris Boehr Bob Shoemaker Jolinda Pizzirani

Warranty Information

The EMUL68™-PC Emulator board, Trace board, pods, Emulator Cable, Serial Box and LanICE hardware are sold with a one-year warranty starting from the date of purchase. Defective components under warranty will either be repaired or replaced at Nohau's discretion.

Each optional adapter, cable, and extender is sold with a 90-day warranty, except that it may be subject to repair charges if damage was caused by the user's actions.

The EMUL68™-PC Windows Emulation software is sold with no warranty, but upgrades will be distributed to all customers up to one year from the date of purchase.

Nohau Corporation makes no other warranties, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. In no event will Nohau Corporation be liable for consequential damages. Third-party software sold by Nohau carries the manufacturer's warranty.

Warning:	***Always turn on the PC before applying power to the target. Always turn off the target power before turning off the PC power.***

Copyright © 1997 Nohau Corporation

Table of Contents

Copyright © 1997 Nohau Corporation

Copyright © 1997 Nohau Corporation

Copyright © 1997 Nohau Corporation

INDEX

LIST OF NOHAU REPRESENTATIVES AND DISTRIBUTORS

Copyright © 1997 Nohau Corporation

Chapter 1: Introducing EMUL68™-PC Windows

Introduction to EMUL68™-PC Windows

EMUL68-PC is a PC-compatible computer based in-circuit emulator for the Motorola 68HC11 8-bit family of microcontrollers. EMUL68-PC consists of an emulator "plug-in" board, a five foot (1.5 m) long cable, various pod boards and an optional trace board.

The EMUL68-PC Windows (also referred to as *Win68*) software interface is a *Microsoft Windows 3.x / 95* application. It follows the MS Windows Multiple Document Interface Standard, resulting in the same look and feel as applications produced by *Microsoft* and others for *MS-Windows*.

The EMUL68-PC Windows user interface is consistent with most other *MS Windows* applications and includes dynamically changing menus, moveable and scrollable "child" windows, function key shortcuts for menu items, and context sensitive help. Anyone familiar with MS Windows applications will be able to use EMUL68-PC Windows with little or no other assistance.

The EMUL68-PC hardware is modular. The software user interface implements an effective high level debugger, showing the user's source code and supporting local and global variables, C typedefs, and C structures. The trace board option adds bus cycle tracing, triggering and filtering functions.

How To Use This Manual

This manual was written with different kinds of users in mind. All users should have *MS Windows* installed and have learned the basic skills taught in the Basic Skills chapter of the Microsoft Windows 3.x User's Guide, or in the Windows '95 tutorial. It also assumes a basic familiarity with the chip you are using. Many of EMUL68-PC's features are designed around the features of the supported chips. Being familiar with the chip is a prerequisite to understanding how to use the emulator productively.

Note: This manual does not include detailed installation instructions for EMUL68-PC, or the many pods available. For detailed installation information, please refer to Section 2 of the EMUL68-PC User's Manual, and also to Section 9 if you have a bankswitched emulator board. Detailed information about pods can be found in Section 6. If you did not receive the EMUL68-PC User's Manual with your purchase, please call Nohau customer support and request one.

If you are new to emulators of any kind

read both this manual and the EMUL68 User's Manual completely including the reference chapters. You may skip the sections in the EMUL68 User's Manual that describe pods you do not have, and the sections on trace or bankswitching if you not have a trace board or a bankswitched emulator board.

Copyright © 1997 Nohau Corporation

If you have used emulators with other microprocessors

but are not familiar with the specific HC11 family member being emulated, you are strongly encouraged to review the features of the chip you have, then thoroughly read the section in the EMUL68 User's Manual that describes the applicable pod before running the emulator.

If you are familiar with emulators, MS Windows, and the chip,

read the Emulator Board and Software User Interface Chapters, skim the section that describes the pod you are using, and then begin using EMUL68™-PC, referring to on-line help, when needed. After a few days of use, skimming the Reference chapters may highlight useful features.

Manual Conventions

Type the words shown in double quotes exactly as shown but without the quotation marks.

Single keys which have multi-character names are shown inside angle brackets - i.e. <Enter>, <Ctrl>, <Tab>, <Del>, <Ins>, and <Alt> keys. Function keys are shown as <F1>.

Use the <Alt> and <Ctrl> keys like shift keys. Hold them down while you press the key that follows them in the text. For example, if the text instructs you to type <Alt>F, press down and hold down the <Alt> key then press the F key. Control key combinations are shown as either <Ctrl>X (press and hold the Ctrl key while pressing the 'X' key) or alternately "^X".

Where you see "<Alt>RF" as the keyboard shortcut, you should type <Alt>R (hold the Alt key down while you then press the R key) to open the Run menu, then press the F key (without the Alt key) to activate the feature of EMUL68-PC Windows that starts emulation without enabling defined breakpoints.

Holding down the Shift key or turning on <CapsLock> is not necessary. Even though the keyboard shortcuts are all shown in capital letters, the shortcuts are not case sensitive.

Window names and labels that appear on the screen are printed in **bold** to set them apart from the rest of the text.

Notes and hints are printed in italics, and warnings have a box around them to set them apart from the rest of the manual text. Please pay careful attention to them.

Quick Installation Instructions

System requirements

EMUL68™-PC requires a personal computer with at least one free ISA (or EISA) bus slot. The PC must also have at least four megabytes of RAM (8MB

Copyright © 1997 Nohau Corporation

recommended for Win95), an 80386, 80486, or Pentium compatible CPU, a hard disk with at least 3 megabytes of unused space and *Microsoft Windows 3.1, 3.11* or *Windows95* installed. A mouse is not required, but is strongly recommended.

Quick setup instructions

The hardware and software are designed to be easily installed and quickly running on most personal computer systems. Users can normally begin using their emulator (without yet connecting to the target) after following these initial steps. However, if you are new to personal computers, if you are unsure about what to do after reading the quick installation instructions, or if your emulator does not work after you follow these instructions, follow the steps for installing and configuring each board and the software as outlined in their respective chapters in this or the EMUL68™-PC [DOS] Users Manual.

Warning: Always turn on the PC before applying power to the target (single board computer, device under test, etc.) Always turn off the target power before turning off the PC power.

Installing The Emulator

Installing the emulator board is much like installing most other AT-style boards:

1. Turn off the power.
2. Remove the PC cover.
3. Remove the slot cover (if present) for an available 8 bit slot.
4. Insert the emulator board into the slot and use a screw to secure the emulator.
5. (If optional trace board is present) insert the trace board into an adjacent 8 bit slot, then connect the supplied short ribbon cable between the two boards.
6. Unless you will be installing a trace board, you can now close the PC's cover
7. Attach the ribbon cable to the emulator and the pod, observing correct orientation of the connector keys.
8. Remove any anti-static foam from the pins on the bottom of the POD and / or adapter before applying power.
9. Before attaching the pod to your target, it is a good idea to power up the PC, install the WIN68 software, and run one of the supplied example files in the *Examples* subdirectory.

Installing The Software

To install this software under *MS-Windows 3.x*, run SETUP.EXE by typing *"win a:setup"* at the DOS prompt or, from within *MS-Windows*, by selecting the RUN item in the Program Manager File menu and typing *"a:setup"* as the file to run.

Copyright © 1997 Nohau Corporation

To install this software under Windows '95, Open the My Computer facility, select the floppy drive containing the installation diskette, and double click on the Setup icon.

After SETUP.EXE is started, a dialog box will ask for a directory for the EMUL68-PC software. Either accept the suggested directory or enter a different one. SETUP will create the directories as needed, decompress and copy the files from the installation floppy to the hard disk and change the paths in the *"emul11.ini"* file. When installed, there will be a NOHAU program group containing the EMUL68 icon. Double-clicking on this icon will start the EMUL68-PC Windows application.

A *readme.txt* text file (and possibly others) is also copied to the specified installation directory. These files contain important information about the latest revisions of the software and known errata. Please take the time to read these files.

Initial Software Configuration

Hint *Nohau recommends that prior to connecting to your target for the first time, that you setup and operate the emulator using the included demo programs to verify correct emulator operation. This requires that you install the emulator boards (and optional trace board if you have one), connect the pod, and possibly configure the pod. The pod should be configured for internal operation by insuring that the crystal and power jumpers (or switches) are set to internal. It may be necessary to further configure some pods, please see Chapter 5: POD Boards for specific pod information.*

Using the **Config | Emulator hardware** menu selection, check the following, and set the options to match your configuration. Due to the large number of possible configurations, the defaults may not be appropriate for your emulator:

1. The emulator port address. This defaults to the same address that emulators are configured for at the factory.
2. The crystal frequency. This should be set to the operating frequency (not the E-clock rate). If the pod is set for internal operation, this should be the same as the crystal on the pod. If you are using an 'HC11 variant, such as the P2 or PH8, with an on-chip PLL, set this field to 4 times the maximum PLL output frequency (E clock source).
3. The processor type. This is selected from a drop-down list to match your target device. The pod you are using must support your intended target, many pods support more than one target device. Do not select a processor type not supported by the pod you are using.
4. The Pod Class. This is set to **Single Chip** or **Expanded**, and must match your (hardware) pod class.

Copyright © 1997 Nohau Corporation

If you did not use the default emulator suggested installation directories, verify and set as needed the **Config | Paths** for emulator and user files.

Also use the **Config | Memory Map..** item to insure that all memory is mapped to the emulator and not write protected.

Follow the instructions below to load the demo program and run it. In the event of difficulty please refer to the detailed instructions for installing the hardware and configuring the software in other chapters in this manual.

*Note: When connecting to your target, the emulator configuration, pod jumpers and switches, and software configuration parameters may require modification. Please recheck these settings and other options in these dialogs for correctness (as well as the **Config | Bankswitching** dialog box if you have a bankswitching emulator) prior to connecting to your target. Thank you.*

Quick Start Instructions

This section describes how to quickly start using EMUL68-PC Windows to debug an existing program or target board once the EMUL68-PC hardware and software are installed. If the Quick Installation instructions do not work, you will most likely need to adjust either the hardware jumpers, the software configuration, or possibly both. Please refer to the appropriate chapters of the EMUL68-PC User's Manual for setting the jumpers on the emulator board, the trace board, or the pod.

To load and execute a program:

1. Select **Load code..** from the **File** menu and identify the file to load using the dialog box.
2. Select **Reset and GO!** from the Run menu, -or-
   click on the **RESET** and **GO** buttons in the tool bar in that order.

To set a breakpoint:

1. Click once on the line number for the source line in any **Source** window. You may only set a breakpoint on a source line that causes instructions to be generated. -or-
2. Click twice on the desired instruction in any **Program** window, -or-
3. Click once on the address in any **Program** window, -or-
4. Highlight (click once on) the desired instruction and select **Toggle** from the **Breakpoints** menu (shortcut: <F2>) -or-
5. Select **At** from the **Breakpoints** (shortcut: <Alt><F2>) menu, then enter an address in the dialog box.

Copyright © 1997 Nohau Corporation

To make a breakpoint inactive:

1. Click once on the line number in the **Source** window or an address in the **Program** window -or-
2. Click on the desired breakpoint in any **Program** or **Source** window, then press F2, -or-
3. Select Setup .. from the **Breakpoints** menu, click on the breakpoint, click on the **Toggle** button, -or-
4. Highlight (click once on) the breakpoint and select **Toggle Breakpoint** from the **Program** or **Breakpoints** menus.

To delete a breakpoint:

1. From the **Breakpoints** menu Select **Setup ..**, click on the breakpoint, click on the **Delete** button, -or-
2. Select **Delete All** from the **Breakpoints** menu.

To use the in-line assembler to add program statements:

1. Scroll a **Program** window until it shows the address to be changed or press <Ctrl>A and type in the desired address...
2. Highlight the instruction to be changed with the cursor or arrow keys...
3. Type the desired mnemonic and operands (this will open an **Enter new instruction:** dialog box) and press <Enter>.

To change a RAM value:

1. Scroll a **Data** window until it shows the data to be changed, then highlight the data to be changed with the cursor or arrow keys -or-
2. Select the **Data** window, then press <Ctrl>A, then enter the hex address or symbol name in the dialog box. C variable and function names may be entered directly, assembly language symbols must be preceded by an ampersand ('&').
3. Type the desired value (this will open an **Enter data** dialog box) and hit <Enter>.

Warning:	***Always turn on the PC before applying power to the target. Always turn off the target power before turning off the PC power.***

Copyright © 1997 Nohau Corporation

Chapter 2: Software User Interface

Detailed Software Installation Instructions

Before installing the software, it is important to have a basic understanding of how to operate *MS Windows*. For help mastering *MS Windows*, please refer to the *Microsoft Windows User's Guide* for the version you are using.

The EMUL68-PC Windows installation diskette includes an *MS Windows* compatible SETUP.EXE installation program. To install this software from Windows 3.1/3.11, run SETUP.EXE by typing "WIN A:SETUP" at the DOS prompt before entering Windows or, from within Windows, by selecting the **RUN** item in the **Program Manager File** menu and typing "A:SETUP" as the file to run. If you installation disk is not in drive A, then replace the letter "A" with the letter corresponding to the appropriate drive. From Windows 95, start the My Computer facility by double clicking on the **My Computer** desktop icon, select the disk drive that contains the installation disk by double clicking on the drive icon, then double click on the Setup icon. Alternatively you may select **Run** from the **Start** menu button, then **Run** from the list, then enter "A:setup" in the dialog box that appears.

When SETUP.EXE has started, a dialog box will ask for a directory for the EMUL68-PC Windows software. Either accept the suggested directory (C:\WIN68) or enter a different one. SETUP will copy files from the floppy to the hard disk directory specified and modify the configuration information stored in the file EMUL11.INI as needed. For windows 3.1 users, the installation program will install the WIN68 icon in the NOHAU program group. Double clicking on the emulator icon will start the emulator application. You may move the icon to another group if desired by using the **Move...** menu item in the Program Manager's **File** menu or alternately by dragging the icon to the new group.

For Windows 95 users, the setup program will add a **NOHAU** menu selection item to the **Programs** list available from the **Start** button. The **NOHAU** menu selection will open a sub-list of available Nohau applications programs. Click on the **WIN68** item to start the EMUL68-PC Windows application. In other words, click **Start**, select **Programs**, select **NOHAU**, click on **WIN68**.

Configuring The Software

This section describes selections available in the **Config** menu. Use these menu items to examine the software configuration in detail and to change it as needed.

Copyright © 1997 Nohau Corporation

Projects

A project is a collection of software configuration settings that are associated with a specific person, target, or software development project. The **Config | Projects** menu item opens a dialog box that allows you to set up named configurations or projects. This is first in the menu and described first because all of the other **Config** menu item settings will be stored as settings for the current project in a file with an extension of ".PRO". There is an ".INI" file called EMUL11.INI and those settings are used if there is no current project. But if the file EMUL11.INI contains the name of the current project, all software settings are taken from ".PRO" file for that project.

Projects behave differently than say, a word processing document. All software configuration settings are written to disk every time you change projects or whenever you exit the emulator software. There is no "exit without saving changes" option. Once you make a change to the configuration, it is immediately effective and will, unless you manually undo the change, be saved to the disk in the project file.

Creating a Project

Users who change the software settings and THEN change the name of the project may believe the old project will remain unchanged. In fact, the moment a new project is created, the current settings will be saved to the old project, not the new project. The new project will be saved when exiting the debugger or when changing projects (again).

Copyright © 1997 Nohau Corporation

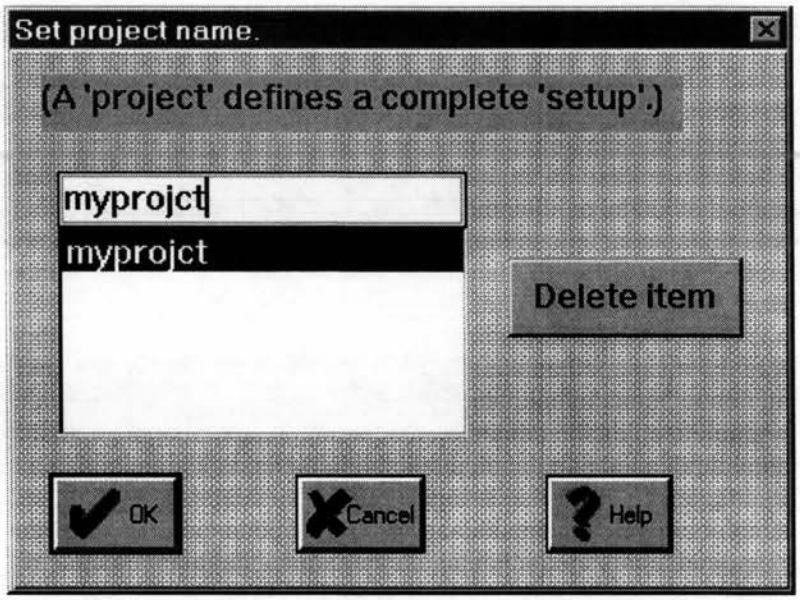


Figure 2-1: Project Name Dialog Box

To add a project, open the **Set Project Name** dialog box and type the new project name over the current name. Because the project name is used as the root of a file name, do not use characters in the name that cannot be used in a file name (like a space character). The new project will inherit all the settings from the old project. Projects are deleted by highlighting the project name you want to be deleted and then clicking on the Delete item button.

Note: The sample project "myprojct" is used throughout this manual.

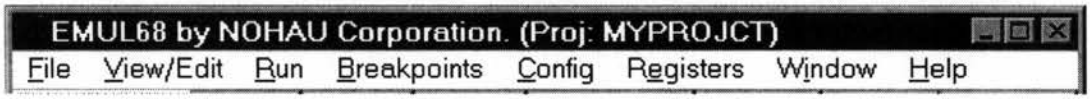


Figure 2-2: EMUL68-PC Title Bar

The name of the current project appears in the main window's title bar, which appears at the very top of the WIN68 background window. *See "Figure 2-2: EMUL68-PC Title Bar".*

Setting The Paths

The next item in the **Config** menu is **Paths** .. which opens the dialog box shown in "Figure 2-3: Paths Dialog Box". The emulator uses the paths identified in this box to find the files it needs.

Each of these fields can hold up to 1024 characters and contain multiple specific paths that will be searched in the order they appear. Multiple paths in one field must be separated by a semi-colon (";") similar to the MS-DOS PATH environment variable.

Copyright © 1997 Nohau Corporation

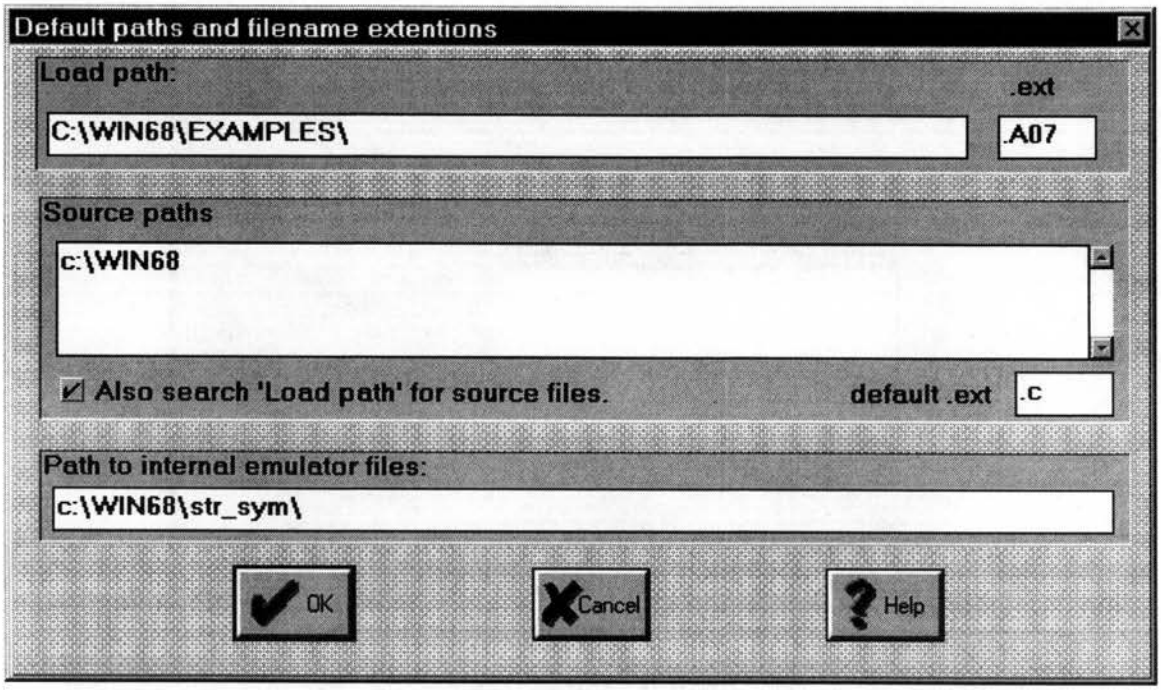


Figure 2-3: Paths Dialog Box

Object Files

The **"Load path"** will be searched for Motorola S record files and binary object files to be loaded into the emulator. The field labeled **".ext"** specifies the default file extension for the files that will be listed in the Load code dialog box. Each time a file is loaded, this field is changed to the extension of the file that was last loaded.

Source Files

With many compilers, the full path name of each source file is contained within the binary output file produced by the linker. If path names appear for source files in the binary object file that EMUL68-PC Windows is loading, the debugger will first look in the path specified in the binary file for source files when updating the Source window.

The second field, **"Source paths:"** identifies other directories to search for missing source files not identified in the object file or files moved since the compile. The directories in this field must be entered by the user. Once entered, directories will stay here until removed by the user. The small check box, when checked, will tell WIN68 to look for source files in the **Load path:** directory as well. Simple projects may have all the source and object files in the same directory (the **"Load path:"**) and may not need any directories in the **"Source paths:"** field.

*Note: The "**.ext**" field specifies the source file extension. If your C modules have the extension ".c", enter that. To see assembler source (.asm) in the **Source** window, enter ".asm" .*

Copyright © 1997 Nohau Corporation

Emulator Files

The **"Path to internal emulator files:"** field will contain the path where the emulator's internal program and data files (.STR and .SYM) are installed. By default, this field is set during the installation process to the STR_SYM subdirectory below the directory where WIN68 is installed on your hard disk and will probably not need to be changed. If you copy or move the EMUL68-PC Windows software to a new directory or drive, you must change this field accordingly to reflect the new location of the .STR and .SYM files.

Mapping Memory

ROM and RAM memory on the target can be emulated by RAM on the emulator board called emulation RAM or emulation memory. The entire 64K address space of the HC11 processor can be mapped to either the target's or emulator's memory in 4K byte blocks for the standard emulator board, or in 64 byte blocks for the bankswitched emulator board.

The **Config | Memory map..** menu item opens the Memory Map Configuration dialog box that controls the mapping of addresses to emulator RAM or target space. If a particular address is mapped to emulation RAM on the emulator board, all memory cycles at that address will reference emulation RAM in the emulator and the user's target board will see an ECLK that is low for the entire bus cycle, and thus will not respond during that cycle. Conversely, for a bus cycle at an address mapped to target memory, the target will see a normal ECLK but the emulator's memory will see an ECLK that is low for the entire cycle and thus will not respond. If your target has a memory mapped I/O device within a block mapped to emulation RAM, such mapping will prevent your application from accessing that device. To avoid this, be sure to map memory blocks that contain target devices to the target.

Copyright © 1997 Nohau Corporation

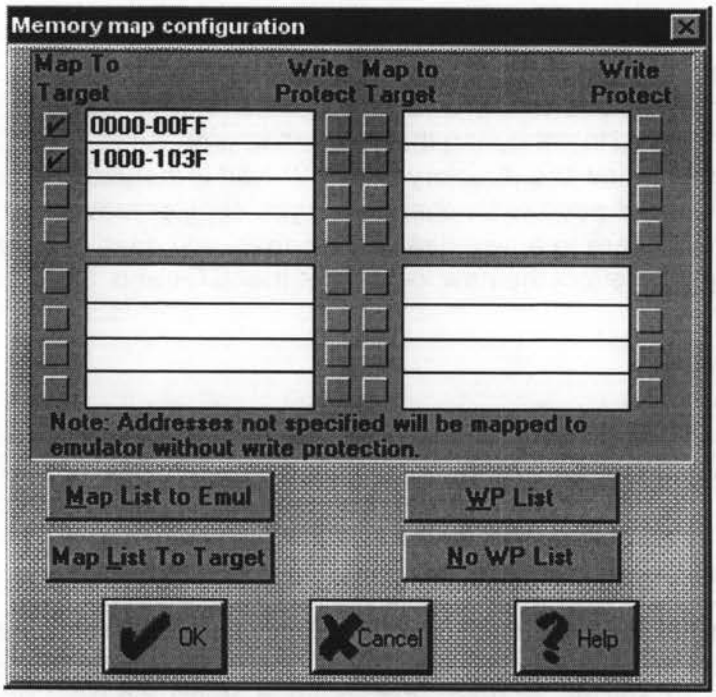


Figure 2-4: Memory Map Dialog Box

The standard emulator divides memory into 4K byte blocks; the bankswitched emulator divides memory into 64 byte blocks. The block size is not adjustable. Each block can be mapped to either emulation (default) or to target memory. Blocks can also be write protected.

Addresses are entered in hexadecimal as a range using the form: *START-END.* For each range, checkboxes indicate if that range is to be mapped to emulator or target memory and whether or not the range is to be write protected. By default, memory is mapped to emulator without write protection. Thus it is only necessary to list blocks in the Memory Map Configuration dialog box which must be mapped to target and/or require write protection. All blocks not listed in this dialog box will be mapped to the default condition - emulator memory, without write protection.

Individual blocks can be write protected. If a write cycle occurs to any address within a write-protected block, an immediate emulation break will occur and a warning box will appear to notify the user that the program attempted to write to a write protected address. This feature is useful to detect program errors whereby the processor attempts to write into program code or data stored in EPROM. Since the interface software cannot load programs into a write protected block, the write protected blocks should be protected after program loading is completed.

Copyright © 1997 Nohau Corporation

For your convenience, these four "speed" buttons are provided:

Maps all listed address ranges to emulator.

Maps all listed address ranges to target.

Write protects all listed blocks.

Removes write protection from all listed blocks.

When this dialog box is completed to your satisfaction, click on the OK button.

Note: If you use the bankswitched emulator board, after pressing OK to exit the memory configuration dialog box it is necessary to reset the emulator (by opening the emulator hardware configuration dialog and clicking on OK), reload your application and reinstall all existing breakpoints after clicking on the OK button.

Setting Up Bankswitching

The Banking Setup selection on the **Config** menu opens the Bank Switching Logic dialog box. This dialog box enables the user to define the bank switching configuration of the target so that EMUL68-PC can successfully emulate it. This box does not apply if you have the standard (non-bankswitched) emulator board.

Copyright © 1997 Nohau Corporation

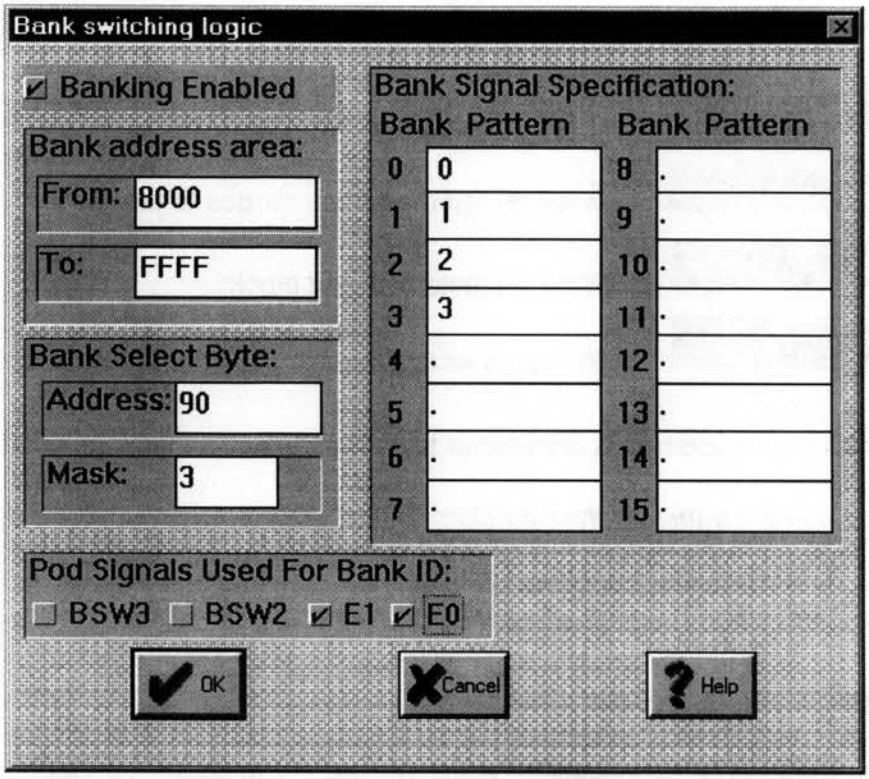


Figure 2-5: Bankswitching Dialog Box shown with typical 4 bank setup

The "Banking enabled" checkbox is used to enable bankswitching. If this box is not checked, then only 64K of non-banked emulation memory is available (similar to a non-bankswitched emulator) and the remaining fields on this dialog box are meaningless.

The "Bank address area" region of the dialog box allows you to define the range of the banked area within the 64K address space of the HC11 processor. The "From" and "To" fields specify the lowest and highest address of the bank area, respectively, in hex. Enter values from 0 to FFFF. For example, if you have a 16K bank area starting at 8000 you would enter 8000 in the From field, and BFFF in the To field. The bank area must start on an 8K boundary and must be a multiple of 8K in length in the range from 8K to 56K.

The "**Bank Select Byte**"[1] region allows the user to define the address and individual bits whereby the user's program selects a particular bank. This feature allows the user to display the contents of an arbitrary bank that was not selected when emulation stopped if the bank is located in target memory. These should only be set to non-zero values if all or part of the bank switched memory area is in target memory.

The four checkboxes under "Pod Signals Used for Bank ID" are to identify which of the four possible bank signal inputs on the pod are actually connected. Check the boxes associated with the signals on the pod that are connected to your bank select lines. Chapter 9 of the (DOS) EMUL68 User's Manual describes how the

[1] Note: This feature is not implemented in the current software release. Set both the Address and Mask fields to zero for correct operation.

Copyright © 1997 Nohau Corporation

user must connect from 1 to 4 digital signals to the pod in order for the emulator to be able to correctly select banks in its own emulation memory while the user's program is running. These signals are called BSW0, BSW1, BSW2 and BSW3, from lowest to highest order, and you must connect enough signals to uniquely identify each bank. For example, if you have only 4 banks then you would need only 2 signals. BSW0 and BSW1 must connect to pins labeled E0 and E1 that appear on every EMUL68-PC pod. The two high-order signals, BSW2 and BSW3, are required if you have more than 4 banks and are usually soldered to pins 38 and 29, respectively, on the pod's 50-pin cable connector.

Note: If you use a POD-11KE, POD-11NE or POD-11PE and you have more than 4 banks, then signals BSW2 and BSW3 may be connected to external pins on the pod labeled B0 and B1, respectively, instead of soldering the signals to the cable connector. Pins B0 and B1 on the pod are routed to connector pins 38 and 29 respectively, obviating the need to solder one or both high-order signals to the connector.

The "Bank Signal Specification" area contains 16 fields, one for each active bank. The term logical bank number is a value that appears in bits 16-19 of an address in the binary file loaded into WIN68 using the program load facility. A 20-bit logical address is a 4-bit logical bank number and a 16-bit address. All addresses shown in the WIN68 interface are logical addresses. The term physical bank number is defined as a 4-bit value that appears on the 4 bank select lines connected to the pod. Under most circumstances, physical and logical bank number are equal and that greatly simplifies working with the emulator. It is strongly suggested that you try to arrange for this to be the case. If not, the "Bank Signal Specification" area allows you to define the precise relationship between logical and physical bank numbers. Numbers in the shaded area under the heading "Bank" are logical bank numbers. For each of the 16 possible logical banks, there is a corresponding field under the heading of "Pattern" where you can enter the associated physical bank number i.e. the pattern that appears on the 4 bank select signals attached to the pod. Please enter the pattern in hexadecimal. Only those bits that are actually used are significant and bit positions corresponding to unused signals should be set to zero. The pattern fields for unused logical banks should be set to the "period" ('.') character.

Note: For the current release of WIN68 <u>when used with the bankswitched emulator board</u> whether or not banking is enabled, after opening the Memory Map Setup dialog box and subsequently closing it by clicking on the OK button you must do a full emulator reset, reload your program and reinstall all breakpoints. This is required even if you didn't change the memory mapping before clicking on OK. If you closed the memory map setup screen by clicking on CANCEL, then no action is required. A full emulator reset can be performed by doing one of the following: a) exit the interface software and restart it, b) open the emulator hardware configuration dialog box and then click on OK, or c) open the emulator hardware configuration dialog box and click on Reset Emulator, then click on OK or CANCEL.

Copyright © 1997 Nohau Corporation

Emulator Hardware Configuration

The **Config | Emulator Hardware** menu item opens the hardware configuration dialog box, which configures the software to correctly control and communicate with the hardware.

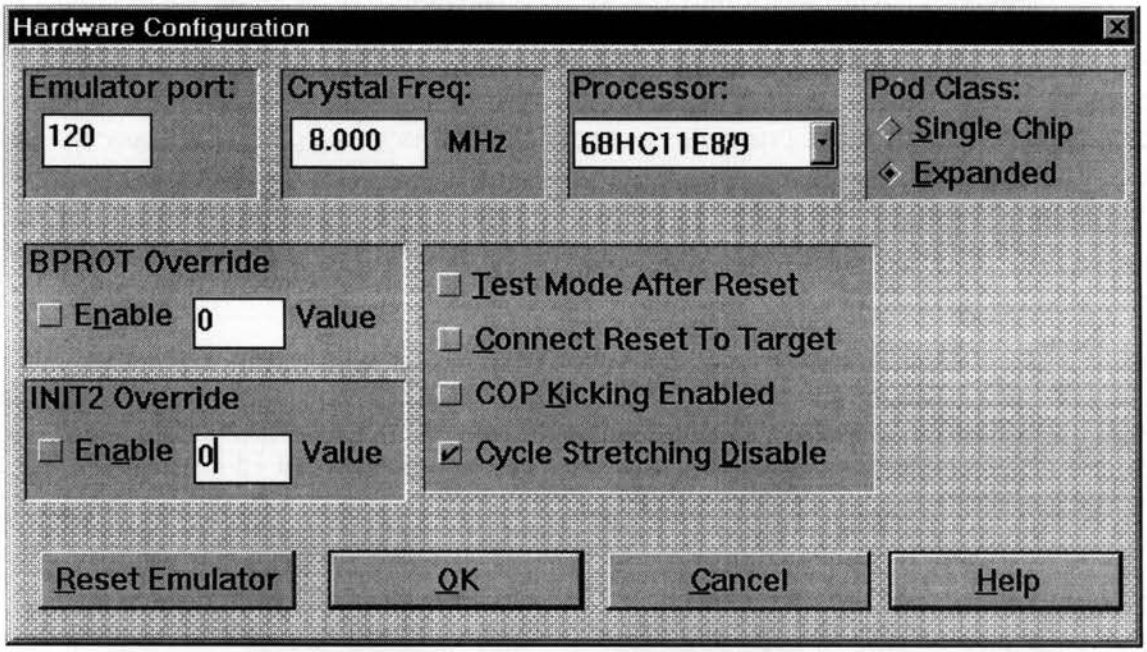


Figure 2-6: Hardware Configuration Dialog Box

The **Emulator port** field contains the hexadecimal address of the emulator in the PC's I/O space. The value entered in the Emulator Port field must agree with the jumper settings for I/O port address on the emulator board. If they do not agree, the Win68 software will not be able to communicate with the emulator board and a warning will automatically be displayed as a reminder that communication has failed and some change is needed. Please refer to Section 2 of the EMUL68-PC User's Manual for a detailed description of how to set the I/O port address jumpers.

Note: There are several other reasons why the PC may not be able to communicate with the emulator. Check that the cards, cable, and POD are all connected correctly and securely. If you check these and the jumpers on the emulator board and they are consistent with the Emulator Port: field on the Hardware Configuration dialog box and you still get error or warning messages, please call Nohau technical support for assistance.

The "**Crystal Freq:**" field specifies the frequency in Megahertz of the crystal being used. It must correspond to the crystal frequency you are using. This field does not refer to the ECLK frequency, which is the crystal frequency divided by 4. If you change the pod's crystal selection jumpers you may need to change the value in this field. The value defaults to 8.000 MHz corresponding to the crystal shipped with most pods. The pod crystal is selected with the jumpers in the "internal" positions.

Copyright © 1997 Nohau Corporation

The **Processor:** field chooses from a long list of supported controllers. Select the entry for the HC11 processor installed in the pod. See the POD Chapter in the EMUL68-PC User's Manual for a description of the various pods, their jumpers, and special instructions.

- If you have purchased a POD-11KE or POD-11KS and have replaced the chip in the processor socket on the pod with a socket adapter containing an HC11KA4 chip, then you should select 68HC11KA0-4, not 68HC11K0-4 which would apply if you had not changed processors.
- Users of pods POD-11PS and POD-11PE should take care to use one of the two selections: 68HC11P2 or 68HC11PA/PH8, not 68HC11K0-4 even though the pod shows "POD-11KE" or "POD-11KS". The 11PE/PS pods are modified versions of the KE/KS pods.

Additional Options:

Pod Class selects between E (expanded mode) and S (Single chip mode). Expanded mode refers to using designated processor port pins for address and data bus signals. In Single chip mode, these pins are used for I/O functions and there is no external address / data bus. The E / S is contained in the middle of the pod part number, i.e. POD-11**E**-PLCC-3.0. This setting must agree with the type of pod you are using.

BPROT Override when checked will attempt to write the specified 8 bit hex value into the processor's BPROT register within the first 64 cycles after a reset caused by the emulator. This feature applies only to processors that have an internal EEPROM and a BPROT register to control it, such as the F1, K4 and P2.

Note: Because of the default cycle stretching enabled after reset in expanded mode for the K family, the emulator is not able to write to the BPROT register for the K family within the first 64 cycles. To change EEPROM values from the emulator for the K family, start the processor in test mode by clicking on the Test Mode After Reset checkbox. The BPROT override option will work in test mode as the 64 cycle limit is not imposed. You will then be able to modify EEPROM by entering a new value for EEPROM locations that appear in a DATA Window.

INIT2 Override field only applies to those processors that have an INIT2 register such as the K, KA, N and P families. If your program writes to the INIT2 register to move the locations of the internal EEPROM, then enable the **INIT2 Override** feature by clicking on the **Enable** checkbox. Also enter the value that your program writes to the INIT2 register in the **Value** field.

Test Mode After Reset when checked places the processor in Test Mode after a Reset. If the box is not checked, the processor will enter expanded (normal operating) mode. Note that if you have a single-chip mode pod such as POD-11S, POD-11KS, etc., the processor actually runs in expanded mode with the lost ports reconstructed using either a Motorola-supplied part such as the 68HC24 port replacement unit, or a gate array.

Connect Reset To Target when checked connects the processor's RESET pin to the target's reset signal via a CMOS switch. If the box is checked, any reset

Copyright © 1997 Nohau Corporation

initiated by the emulator will cause the RESET signal on the target to be driven low (active), and vice-versa. If you have an external watchdog device on the target which will drive reset low when not serviced regularly, Nohau recommends that you do not enable this option. This will prevent the processor from being forced into RESET after the emulator stops at a breakpoint leaving the emulator in a confused and unusable state.

COP Kicking Enabled when checked requests the EMUL68-PC to regularly kick (reset) the HC11's internal COP circuitry while in **monitor** mode. **Monitor** mode is defined as the state the emulator is in when not executing the user's program, i.e. "stopped" as opposed to "running". If this box is not checked, then COP kicking in monitor mode is disabled. If the COP on the HC11 is enabled by writing a zero to the NOCOP bit in the CONFIG register, and this box is not checked, then the emulator will not function when the chip is operating in expanded mode. (This is not a problem in test mode because the COP is then automatically disabled even if NOCOP is set to zero.) It is recommended that this box be checked when using a 'D' family processor, or whenever the NOCOP bit in the processor's CONFIG register is set to zero.

Cycle Stretching Disable when checked disables cycle stretching in processors that support it (All K family parts except KA. The KA family does not have this feature and if you have replaced the K processor in a POD-11KE or POD-11KS with a KA part, this checkbox does not apply). After reset in expanded mode, K family processors effectively double the length of each bus cycle that accesses memory outside the chip. Such cycle stretching can be disabled by writing a zero to the clock stretch control register, CSCSTR at location $5A. The emulator will work with cycle stretching enabled, but response time for operator commands and emulator initialization are effectively doubled. This may result in uncomfortable delays depending on the emulator type and crystal speed.

Note: For the current release, if you use a POD-11KE with a K0, K1, K3 or K4 processor in the pod (as is the default shipping configuration) and enable bankswitching, you must check the "Disable Cycle Stretching" checkbox in the Emulator Hardware Configuration dialog box.

*Note: For 68HC11K0-4 users only: When checked, the emulator will write a zero to CSCSTR to disable cycle stretching after each reset that it initiates with one important exception, the **Reset and Go!** selection on the Run menu.*

OK button resets the emulator and trace boards completely and closes the dialog box.

Reset Emulator also resets the emulator and trace boards completely but without closing the dialog box.

Miscellaneous Configuration

The **Miscellaneous** item in the **Config** menu opens a dialog box that controls special features of EMUL68-PC Windows:

1. when and if automatic resets occur.

Copyright © 1997 Nohau Corporation

2. optional startup address and stack pointer values, that would override values set during reset.
3. the source code address range for limiting where breakpoints are set.
4. tab size for expanding tabs in the source window

By default, the emulator resets the controller when the EMUL68-PC Windows software is started and after an object file is loaded. The **Reset chip at start up:** and the **Reset chip after load file:** radio buttons can disable either of those resets and may be helpful during particularly difficult or unusual debugging circumstances.

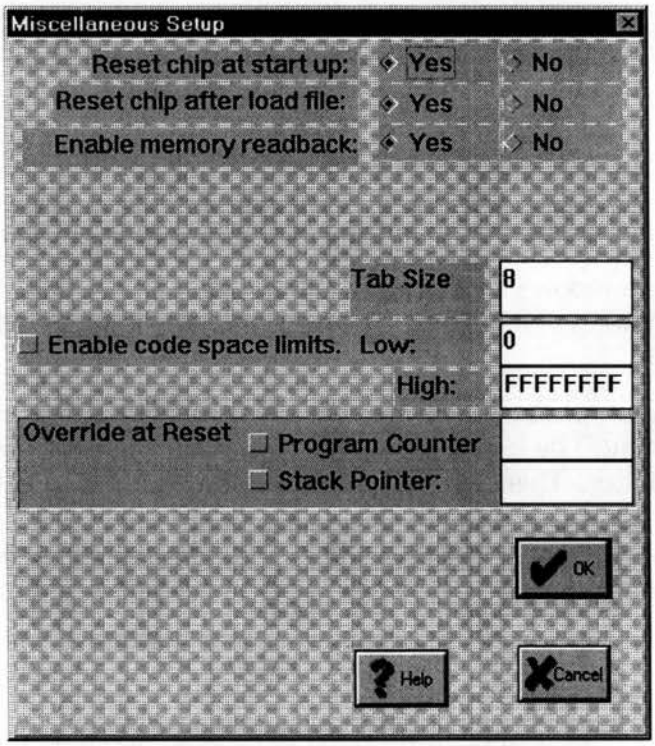


Figure 2-7: Miscellaneous Setup Options

Enable memory readback: when enabled, requests the software interface to redraw any memory dump window (with updated values) after a value in that window has been changed by the user, effectively verifying writes. (To change a value in a memory dump window, select the value to change using the mouse or arrow keys, type a new value, and press <Enter>.) Disabling this feature can be useful to prevent reading a location, where reading causes an undesirable side effect. The **Write without readback** item in the **View/Edit** menu performs the same operation.

Tab size: specifies the number of spaces that a tab character expands to, when displaying source code in the source window. The default is 8 spaces.

Enable Code Space Limits when checked will not allow breakpoints to be declared outside of the defined address range. Enter the high and low addresses in hexadecimal.

Copyright © 1997 Nohau Corporation

(Override at Reset) Program Counter: when checked will load the specified address into the processors' Program Counter when the emulator resets the device. If you have patched your startup code or if you want to only test a certain subroutine, the ability to reset the chip and start at an arbitrary address can be quite helpful

(Override at Reset) Stack Pointer: accomplishes the same function for the stack pointer. Be certain to enter a valid RAM address for the stack if this feature is enabled.

Window Colors

The **Config | Colors** dialog box allows you to customize the appearance of EMUL68-PC Windows to suite your tastes. For example, all Program windows can be set to have a dark blue background with white text to differentiate them from other kinds of windows. At the same time, all Data windows can be green with Black text, and all Source windows set to have white background and red text. It is possible to make the screen quite attractive, or conversely, garish enough to keep others from looking over your shoulder while you're working.

Choose the window class you wish to modify from the drop list. While that class name is showing in that field, the colors you select will be assigned to that class of windows.

After setting all of the colors the way you want them, you can name your creation ("color scheme") by typing the name in the **Color scheme** field and then clicking the **Save** button. This color scheme can then be recalled in the future.

Note: Not all combinations of background and foreground colors are possible due to constraints imposed by MS-Windows and your video configuration.

Menus

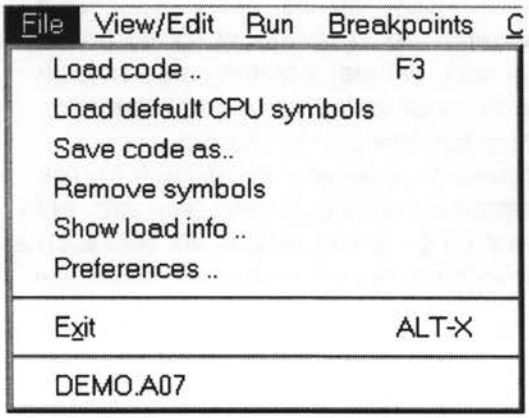
The primary means of controlling the debugger, and thus emulation, is through menus. EMUL68-PC Windows menus conform to the Microsoft MDI standard. Only those menu items that have meaning or can be used with the current selection will highlight when the mouse is pointing to them. Menus are organized to hide items that are out of context. Menu items which are not available in the current context will be grayed out.

Most menu items have "Hot Key" equivalents. That is, there is some combination of function keys, character keys, and modifier keys (Control, Shift, or Alt keys) to select most menu items. The Hot Key for each menu item is shown in that menu to the right of the item name, and are also shown below. Where you see "<Alt>RF" as the keyboard shortcut, you should type <Alt>R (hold the Alt key down while you then press the R key) to open the Run menu, then press the F key (without the Alt key) to activate the feature of EMUL68-PC Windows that begins emulation without enabling defined breakpoints. Holding down the Shift key or turning on CapsLock is not necessary. Even though the keyboard shortcuts are all shown in capital letters, the shortcuts are not case sensitive. Control key combinations are shown as either <Ctrl>X (press and hold the Ctrl

Copyright © 1997 Nohau Corporation

key while pressing the 'X' key) or alternately "^X". Function keys are shown as <F1>, other keys such as the "Delete" key are shown by <Del>.

File Menu



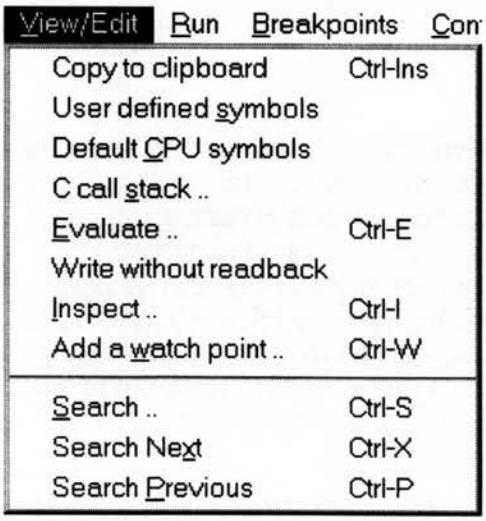
The **File Menu** controls the loading and saving of code (your executable), the loading of symbol information used in debugging (your module, function, and variable names, etc. and their locations). You may also set your preferences for how this information is loaded. At the bottom is a short list of most recently loaded files in order from most recent to least. This list allows rapid access to the files you are currently working with.

Figure 2-8: File Menu

Menu Commands	HotKey	Description
Load Code	<F3>	Loads a supported binary, hex (Intel or Motorola), or object file. Loading of code and symbolic information is controlled by the **File I Preferences** menu item. See "whatsnew.txt" file for supported object formats.
Load Default CPU Symbols		Note: This selection is not required in the current release to access the default CPU symbols. The user should ignore this selection.
Save Code As		Writes memory contents to disk in Motorola S1 file format.
Remove symbols		Deletes symbolic information from the symbol table stored in emulator memory.
Show Load Info..		Displays summary information for the file loaded last.
Preferences..		Allows user control over file loading. The default is to load code and symbolic information using Motorola formats for integers and floats.
Exit	<Alt>X	Exits the emulator application after saving project information.
(recently used file list)		This list shows the most recently loaded files. Clicking on the file name will load the file for debugging.

Copyright © 1997 Nohau Corporation

View/Edit Menu



The **View/Edit** menu allows quick inspection of variables, the C language call stack (where am I and how did I get here), the user's symbol list, or the processors default symbol (internal I/O or control registers) values. You may also change the value of variables or add them to a watch window for monitoring. An expression evaluator is built in for quick programming calculations. You may also search the current window for text such as variable names or values, an instruction, etc.

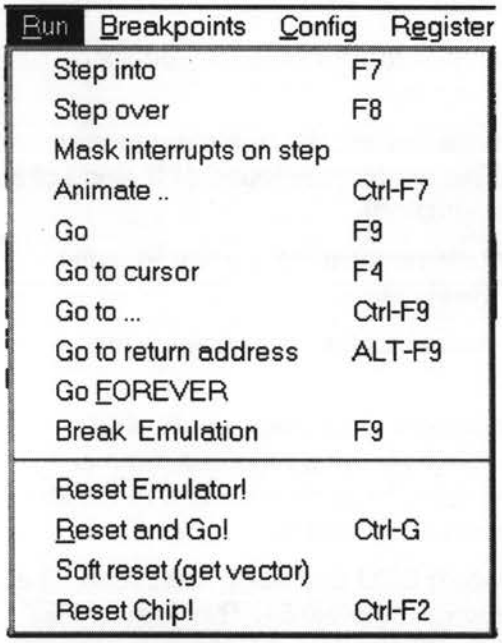
Figure 2-9: View/Edit Menu

Menu Commands	HotKey	Description
Copy to clipboard	<Ctrl> <Ins>	Copies the text (without font or formatting information) of the entire active window to the Windows clipboard.
User defined symbols		Opens a dialog box that allows selecting a source module for which all symbolic information may be viewed and optionally copied to the clipboard.
Default CPU symbols		Displays internal CPU I/O and control registers symbolically .
C call stack		Displays the C call stack and passed parameters required to reach the current Program Counter value (program address). Only one copy of this window is allowed open at one time.
Evaluate	<Ctrl>E	Opens a dialog box that evaluates C syntax expressions. Expressions may contain arithmetic operators ('+', '-', '*', '/'), variable names, indices, address operator ('&'), and structure operators ('.', '->'). Assignment expressions such as *"var=33"* may be used to change the value of a variable, including structure and array members.
Write without readback		Opens a dialog box that writes a value to a memory address but does not perform a read-back to verify the action. Normally values written to memory are read back for verification. This feature prevents errors if "poking" a value to a peripheral device or register that does not return the identical value just written when read.

Copyright © 1997 Nohau Corporation

Inspect	<Ctrl>I	Open a dialog box that displays the contents of a single variable, structure, or array in detail. An inspect window can also be opened by double clicking on the name of a variable or structure member in the source window.
Add watch point	<Ctrl>W	Open a child window that displays groups of variables that are updated every time emulation stops.
Search	<Ctrl>S	Allows searching the active window for the "kind" of data displayed in that window.
Search Next	<Ctrl>X	Find the next occurrence of the search string, from the cursor forward.
Search Previous	<Ctrl>P	Find the next occurrence of the search string, from the cursor backwards.

Run Menu



The **Run** menu is the "control center" for the emulator. It provides several different ways to reset the emulator, the microcontroller being emulated, or both. It also controls starting and stopping of emulation as well as single stepping through your code. Many of the important features of this menu are available through "hot keys" - offering speedier access to these functions which you will use often.

Execution is always at full speed with the exception of the **Animate** selection which runs the user application program at reduced speed to make program flow more visible.

Figure 2-10: Run menu

Menu Commands	HotKey	Description
Step into	<F7>	Single step, including calls and jumps. In a Program window this executes one instruction. If a Source window is selected, this executes all of the instructions for one source line of C.
Step over	<F8>	Similar to "Step into" above, but executes functions at full speed and breaks at the next instruction or source line following the call. When "stepping over" at the assembly language /

Copyright © 1997 Nohau Corporation

		instruction level, it is important that the subroutine actually return to the address immediately following the JSR instruction or emulation will continue forever. [2]
Mask interrupts on step		When checked, the effect is to mask interrupts during single stepping to prevent (unintentionally) stepping into interrupt code when an interrupt is pending and enabled. This flag has no effect while running.
Animate..	<Ctrl> <F7>	Continuously single steps instructions or source lines, updating all open windows after each step. In the program window, it steps an instruction at a time. In the Source window, it steps a source line at a time.
Go	<F9>	Begin execution from the current PC until a breakpoint is reached.
Go to cursor	<F4>	Begin execution from the current PC to the current cursor position or until a breakpoint is reached.
Go to..	<Ctrl> <F9>	Execute from the current PC to the specified address.
Go to return address	<Alt> <F4>	Execute from the current PC to the next return instruction. (This is normally found at the end of a subroutine or function). [2]
Go FOREVER		Execute instructions from the current PC after disabling all breakpoints.
Break Emulation	<F9>	Suspend execution as if a breakpoint was encountered.
Reset emulator!		Resets the emulator to its initial state, which includes resetting emulator registers, memory mapping, and reloading the .STR and .SYM files. All user symbols are deleted.
Reset and Go!	<Ctrl>G	Reset (hardware) CPU and begin execution. The reset lasts approximately 0.5s. This form of "Go" should be used initially if your code writes to any registers that are locked after the first 64 E-clock cycles in expanded mode.
Soft reset (get vector)		Load the reset vector into the program.
Reset Chip!	<Ctrl> <F2>	Reset CPU without executing any instructions. This has the same effect as pressing the reset button on the pod.

[2] Some compiler optimizations may prevent this feature from being available during portions of program execution.

Copyright © 1997 Nohau Corporation

Breakpoints Menu

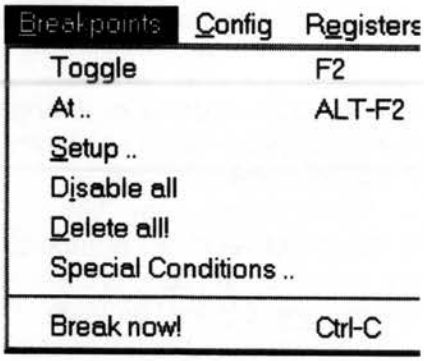


Figure 2-11 Breakpoints menu

The **Breakpoints** menu allows you to set and reset breakpoints in your code. There are two fully independent mechanisms for defining, deleting, and toggling, breakpoints. For this reason, breakpoints are divided into two classes: Class 1 and Class 2. Class 1 breakpoints are set and reset using a variety of techniques in the Program and Source windows. Class 2 breakpoints are only available using the **Breakpoints I Special Conditions** dialog box. Both are implemented in hardware and thus will work for code executed out of external EPROM or EEPROM. This menu is very useful to disable or delete all Class 1 breakpoints at once.

Class 1 Breakpoints

Class 1 breaks are defined, deleted, or toggled between active and inactive states by clicking on an address in the program window, clicking on a line number in the source window, or using any selection on the **Breakpoints** pull-down menu except **Special Conditions**. A Class 1 break is an opcode fetch type break at a single address, usually the first byte of an instruction opcode. An opcode fetch type break is active when LIR/ is driven low as the processor fetches the first byte of an instruction[3]. On this type of break, emulation always stops before the instruction at the breakpoint is executed. The **Breakpoint I Disable All**, **Breakpoint I Delete All,** and **Breakpoint I Setup** (dialog box) menu items only operate on Class 1 breaks, and not on Class 2.

> *Hint: To quickly toggle a Class 1 breakpoint active / inactive using the mouse: In a Program window, click in the address field. In a Source window, click on or to the left of the line number on any executable line (you can only break on lines which produce code, i.e. you cannot break on a comment!).*

Class 2 Breakpoints

Class 2 breaks are defined, toggled, or removed only through the **Breakpoint I Special Conditions** dialog box. The methods described above for Class 1 breaks do not work for setting, clearing, or toggling Class 2 breakpoints. Class 2 breaks are more flexible than Class 1 breaks in that they can specify an address or address range and one of six combinations of cycle type and the signal on the E0 pin. Class 2 breaks will cause emulation to stop before the instruction at a breakpoint is executed only if the breakpoint specifies an opcode fetch type

[3] This is the type of break defined using the current EMUL68 DOS software when the user presses F10 or clicks the mouse on a source line or disassembled instruction in the code window

Copyright © 1997 Nohau Corporation

breakpoint and if it is detected during the fetching of the first byte of an instruction. Note that only Class 1 type breaks are highlighted in the Source and Program windows. See the description of the Special Conditions dialog box on page 2-44 of this manual for further information.

The address sub-field of a Class 2 break can specify a single address or a range of addresses, and can be expressed in one of three acceptable forms as shown below. The terms "*expr1*" and "*expr2*" represent C expressions involving operators, symbols and/or hexadecimal constants. There are two other key words permitted; "to" is used to describe a from/to range between two numbers, and "len" used to describe a length. No spaces are permitted in the expressions between symbols, operators and constants, but spaces are required on each side of the words "to" or "len".

1. *expr1* a single address specified by *expr1*
2. *expr1* **to** *expr2* a range from *expr1* to *expr2*
3. *expr1* **len** *expr2* a range from *expr1* of length *expr2*

The rules for the use of symbols in address expressions is as follows.

1. The name of a C function represents the address of the first instruction that function generates.
2. If you enter the name of a C or assembly language variable, the ampersand symbol (&) must precede the name to obtain its address.
3. If the symbol name appears without a preceding ampersand, then the symbol's current value at the time of evaluation will be used.
4. Expressions are evaluated each time the **Special conditions** setup is modified.

Note *It is important to remember that if you use a symbol name in an expression, the value of the symbol used in the expression is evaluated each time the **Special conditions** dialog box is closed by clicking the **OK** button. When that occurs, it is possible for the value of a previously defined expression which evaluates to a range or location to also change.*

For example, suppose that *func1()* and *func2()* are functions in your program, and *foo* is a variable. The following examples will help to clarify the rules.

Example #	Expression in address field:	Definition of expression:
1	1000	a single address at $1000
2	func1 to &foo+1	a range of addresses from the first address of *func1()* to the address of (*foo* + 1 location)
3	func1 len 1000	a range of addresses from the first address of *func1()* to the address defined by (*func1()* + $FFF)

Copyright © 1997 Nohau Corporation

4	func1 len foo	a range of addresses from the first address of *func1()* to the address defined by (*func1()* + the value of *foo-1*)
5	&foo	a single address, the address of variable *foo*
6	func1 to func2-1	a range of addresses from the first address of *func1()* to the address one before *func2()*. Assuming that *func1()* and *func2()* were adjacent in memory (compiler dependent, see your map file) this would evaluate to contain all of the addresses of *func1()*

Menu Commands	HotKey	Description
Toggle	<F2>	Enable or disable a Class 1 breakpoint at the cursor.
At..	<Alt> <F2>	Set a Class 1 breakpoint at an address. You are prompted for the address.
Setup..		Opens a dialog box that allows set / reset or deletion of Class 1 breakpoints.
Disable all		Disables all Class 1 breakpoints.
Delete all!		Deletes (clears) all existing Class 1 breakpoints.
Special Conditions		Manipulates Class 2 breaks only. Allows specification of the address range, bus cycle type, and external E0 signal as part of the break condition. *Note that breakpoints set in this dialog box do not show up in other dialog boxes and are not highlighted in Program or Source windows.*
Break now!	<Ctrl>C	Stops execution immediately. Also available as a button on the tool bar.

Copyright © 1997 Nohau Corporation

Config Menu

The **Config** menu is used to tailor the emulator to your specific hardware setup and project preferences. The Project name and Paths selections are used to tell the emulator software what files you are using and where to find them. The Memory map, Banking Setup, and Emulator Hardware selections are used to configure the emulator hardware to match your environment. The trace and PPA (Program Performance Analyzer) selections are discussed in detail in Chapter 4: **Trace Board**. The **Config** menu is discussed in detail at the beginning of this chapter.

Figure 2-12 Configuration Menu

Menu Commands	HotKey	Description
Project name		Create a new project or select from a list of existing projects. Projects store software configuration settings and definitions. The last project opened is reopened by default when WIN68 is started.
Paths..		Set default paths for finding object, source, and emulator files.
Memory map..		Opens a dialog box for assigning memory space in blocks to either emulation RAM or the target.
Banking setup..		(Applicable to bankswitching emulators only.) Configures the emulator banking scheme, sizes, hardware, etc.
Convert cycles to time!		This toggles between displaying the number of elapsed cycles, or alternately the elapsed time the emulator has "run" code at the right side of the toolbar. The displayed time is an arithmetic conversion of the measured E-cycle count.
Emulator hardware..		Sets up the emulator hardware. See "Emulator Hardware Configuration" on page 2-10 at the beginning of this chapter for more detailed information.
Miscellaneous		Set up miscellaneous default conditions. See the "Miscellaneous" section on page 2-13 for more information.
Color..		Allows the user to configure the window colors to suit his / her tastes.
Trace..		*(Only applicable if optional real-time trace card*

Copyright © 1997 Nohau Corporation

installed.) Opens a dialog box to set trace conditions. See Chapter 4: **Trace Board** for further details.

PPA.. *(Only applicable if optional real-time trace card installed.)* Opens a dialog box to setup Program Performance Analyzer. See Chapter 4: **Trace Board** for further details.

Local Menus

Local menus are a special type of menu that is specific to the window currently selected, such as the Program window, Source window, Data window, etc. The next position on the menu bar is reserved for the local menus. The menu and sub-menu in this position will change depending on which window is currently selected. There are nine local menus corresponding to the Program, Source, Data, Registers, Special registers, Watch, Inspect, Trace buffer, and Call Stack windows.

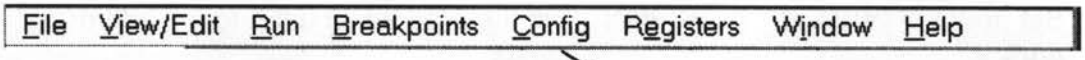


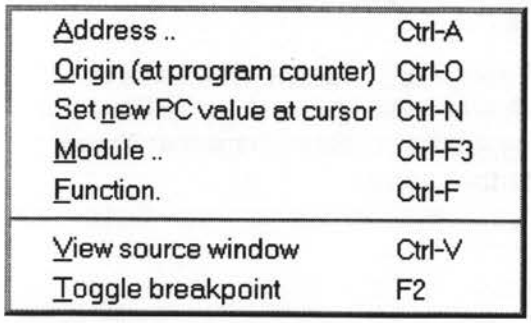
Figure 2-13: Menu bar - local menu positions

Local menus are very handy - they offer commands specific to the current window that are most often used. They are a shortcut to working with the data displayed in the window. While many of the commands available in local menus have shortcut keystrokes also, the right mouse button is a handy way to access these features until the shortcuts you use the most have been memorized. Just point the mouse cursor to the desired window and right click!

To select a window (thereby choosing the local menu displayed) use the "Window" selection from the menu bar (explained below) with the arrow keys or simply click the mouse in the desired window. A left click will select the window, a right button click will select the window and automatically open the local menu applicable to that window. A right button click is the shortcut method to open a local menu.

Copyright © 1997 Nohau Corporation

Local Menu - Program window

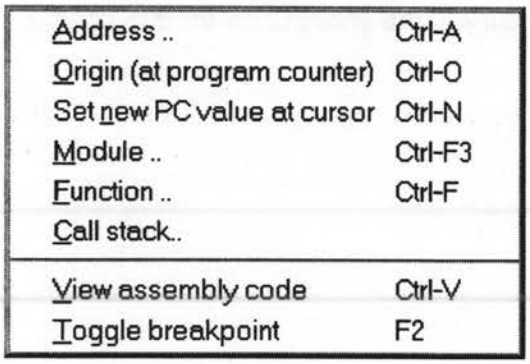


Available when the **Program** window is selected.

Figure 2-14: Local menu - Program window

Menu Commands	HotKey	Description
Address..	<Ctrl>A	Position the window to the specified address.
Origin (at program counter)	<Ctrl>O	Position the window at the current program counter value.
Set new PC value at cursor	<Ctrl>N	Set the program counter to the address at the cursor.
Module..	<F3>	Opens a dialog box that allows quickly positioning the cursor at the start of any module.
Function..	<Ctrl>F	Opens a list of all functions loaded, selecting one immediately positions the cursor at the function.
View source window	<Ctrl>V	Opens or shifts focus to a Source window.
Toggle breakpoint	<F2>	Sets / resets a breakpoint at the cursor.

Local Menu - Source window



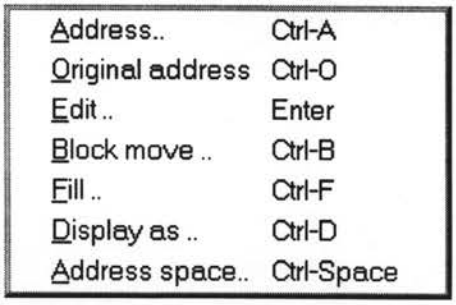
Available when the **Source** window is selected.

Figure 2-15: Local menu - Source window

Copyright © 1997 Nohau Corporation

Menu Commands	HotKey	Description
Address..	<Ctrl>A	Position the window to the specified address.
Origin (at program counter)	<Ctrl>O	Position the window at the current program counter value.
Set new PC value at cursor	<Ctrl>N	Set the program counter to the address at the cursor.
Module..	<F3>	Opens a dialog box that allows quickly positioning the cursor at the start of any module.
Function..	<Ctrl>F	Opens a list of all functions loaded, selecting one immediately positions the cursor at the function.
Call stack		Displays the C call stack and passed parameters required to reach the current Program Counter value (program address). Only one copy of this window is allowed open at one time.
View assembly code	<Ctrl>V	Opens or shifts focus to a Program window and positions the cursor at the current program counter value (which may not be the code equivalent to the source line at the cursor).
Toggle breakpoint	<F2>	Sets / resets a breakpoint at the cursor.

Local Menu - Data window



Available when a Data window is selected.

Figure 2-16: Local menu - Data window

Menu Commands	HotKey	Description
Address..	<Ctrl>A	Position the cursor to the specified address.
Original address	<Ctrl>O	Scroll the selected Data window to the last address used in an **Address..** menu command
Edit	<Enter>	Edits the data under the cursor. You may alternately position the cursor over the data and begin typing a new value, a dialog box is

Copyright © 1997 Nohau Corporation

		automatically opened.
Block move..	<Ctrl>B	Moves a block of data specified by start address and either end address or length.
Fill..	<Ctrl>F	Fills a block of data specified by start address and either end address or length with a user supplied value.
Display as..	<Ctrl>D	Changes display format (via a sub-menu) of data to one of 13 supported formats including binary, hex, ASCII, integer, and float. A *custom format..* selection is also available which allows the user to enter a C format specifier using standard printf() syntax.
Address space..	<Ctrl> <space>	Allows user to switch between display of DATA (default), Special Function (CPU) Registers (where data window address 0000 is first SFR address, regardless of SFR offset in physical address space), and BREAK RAM reserved for Nohau diagnostic use only.

Local Menu - Registers window

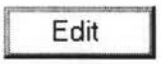
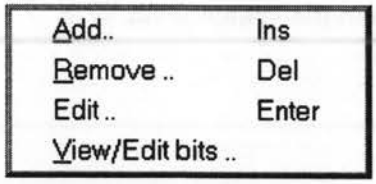


Figure 2-17: Local menu - Registers window

You may use this button to edit a register value in the register window, or simply highlight the value to change with the mouse and type in a new value - a dialog box will automatically open for entry.

Copyright © 1997 Nohau Corporation

Local Menu - Special Registers window



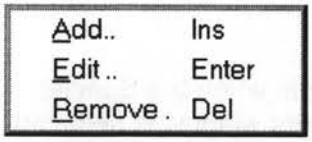
Available when the Special Registers window is selected.

Figure 2-18: Local menu - Special Registers window

Menu Commands	HotKey	Description
Add	<Ins>	Opens a list of special, I/O and control registers that may be added (by clicking) to the Special Registers window.
Remove..	<Del>	Removes a register from the SpecialRegs window.
Edit..	<Enter>	Opens a dialog box to modify the value of a register. Alternately, double click on the register name in the window.
View / edit bits..		Opens a dialog box to modify individual bits in registers that allow it.

Copyright © 1997 Nohau Corporation

Local Menu - Watch window



Available when the Watch window is selected.

Figure 2-19: Watch window local menu

Menu Commands	HotKey	Description
Add..	<Ins>	Add a variable to the Watch window. Watch variables are updated when the emulator stops.
Edit..	<Enter>	Modify a Watch variable. Alternately, double click on the variable in the Watch window or select the variable and then begin typing a new value to open the same dialog box.
Remove	<Del>	Remove a Watch variable from the Watch window.

Local Menu - Inspect window

This feature is not implemented in the current software release.

Copyright © 1997 Nohau Corporation

Local Menu - Trace window

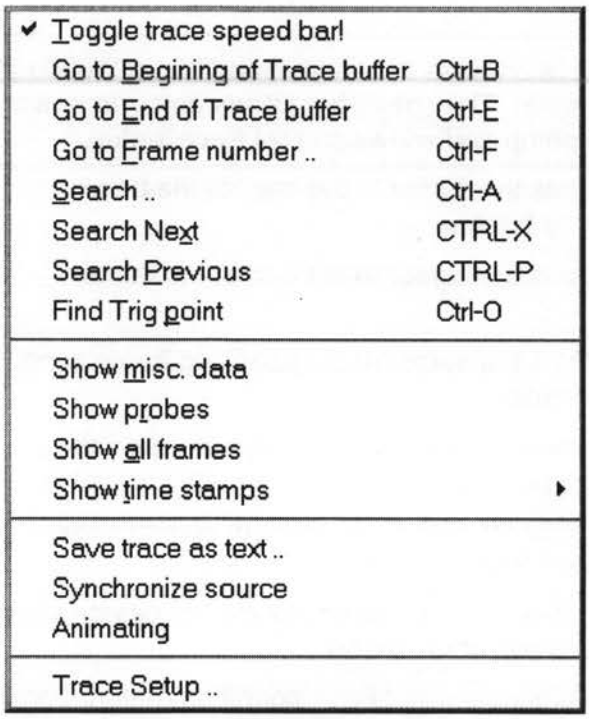


Figure 2-20: Trace Buffer window - local menu

Available when the Trace buffer window is selected. Refer to Chapter 4: **Trace Board** for additional information on items available in this local menu.

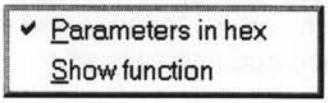
Copyright © 1997 Nohau Corporation

Menu Commands	HotKey	Description
Toggle trace speed bar!		Displays or hides the trace speed bar at the top of the screen. The speed bar allows quick access to trace setup and shows current trace status.
Go to beginning of trace buffer	<Ctrl>B	Positions the cursor to the start of the trace memory buffer.
Go to end of trace buffer	<Ctrl>E	Positions the cursor to the end of the buffer.
Go to frame number..	<Ctrl>F	Positions the cursor at the specified frame number in the buffer.
Search..	<Ctrl>A	Searches the trace buffer for an address, data, cycle type, or probe (usually an I/O port) value.
Search next	<Ctrl>X	Searches for next occurrence of search string (forward from cursor).
Search previous	<Ctrl>P	Searches for previous occurrence of search string (backwards from cursor).
Find trigger point	<Ctrl>O	Positions cursor at trigger condition (trigger frame) in buffer.
Show misc. data		Displays or hides *time stamp counter overflow* and E0/E1 pin data in frames.
Show probes		Displays or hides probe data in frames. Probes are by default connected to I/O ports, see pod documentation for details.
Show all frames		When *off* (default), instructions which produce multi-byte reads, writes, or fetches are shown on one line. When *on*, each frame (bus cycle) is shown as a separate line.
Show time stamps		Displays or hides timestamp information in frames. Timestamps may be displayed in Absolute cycles, Absolute time, Relative cycles, or Relative time from the sub-menu displayed.
Save trace as text		Opens a dialog box where you specify a file name and starting and ending frame numbers, then saves the selected portion of the trace buffer to the file in ASCII text format, suitable for printing.
Synchronize source		Synchronizes cursors in the Program and Source windows to the highlighted instruction in the Trace Buffer window.
Animating		Scrolls slowly through the trace buffer. This allows an "instant replay" of the execution history just captured. If **Synchronize Source** is checked, the

Copyright © 1997 Nohau Corporation

	Program and Source windows will track the instruction currently highlighted in the trace buffer window.
Trace setup..	Opens the Trace Setup dialog box. This dialog box may also be opened by clicking on **Setup** on the Trace Speed Bar.

Local menu - Call Stack



Available when the Call Stack window is selected.

Figure 2-21: Call Stack window - local menu

Menu Commands	HotKey	Description
Parameters in hex		When checked, all passed parameters are displayed in hexadecimal format. The default (unchecked) is to show parameters formatted by type - i.e. floats' as float, char as ASCII, int as hex.
Show function		Positions the cursor in the Source window to the highlighted function in the Call Stack window. Alternately double click on the function in the Call Stack window.

Copyright © 1997 Nohau Corporation

Window Menu

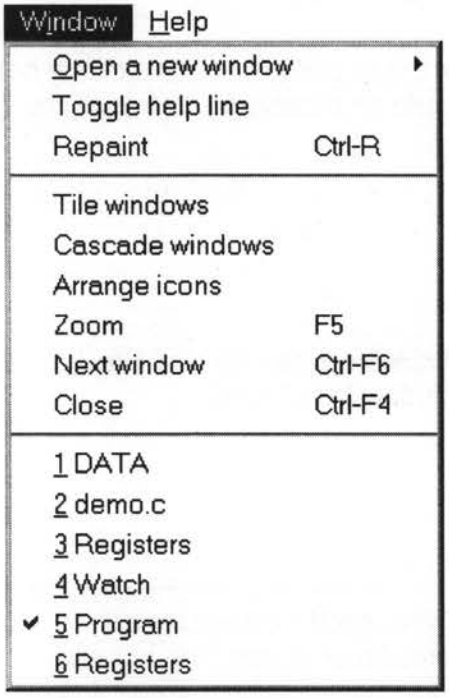


Figure 2-22: Window menu

The **Window** menu is used to manipulate the display. You may open new windows, close existing windows, select windows, and arrange windows on the screen. There are nine different types of windows available: Program, Source, Data, Registers, Special Registers, Trace Buffer, Watch, Inspect, and Call Stack. These windows are updated when emulation is stopped. Each window may be custom sized and have user selected colors. Window configurations are saved as a part of the project information automatically when Win68 is closed.

The *current window* is selected by clicking on the window with the mouse, typing <Ctrl><Tab>, or <Ctrl><F6>. You may have multiple windows of any type open at the same time, except the Call Stack window, only one of which is permitted. Windows may be easily sized by dragging with the mouse at the window corners or edges, or by using the minimize / maximize / restore icons in the title bar at the top of each window. Windows may be closed using the icons in the title bar of each window just like the windows in other MS-Windows applications.

Menu Commands	HotKey	Description
Open a new window		Opens a new window, the type is selected from the Open Window sub-menu (see Figure 2-23 below). The *Inspect,* and *Call Stack* windows are opened from the **View/Edit** menu.
Toggle Help line		Displays or hides the command help line at the bottom of the display.
Repaint	<Ctrl>R	Redraws the screen.
Tile windows		Resize and arrange the displayed windows without overlapping them.
Cascade windows		Resize and overlap the displayed windows.
Arrange icons		Line up any closed EMUL68-PC Windows icons at the bottom of the main window.
Zoom	<F5>	Expand the current window to fill the main Win68 background window.
Next window	<Ctrl>	Make the next window the current window.

Copyright © 1997 Nohau Corporation

	<F6>	Alternately just point and click with the mouse.
Close	<Ctrl> <F4>	Close the current window. Alternately use the icons in the title bar at the top of the window you wish to close.
...Open window list...		This list shows open windows with a check mark next to the current window. Selecting a window name from this list will restore it and make it the current window.

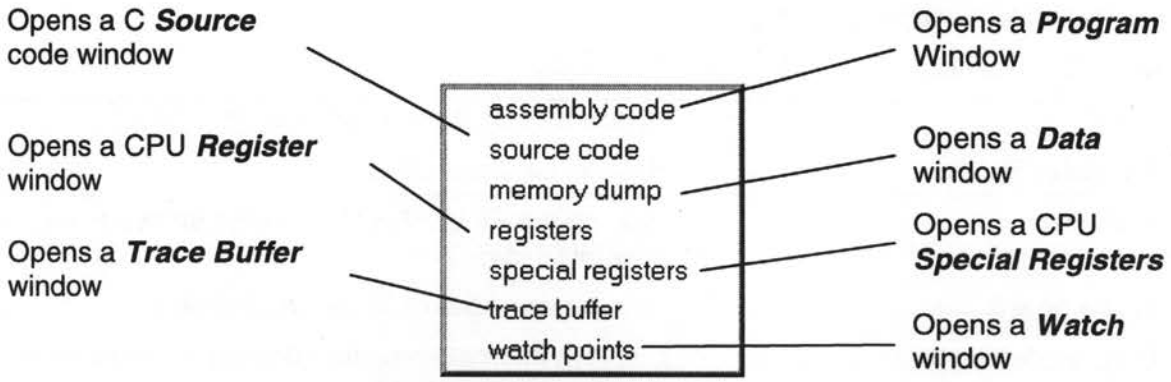
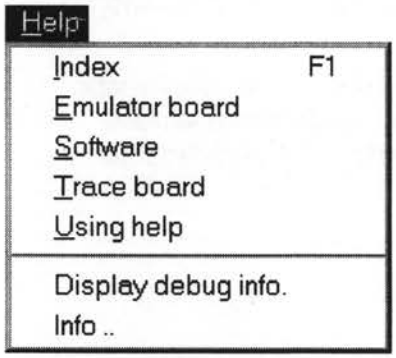


Figure 2-23: "Open Window " sub-menu

Copyright © 1997 Nohau Corporation

Help Menu



This menu provides on line help for commands, emulator setup, and emulator operation. Help is provided in standard Microsoft Windows help style and should be familiar to MS-Windows users. See the documentation provided with your version of Windows for further information on using the Windows help system.

Figure 2-24: Help menu

Menu Commands	HotKey	Description
Index	<F1>	Opens an alphabetical index of help topics.
Emulator Board		Help on emulator setup.
Software		Help on the EMUL68-PC Windows software and user interface.
Trace board		Help on using the optional Trace board.
Display debug info		Displays internal emulator information. For further details, see page 2-44.
Info		Displays the version number and date. Please have this information handy when calling Nohau for technical support.

Child Window Types - The Details

There are nine primary child windows created by the EMUL68-PC Windows application: Program, Source, Data, Registers, Special Registers, Trace Buffer, Watch, Inspect, and Call Stack. Seven are opened from the **"Window I Open a new window"** menu, the Inspect and Call Stack windows are opened from the **View/Edit** menu. (The Watch window may be opened from either the **Open a New Window** menu or the **View/Edit** menu.) Any number of child windows may be open at the same time. Any number of child windows can overlap but only one child window is active (selected or has-the-focus) at a time.

Each child window has a corresponding *local menu* whose name appears on the menu bar between the **Config** and the **Window** items. The local menu contains items that make sense within the context of that window. Following is a description of all local menus.

All windows may be resized and relocated as desired to fit your debugging environment. Some may be scrolled and resized to view any address desired. Using the **Config I Color..** menu you may change the color for each type of

Copyright © 1997 Nohau Corporation

window to suit your tastes. Window locations, sizes and colors are saved to the current project file when EMUL68-PC Windows exits, and will be automatically restored when the software is started.

Registers window

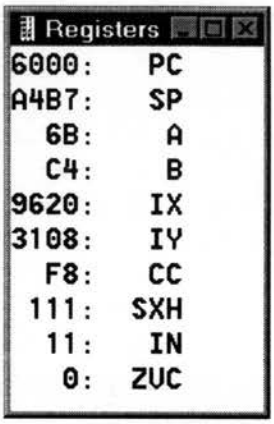


Figure 2-25: CPU Registers display

The Registers window displays the 68HC11 CPU registers in hex: Program Counter, Stack Pointer, (accumulators) A, B, Index X, Index Y, and Condition Codes. Also displayed in binary are the CCR flags Sign, X Interrupt Mask, Half-carry; I Interrupt Mask, Negative; Zero, Overflow, and Carry. Right-clicking brings up the Registers local menu. You may edit the value of a register by using the local menu, or highlighting a value and typing - a dialog box automatically opens. Register values that have changed since the last time the emulator stopped are highlighted.

SpecialRegs Window

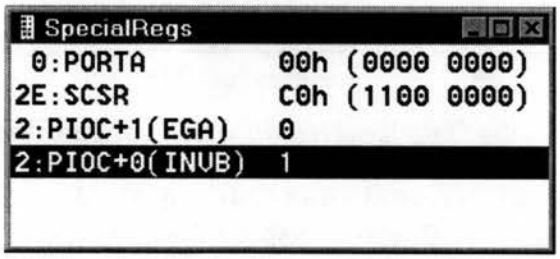


Figure 2-26: Special registers window

In addition to the Registers window, Win68 has a customizable window for special function registers - I/O ports, timers and other control registers. This window is empty when initially opened. The local menu for this window allows you to add registers to the window by picking from a list specific to the processor you are using.

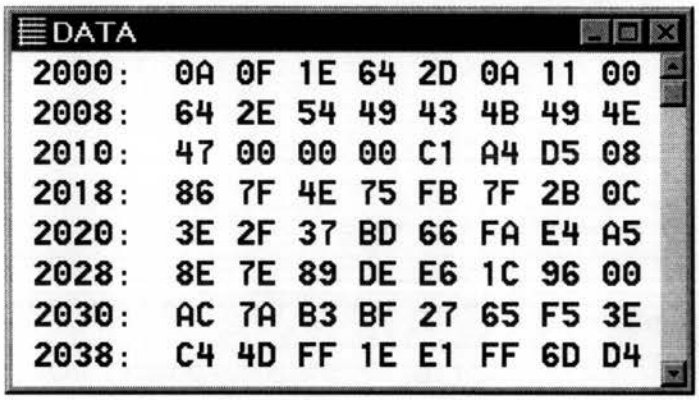
You may also edit or delete items in the window. You may add the entire register to the window, or in some cases just a single bit. The register or bit names placed in the window for display are saved as a part of the project file and restored when the emulator is started. Each eight bit register is displayed with its address, mnemonic, and current value in hexadecimal and binary. Individual bits are displayed in binary only. Note that the address is the offset from the start of the Special Registers space, not the absolute physical address. For instance the SCSR register is always shown as address (offset) $2E, although in many 'HC11 variants the physical address of this register is $102E by default. This is true even if the user has relocated the starting address of the register space by reprogramming the INIT register in processors that support this feature.

Copyright © 1997 Nohau Corporation

Shown in the example above are the following registers:

Register name:	Address	Value shown in example above
PORTA (I/O port 'A')	0000	$00
SCSR (Serial Comm. Status Reg.)	002E	$C0
PIOC (EGA bit) (Parallel I/O Control Reg.)	0002	0 (binary)
PIOC (INVB bit) (Parallel I/O Control Reg.)	0002	1 (binary)

Data Windows



The Data window provides a view of the 68HC11 memory space. Use Data windows to examine or modify emulation or target memory directly. Asterisks are displayed while the emulator is running and the window is updated when emulation stops. Data can be displayed or modified in various formats.

Right clicking any Data window displays the Data local menu, which supports the following items: memory fill, block move, positioning, address space selection, and display format. Display formats include hex (8, 16 and 32 bit), decimal, binary, ASCII, float, double, and custom. The **Custom Display Format** option opens a dialog box that lets you input a C printf() syntax format specifier string. All standard C formats are allowed, including the newline character.

Editing Memory with a Data Window

Changing a value at any memory location is as easy as selecting the byte, word, long word or float to change with the mouse or arrow keys and then typing the new value. The first character you type will open a quick data entry dialog. To avoid confusion, always enter the new data in the same format that the data is displayed. If the Data window is displaying ASCII characters, type the new character (not a string) in ASCII. If the Data window is displaying signed integers, enter the new value as a decimal number. The number of bytes written to memory corresponds to the displayed format. If you display the data in bytes, only a byte will be written to memory for each update. In other words, updating one byte uses a single bus cycle that is one byte wide. If you display signed decimal ints, two bytes are written for each value entered.

Copyright © 1997 Nohau Corporation

Symbols are supported but their type is ignored. You may use symbol names in some operations such as positioning the data window cursor. To locate the cursor to a (user) variable named "timer.sec" you would right click in the DATA window to open the local menu, then click on "**Address**", and enter "&timer.sec" in the dialog box. This would position the cursor to the first byte of structure element timer.sec.

EMUL68-PC Windows can access three address space types in the Data window: Data, SFR, and Break. Data space is where your code and user variables are stored. The SFR space contains the CPU Special Function Registers for control of on board peripherals such as I/O ports, timers, SCI, SPI, etc. The SFR space also contains the INIT, OPTION, and similar CPU control registers. When SFR space is displayed in the Data window it is shown relative to the current SFR starting address. That is, address 0 always refers to the first (lowest) SFR, regardless of the physical starting address of SFR space. Even if you relocate SFR space using the INIT register or you are using a CPU that maps the SFR space at a non-zero location by default, the SFR registers are always shown as beginning at 0 in the Data window when set to display SFR space. This feature displays SFR register addresses at the same locations referenced in the Motorola M68HC11 Reference Manual whether they have been relocated or not, saving the programmer the inconvenience of translating addresses when viewing the SFR's.

Program Windows

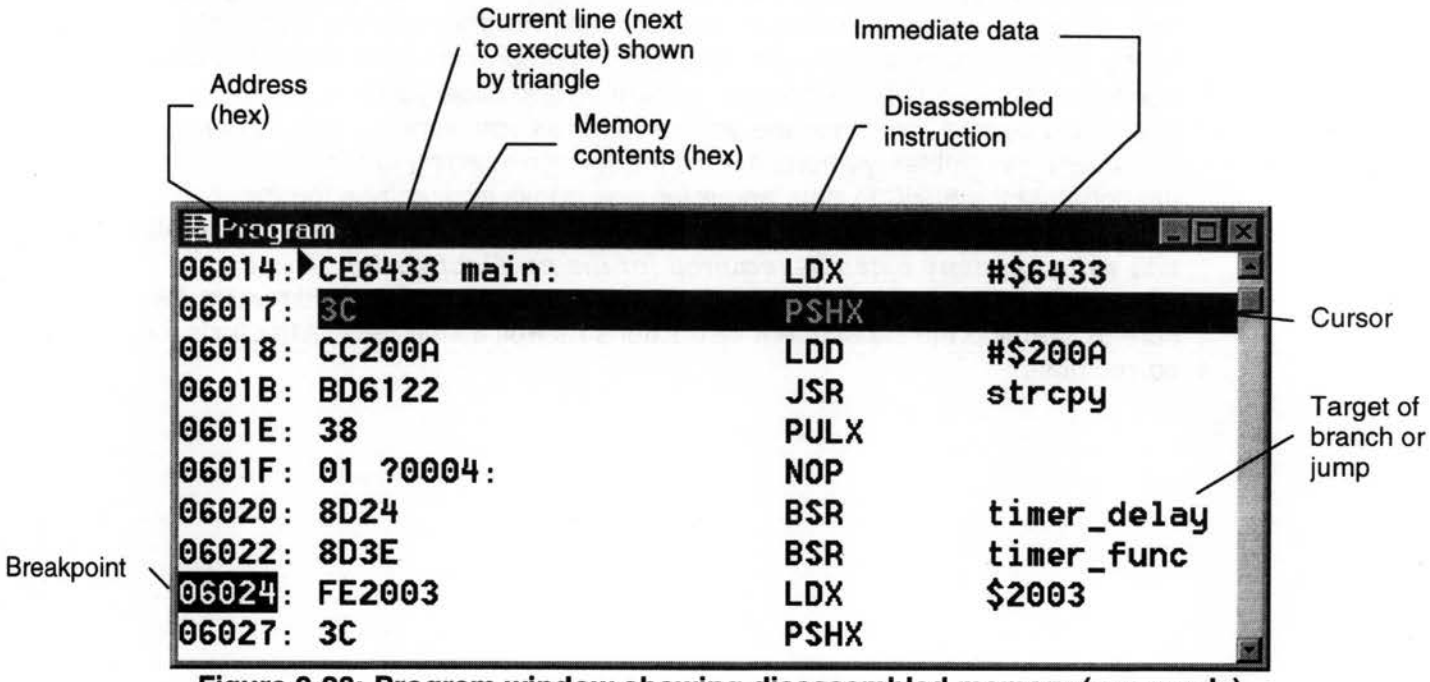


Figure 2-28: Program window showing disassembled memory (user code)

Copyright © 1997 Nohau Corporation

A Program window disassembles and displays code memory. It also features a built in assembler for quick program edits. The current line in the Program window is always highlighted. This is known as the cursor. Use the cursor to set and reset breakpoints, set the program counter, and invoke the in-line assembler. By moving the cursor to a line, then pressing <F4> (**Run IGo to cursor**), your program will execute to the cursor by automatically setting a temporary breakpoint at the cursor and running your program.

The first column is the hexadecimal address. If the address is highlighted (reverse video), there is a breakpoint at that address. The second column is the hexadecimal value contained at that address which may be a single or multi-byte opcode. Between the address and the hexadecimal data may be an arrow pointing to the right, indicating the current program counter. This is the next line to execute. The third column contains the disassembled instructions and operands. This is the actual disassembly of the current memory contents.

Program windows can control emulation. You may set or inactivate a breakpoint by clicking on the address. To set a breakpoint, click once on the address portion of the instruction where you want the break, the address will be highlighted. Clicking again will deactivate the breakpoint. Alternately, you may click once on the desired instruction (to highlight that instruction) and then click on it again to highlight the address. To delete the breakpoint (as opposed to disabling it), use the **Breakpoints Setup ..** menu.

In-line Assembler - The in-line assembler is easy to use - simply highlight the instruction or address you wish to change in the Program window and begin typing the new instruction(s) and data. The first character typed will automatically open an **Edit** dialog box to input your changes and allow you to edit your assembler source line. Once the source line is as you want it, press <Enter>. The in-line assembler will translate the input line according to the syntax described in the 68HC11 data and reference books and replace the former opcode(s) and data with the new opcode(s) and data. ***Note that the assembler will write as many bytes as required for the new instruction.*** This may overwrite part or all of subsequent instructions causing unexpected results. Be sure to examine the subsequent instructions as well as the new instructions for correctness.

Copyright © 1997 Nohau Corporation

Source Windows

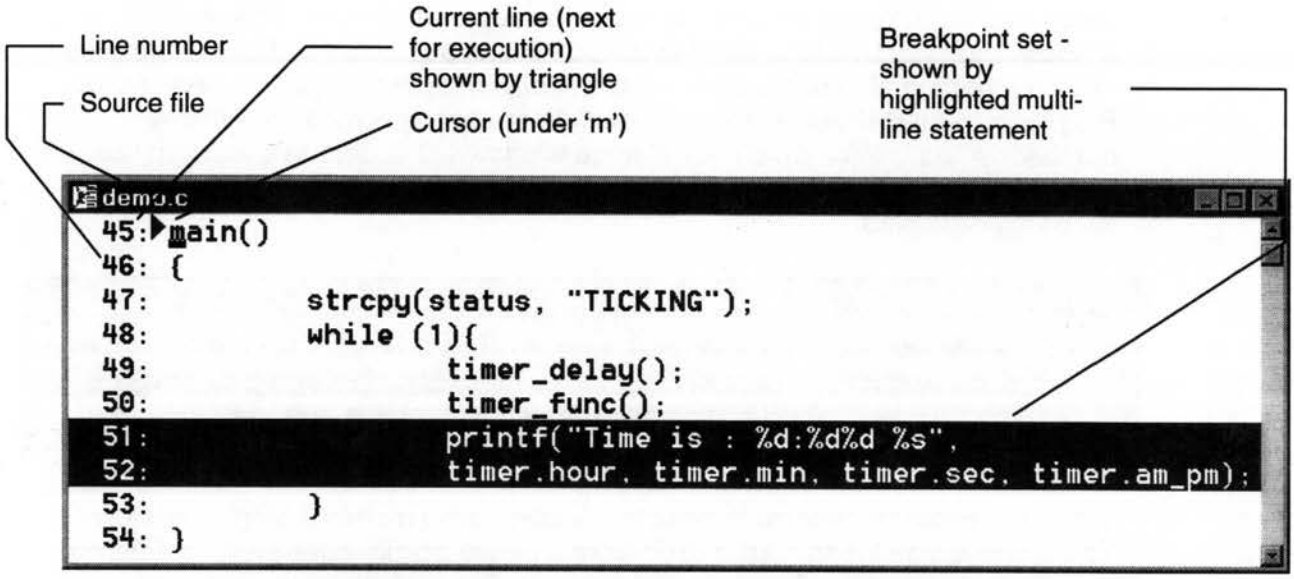


Figure 2-29: Source code window (C source code shown)

The Source window displays the C source code of the current module. The current module is the one into which the Program Counter (PC) currently points. Assemblers which support *Source Line Debugging* (many do not) may also have their assembly source code (not the disassembled code from memory as shown in the Program window) displayed in this window. Like a Program window, a Source window displays the source text, line numbers, a cursor (in this case a blinking underline), and a small arrow between the line numbers and the source text to indicate the current Program Counter value.

You may use the mouse or arrow keys to position the cursor, then use <F4> to execute your program to the current cursor position in the same way as in the program window. Breakpoints are indicated by highlighting (reverse video) the entire line(s). You may set and reset breakpoints by clicking on or to the left of the line number near the left border, or alternately, by pressing <F2> on the current line indicated by the underline cursor. You may only set breakpoints on lines which produce executable code - in other words you cannot set a breakpoint on a comment (only) line. Occasionally when you click on a line to set a breakpoint you will notice that several lines become highlighted as in the example above. This is because in some cases several lines of C code resolve to a single execution point in the program code. This is the case with multi-line statements and lines which produce no code, such as the label ***main()*** and the subsequent opening brace. An example of a multi-line statement is shown above in Figure 2-29 where the ***printf()*** statement is on two source lines but is actually a single C source statement terminated with a semi-colon.

After each single step, and during each animation pause, the Source window scrolls to show the source line that generated the instruction pointed to by the

Copyright © 1997 Nohau Corporation

new Program Counter, if it was generated by a source line. When a Source window appears blank with the window title "Source", it usually means that the program counter is pointing to instructions in your code derived from a module with no debugging information such as a module stored in a library supplied by the compiler vendor. The absence of debugging information can also be the result of compiling with incorrect compiler switches or options. The source window may also be blank when your code is stopped outside of any C modules - i.e. in a portion of your program written in assembly language. As soon as the PC points to an instruction from a C module or assembly module with line number symbolic information, the Source window will update to show that text, and the title on the window will change from "Source" to the name of the source file being displayed.

Hint: The simplest way to find the first line of source text is to reset the controller (use the Reset button on the Button Bar), click on the Source window title bar to select it, and then execute a single step by pressing the F7 key (or by clicking on the Step button on the speed bar).

When the Program window is selected, a single step means a single instruction. The same is true for animated execution: a pause occurs after every instruction is executed. When the Source window is selected, a single step means execute a single source line. Animation will execute faster when the Source window is selected than when the Program window is selected because most source lines compile into multiple machine instructions. If the animation is running faster or slower than you expect, or if single stepping executes more or fewer instructions than you expect, visually confirm that the selected window is the one you want to be selected. If in doubt about which window is selected, click on the title bar of the window you wish to be selected.

High Level Language Debugging Windows

Three more child windows used for high level debugging in C are available: the **Inspect** window, the **Watch** window and the C **Call Stack** window, . These windows are opened by selecting their respective items in the **View/Edit** menu. Like the other child windows, selecting or opening one of these windows will bring a corresponding unique local-menu selection up between the Config and Window menus. Local menus may also be activated with a right mouse click in the corresponding window.

Inspect Window

The **Inspect** window displays a single variable, and allows you to modify it. Inspect windows may be left open during emulation and are updated each time emulation stops. The variable being displayed may be part of an expression written using C syntax and following the rules of C that produces a single scalar answer or the name of a structure, union or array.

Copyright © 1997 Nohau Corporation

Note: If you have an open Inspect window with an expression that includes an assignment statement (i.e. var=0x33), then each time the emulator stops, the expression will be evaluated and the variable will be updated - var will be set to 0x33! Even if your application modifies the value when running, the variable will appear as though your application is not changing it.

There are three methods available to open an **Inspect** window:

1. Double click on the name of the variable in the Source window.
2. Press <Ctrl>I
3. Use the **View/Edit** I **Inspect** menu selection.

In the latter two cases, if the cursor in the Source window is at a variable name, an Inspect window will open with the variable at the cursor highlighted, press <Enter> to display it. If the cursor is not at a legitimate symbol name, the Inspect window will open and wait for you to type in the name of the desired variable, followed by <Enter>. Note that symbol names are case sensitive. If a symbol cannot be found, try the same name with a different case. Also note that some assemblers shift all symbols to uppercase, so you may need to enter the variable name in all uppercase letters even though it appears in the source window in lower or mixed case.

Hint: To inspect an entire structure, use the struct name without the '.' or '->' operators, i.e. time. *To inspect a single structure member, use the struct and member names, i.e.* time.minutes. *Likewise with an array you may inspect a single element or the entire array by using the array name with or without brackets and element reference. To quickly traverse a linked list of structures, double click on the pointer to the next element in the list. To view a structure, union or array within a structure, double click on it's name.*

Watch Window

The Watch window displays one or more variables and is very useful for examining variables throughout a debugging session. Only one watch window may be open at a time. The contents of the Watch window are updated each time the emulator stops. The Watch window displays one variable per line. Any local variable in the Watch window that is not in scope will be displayed with three question marks instead of its value.

To open a Watch window, use the **Window I Open a new window I watch points** menu selection (or the **View/Edit** menu, by selecting **"Add a watch point"**), then use the right mouse button to open a **Watch I local menu**, then click **Add..** and fill in the dialog box with a variable name. Alternately you may place the cursor on the desired variable and press <Ctrl>W, this will open the **Add..** dialog box automatically and place the variable in it, press the <Enter> key

Copyright © 1997 Nohau Corporation

or click"OK" with the mouse and a Watch window will appear. If a Watch window is already open, the specified variable will be added to it.

Watch windows have two important distinctions when compared with Inspect windows. The first is that Watch windows may have one or more variables displayed (in contrast to an Inspect window which is limited to a single variable), but do not permit expressions. The second important difference is that you may change the value of a variable in an Inspect window but Watch windows are for display only.

Call Stack Window

The Call Stack window displays the list of functions called to reach the current point in the application, and the current value of parameters passed to them. By default parameters are shown in the format appropriate for the variable type, chars are shown in ASCII, ints as decimal (signed or unsigned as required), and floats or doubles in floating point format. Using the local menu, you may select all parameters shown in hexadecimal format. Clicking on the local menu item **"Show function"** will position the cursor in the source window on the function body for the function highlighted in the Call Stack window.

Other Dialog Boxes and Windows

Evaluate dialog box

This dialog box provides a handy programming aid. It is opened from the **View/Edit** menu using "<Alt>VE", or by the shortcut key combination "<Ctrl>E". The dialog box evaluates C syntax expressions. Expressions may contain arithmetic operators ('+', '-', '*', '/'), variable names, indices, address operator ('&'), and structure operators ('.', '->'). Assignment expressions such as *"var=33"* may be used to change the value of a variable, including structure and array members. To use this as a quick calculator, do not enter an '=' (equals sign) - i.e. to subtract hex 20 from hex 7A, enter "0x7a - 0x20" and press <Enter>, the answer appears in the lower field. Numbers may be entered in any C format except float, only integers are evaluated.

Copyright © 1997 Nohau Corporation

Special Conditions (Class 2 breakpoints) dialog box

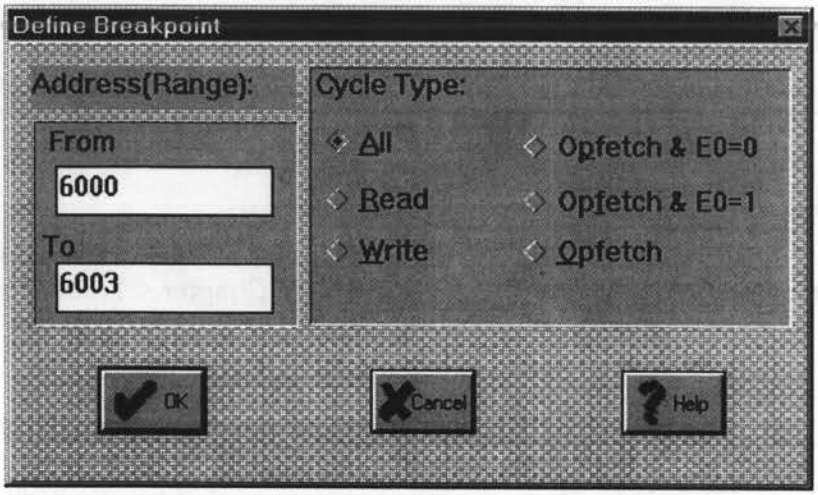


Figure 2-30: Special Conditions (Breakpoints) dialog box

This dialog box allows the user to define Class 2 (only) breakpoints. It is accessed from the **"Add"** button in the **Breakpoints I Special conditions** dialog box. The dialog box allows the user to enter an address range, or a single address when the From and To fields are set the same. The cycle types may also be specified, see below.

A break occurs on a bus cycle whose address falls within the specified range and the LIR\, R/W\, and E0 signals match the specified conditions. See page 2-20 for further information on breakpoint Classes.

Cycle Type:	Description:
All	Any valid bus cycle type (Cycle Type = don't care).
Read	Any valid Read cycle, which includes both data and opcode reads. Note that for multi-byte opcode fetches, only the first byte fetched is an "Opfetch" cycle, bytes 2+ and any data associated with the instruction such as immediate data are Read cycles.
Write	Any valid Write cycle.
Opfetch (& E0 = don't care)	An opcode fetch ("Opfetch") cycle is defined as active when LIR/ is driven low as the processor fetches the first byte of an instruction. For multi-byte instructions this is only true during the fetch of the first byte. This is similar to the action of a Class 1 type breakpoint.
Opfetch & E0 = 1	Same as "Opfetch" above AND the external signal E0 must be high. The E0 signal is available as a pin on most pods and is useful as a hardware qualification in addition to software conditions necessary for a break.

Copyright © 1997 Nohau Corporation

Opfetch & E0 = 0 Same as "Opfetch" above AND the external signal E0 must be low.

Trace Window

For information about the Trace window and using the optional trace board, please refer to Chapter 4:**Trace Board**

PPA Window

The Program Performance Analyzer requires the optional trace board. For information on using the PPA, please refer to Chapter 4:**Trace Board**

Debug Information

The Debug information dialog box (available from the **Help | Debug Information..** menu) displays internal emulator information. This information is provided primarily for Nohau's technical support staff to help in diagnosing problems you may encounter. The dialog box displays the following fields:

Field:	Contents:
STR version	The version number of the emulator's (internal) .STR file.
LUT	A lookup table used by the emulator to find memory blocks that have no internal memory.
Reserved	This field is reserved.
SFR	The range where the emulator has determined that the SFR's are located.
intRAM	The range where the emulator has determined that the internal RAM is located.
EEPROM	The range where the emulator has determined that the EEPROM is located. A range from 0 to 0 implies the EEPROM is not enabled.
INIT	The emulator's latched copy of the low-order 4 bits of the INIT register (which define the 4 high- order address bits of the SFR registers locations).

Button Bar

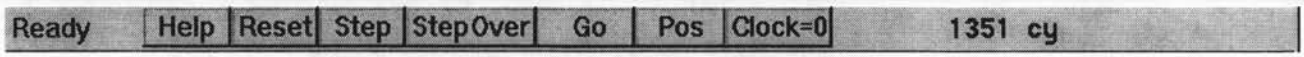


Figure 2-31: Emulator Button (Tool) Bar

Just below the menu bar is the "Tool Bar" containing icons or buttons that, like Hot Keys, execute frequently needed menu options when clicked. At the far left the current emulator status is shown. At the far right is the cycle count which is the total number of E-clock cycles since emulation was started or the cycle

Copyright © 1997 Nohau Corporation

counter was reset by the user. This count may be changed to display elapsed time rather than cycles by choosing the **"Config | Convert cycles to time"** menu option. The displayed time is computed from the cycle count and depends on the user correctly setting the **"Crystal Frequency"** field from the **"Config | Emulator hardware"** dialog box.

Button	Function
Help	This button opens the MS Windows Help application to the page with help for the current Win68 context.
Reset	This button resets the microcontroller. Same as **Run	Reset chip**.
Step	This button emulates one source line or one instruction depending upon which window was last active, Source or Program respectively.
Step Over	This button is the same as Step, except that interrupt handlers and functions called are run at full speed, with emulation stopping at the following source line in the Source window or instruction in the Program window (depending on which window is selected).
Go	This (toggle) button starts full speed emulation that will continue until a break occurs. After starting emulation, the **Go** button changes to **Break.**
Break	This (toggle) button halts emulation. When emulation is stopped, this button becomes the **Go** button.
Pos	This button acts just like typing <Ctrl>A and will open a dialog box that lets you jump the window cursor to a specified address.
Clock = 0	Resets the cycle count (or displayed time). Resetting the microcontroller (see above) also resets the count.

Copyright © 1997 Nohau Corporation

Help Line

At the bottom of the Win68 window is a line of text that, depending upon the context, explains what the selected item under the mouse pointer is, what it does, or keystrokes most often used in the active window. This context-sensitive help is turned on and off with the "**Window | Toggle help line**" selection. You can get a small amount of displayable area back with this function disabled after you become familiar with emulator's operation.

Trace Speed Bar

Figure 2-32: Trace Speed Bar for rapid access to trace setup and display

The Trace Speed Bar allows quick access to the most common trace functions. The trace board is optional and for further information about the trace card see Chapter 4: **Trace Board**. This bar is enabled by using the local menu item "Toggle trace speed bar" from the Trace Buffer window's local menu.

At the far left of the bar the current trace status is shown (in the example shown, tracing is ***Stopped***). This indicates the status of the trace board (i.e. tracing, stopped) and not the emulator. With tracing started, you must also run the emulator to collect information. In the rightmost box is shown the **frames** counter, this is the number of frames collected that meet the trace filter conditions. Next to this are setup parameters **trig delay** and **loop count**. See the chapter on the trace board for more details on these parameters.

The **Trace** button starts and stops the trace feature. Tracing may be started and stopped without the need to break emulation.

The **Setup** button brings up the **Trace Setup** dialog box which is covered in detail in Chapter 4: **Trace Board**.

Copyright © 1997 Nohau Corporation

Chapter 3: Emulator Board

Please refer to the EMUL68™-PC (DOS) User's Manual for details on the emulator hardware.

Copyright © 1997 Nohau Corporation

Chapter 4: Trace Board

Trace Board Introduction

The EMUL68™-PC emulator provides visibility into your code and the CPU. It also gives you the ability to inspect, watch, and modify memory, set breakpoints and view peripherals, giving you total control over the debugging process. However, it does so only when emulation is stopped.

The EMUL68™-PC optional trace board enhances these abilities by providing the ability to capture bus transactions in real time. It captures a snapshot of your code as it executes showing such details as: where your code was when an interrupt occurred, the results of interrupt processing, multiple interrupts of the same kind (such as all of the characters in a serial packet), interrupt execution order, updates to variables, execution times, histograms of time spent in specific sections of code, probe (I/O port) values, and loop iterations. It does all of this in real-time as your code executes at full speed. Moreover, the trace conditions can be changed, new data captured, and the trace buffer viewed - all without breaking emulation, allowing multiple "peeks" into your code while it is running non-stop! The trace can free-run or be triggered by events in your code and then capture *n* cycles before or after the trigger event. It can be setup to capture all bus cycles, or just ones meeting certain user criteria (filtering). You can find difficult-to-diagnose problems such as stack overflow, random writers, null pointer assignments, interrupt conflicts, and failures that only occur at-speed, with relative ease as compared to tracking down these bugs without trace.

The EMUL68™-PC Windows trace provides RAM and logic to capture frames of information. Frames (trace records) are 64 bits wide; the trace card captures either 4,096 (TR4) or 16,384 (TR16) frames depending on the model. Each frame records 16 address bits, 8 data bits, 16 probe bits (which are jumper connected to I/O ports on most pods), 4 miscellaneous bits that capture cycle type, 16 bits of time stamping, and 4 bits for internal use. The captured frames provide a history of your code execution and the binary levels on the probes.

Detailed Installation Instructions

Installation Requirements

> ***Warning: Turn off power to the target and the PC before installing the trace board.***

The emulator board has the logic and connector necessary to support a trace board. No additional connection to your target is required. The trace board is a full-length ISA-style bus card. It may occupy any available 8-bit slot as long as the ribbon cable can reach from the emulator card to the trace card. A block of

Copyright © 1997 Nohau Corporation

16 consecutive I/O addresses (see below) is required. No hardware interrupts or DMA is required.

The trace board typically requires 1.3A from the PC's 5 volt supply. Before proceeding, check that the PC's 5V power supply is sufficient (most are) to deliver the necessary current. If it can't, a larger power supply will have to be purchased from your dealer and installed.

I/O Address

Jumper(s) W5 configure the trace board's address in the PC's I/O space. The trace board address jumpers have been factory preset to 130 hex ($130) while the emulator is factory preset to $120. This setting is appropriate for a typical system and was chosen to work without modification in most computers. If your system is presently using any locations from $130 to $13F, an alternate I/O address location must be selected. Once you have decided on an address, change the W5 jumpers to select the new address, and also be sure to change the trace board address field using the menu choice **Config | Trace** to match your new hardware configuration.

Each pair of pins in the address header W5 represents one bit in the 10 bit address. Address bits 0-3 do not have jumpers and restrict the user to selecting blocks of 16 ($10) consecutive required addresses. The remaining upper 6 address bits are set with the jumpers A4 through A9. Shorting two pins represents a 0 in the address. A pair of pins with no jumper represents a 1. Settings from $100 to $3FF are possible, but many addresses are predefined in the PC either by IBM originally, or by convention (see table below).

The table below shows how a typical PC system uses its I/O address locations. Your PC configuration will have some, but rarely all of the devices listed below, allowing you to configure the emulator and trace boards in your system. If you are unsure of the locations available, there are several commercial software utility programs available to assist you in identifying the hardware and resources in your system. On systems with MS-DOS V6.0 or greater, Microsoft supplies a DOS based diagnostics utility "msd" that will aid you. Windows 95 users can use the "**My Computer | Properties**" facility to discover addresses of installed hardware components.

Copyright © 1997 Nohau Corporation

Common PC I/O address usage:

Most installations can use the factory set default address $130. Use this table to help in choosing an appropriate alternate I/O address if required for the trace card.

Hex Location	Typical Use
000 - 0FF	Used by system
1F0 - 1F8	Hard Disk
200 - 207	Game Adapter
210 - 213	Expansion Unit
220 - 22F	Sound Blaster (common)
238 - 23F	Bus mouse / alternate bus mouse
278 - 27F	Parallel Printer Port 2
2B0 - 2DF	EGA
2E0 - 2E7	GPIB adapter (AT)
2E8 - 2EF	Asynchronous Adapter (COM4)
2F8 - 2FF	Async. Adapter (COM2)
300 - 31F	Prototype Card (many Network cards use 300 / 308)
320 - 323	Hard Disk Controller (XT)
330 - 331	some sound cards use this in addition to $220
360 - 36F	IBM Reserved
378 - 37F	Parallel printer adapter
380 - 38F	SDLC or BiSync 2 Adapter, some sound cards
3A0 - 3AF	Binary Synchronous Communications Adapter 1
3B0 - 3BB	Monochrome Display
3BC - 3BE	Parallel printer port (on monochrome adapter)
3C0 - 3CF	Reserved EGA
3D0 - 3DF	Color/Graphics Adapter
3E8 - 3EF	Async. Adapter (COM3)
3F0 - 3F7	Floppy Disk Controller
3F8 - 3FF	Async. Adapter (COM1)

Copyright © 1997 Nohau Corporation

Card Installation

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

1. Turn power off.
2. Insert the trace board into an 8-bit slot (near the emulator card) with the short ribbon cable connected on to it. Be sure the edge fingers are securely inserted in the PC motherboard connector.
3. Connect the ribbon cable to the emulator board. Use care not to skew the connector or bend any pins.
4. Attach the bracket to the PC frame with a screw.

Other Miscellaneous Jumpers

There is one additional jumper on the trace board, located in the upper left corner of the card. It corresponds to the cards frame size (4K / 16K), and is preset at the factory to the correct position. The user should not modify the position of this jumper.

Trace terms:

Address The microcontroller's 16 bit address bus.

Cycle The type of bus cycle. Opcode Fetch, Read, Write, Don't care. Opcode Fetch (OpFetch) is defined as the first cycle of an instruction fetch while the LIR\ signal is active (low).

Data The microcontroller's 8 bit data bus.

Filter The A or B condition may be set to filter out unwanted frames based on the address, data, probe value, or cycle type.

Frames Frames hold the captured information and are 64 bits wide. The trace card records 4K (model TR4) or 16K (TR16) frames.

- 16 address bits
- 8 data bits
- 4 Miscellaneous bits (cycle type)
- 8 Probe 0 bits
- 8 Probe 1 bits
- 16 time stamp bits
- 4 internal use bits (not user accessible)

Probe 0 8 "hardware" bits available on the pod for connecting logic signals. By default these are connected to Port D on most pods with removable jumpers. The user may remove these jumpers and connect suitable logic level signals here for monitoring in the trace buffer display.

Copyright © 1997 Nohau Corporation

Probe 1 8 additional "hardware" bits available on the pod for connecting logic signals. By default these are connected to Port A or Port E on most pods with removable jumpers.

Trace Buffer The memory that holds the captured frames. The memory is configured as a FIFO buffer and by default captures all microcontroller activity unless a **Filter** or **Trigger** condition is set and matched.

Time Stamp The trace board captures as part of each frame a 16 bit time stamp which may then be viewed as a part of the buffer. The resolution of the timer is controlled by the trace setup **CPU Speed** and **Prescaler** fields.

Trigger When triggered (by an A or B condition), and following a user programmed delay, the trace card stops recording frames. This is useful to discover what the program did prior to or subsequent to a given condition. The trace trigger can also be used to break emulation.

Copyright © 1997 Nohau Corporation

Trace Setup

The trace board by default records all microcontroller activity. While this is occasionally useful, you will often wish to restrict what is recorded to just frames of interest for a particular debugging situation. The Trace Setup dialog box allows you to setup the trace board to capture just those frames meeting certain conditions. The result of this is that the trace buffer (holding the captured frames) contains more information that is pertinent to the problem you are trying to solve. The following paragraphs describe the options selectable from the Trace Setup dialog box and follow with some examples for setting up the trace board.

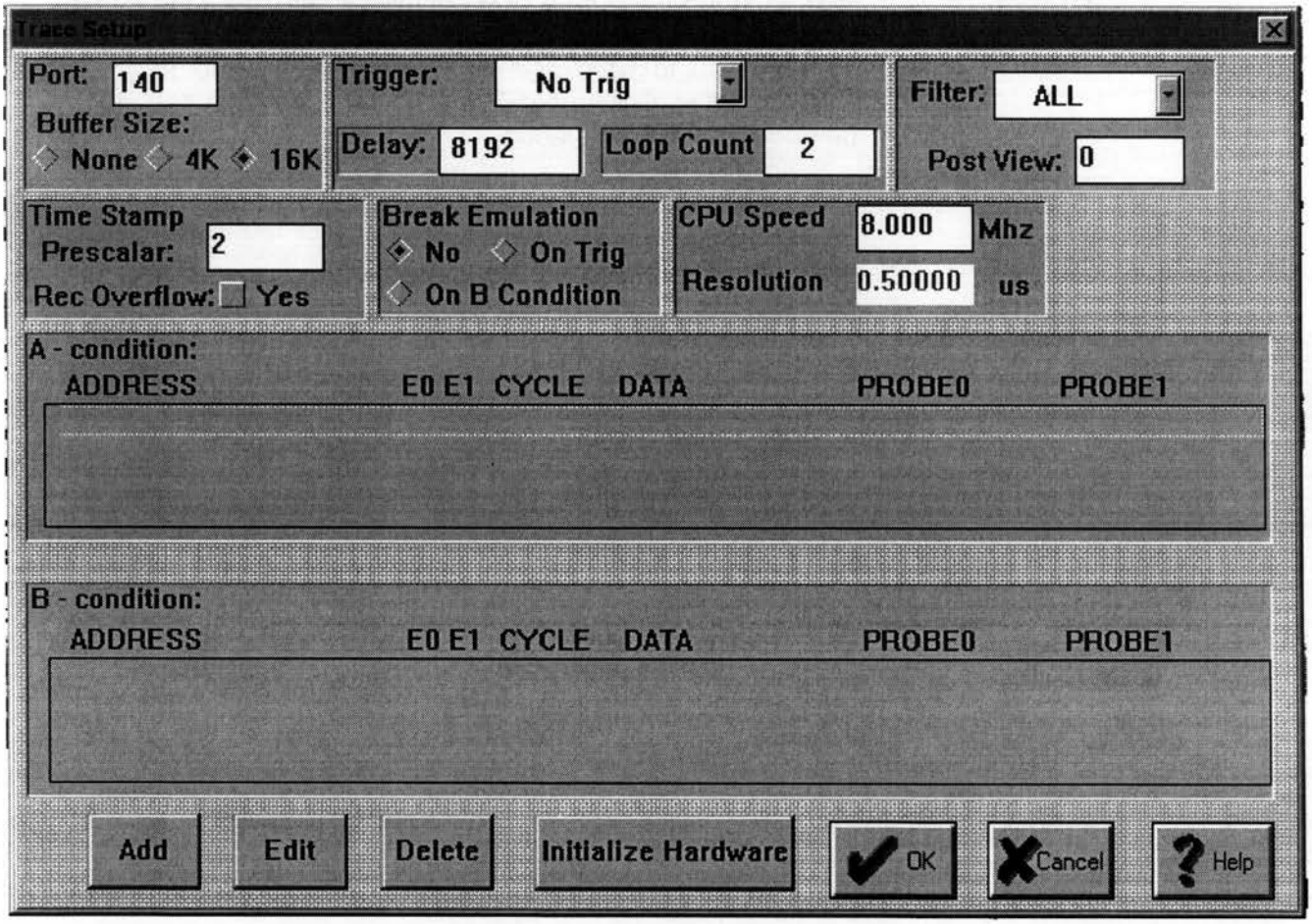


Figure 4-1: Trace setup dialog

Copyright © 1997 Nohau Corporation

Port address and model

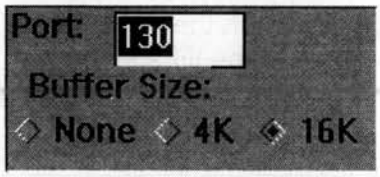


Figure 4-2: Trace address

The **Port** address setting (in hex) must agree with the address set on the trace board hardware via the jumpers. The default trace address is $130. Set the buffer size according to the frame depth you have, 4K for model TR4, 16K for model TR16, and None if you do not have a trace card installed.

CPU Speed

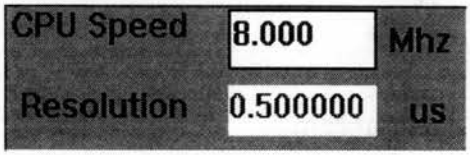


Figure 4-3: Trace CPU speed field

The CPU speed should be set to the actual crystal frequency used on your target (crystal jumpers / switches set to "Ext"), or the pod crystal (jumpers set to "Int"). This rate is divided by 4 to get the E clock rate. The E clock period determines the maximum resolution of the Time Stamps.

Time Stamp

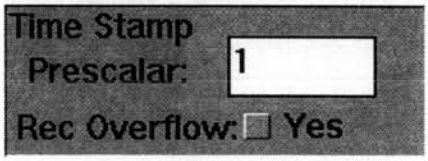


Figure 4-4: Time Stamp setup

The trace board hardware includes a 16 bit counter that is used as an execution timer. For each frame recorded in the trace buffer, the current value of this counter is saved as part of the frame and referred to as the "Time Stamp" for the frame.

Time stamps are optionally displayed in the Trace Buffer window as Absolute time or cycles, or Relative time or cycles. The maximum Time Stamp resolution is equivalent to the E clock period ($4 / f_{osc}$). The prescaler is used to reduce the resolution and thereby extend the maximum range of the Time Stamp counter to prevent overflow of the counter. Increase the prescaler value to extend the maximum recording time of Time Stamps without overflowing the counter, i.e. set the Prescaler to 32 to record with 16µS resolution (8.0MHz f_{osc}) and extend the overflow period from ~33mS to ~1.05S.

Note: If you plan to use the time stamp feature, be sure to set the prescale factor on the trace setup screen to something other than zero, otherwise all time stamps will show as zero. A value of one is the normal setting and provides maximum resolution.

The Overflow flag is used to detect and record Time Stamp counter overflow each time the counter wraps around from 65,535 to 0. When this flag is checked, an "overflow" frame is recorded in the trace buffer, and this extra frame is recorded regardless of the current filter settings. The frames recorded as a

Copyright © 1997 Nohau Corporation

result of an overflow are marked with an '**O**' in the memory access type field of the buffer display, i.e. instead of F, R, W (for Fetch, Read, Write) you will see FO, RO, WO. The overflow recording can be very useful when the trace filter setup results in infrequent frame recording but you still need the time stamp value in cycles.

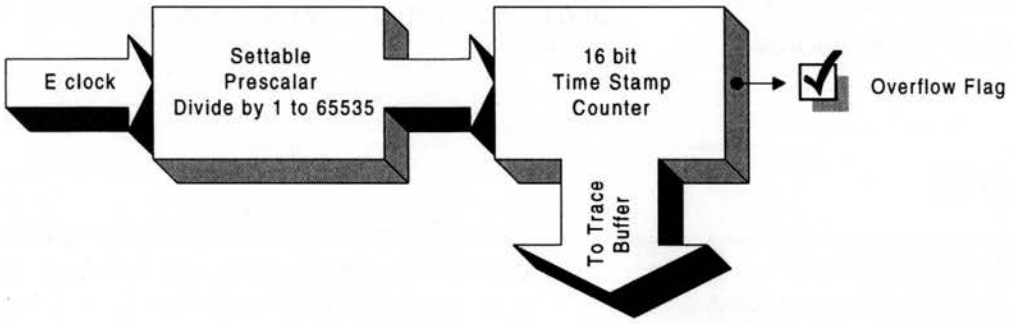


Figure 4-5: Time stamp block diagram

Break Emulation

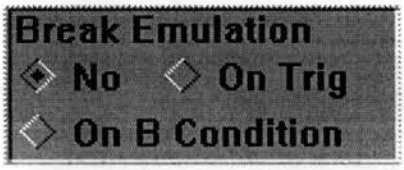


Figure 4-6: Trace - Break emulation

The Break Emulation box controls whether or not the emulator stops when a trace trigger occurs or 'B' condition is satisfied. The three choices are mutually exclusive.

Setting:	Action:
No	Emulation does not stop as a result of trace conditions. Emulation may still be stopped at an active Source, Program, or Special conditions window breakpoint or manually. This is very useful when it is desirable to let the processor run but snapshot conditions or code flow using the trace feature.
On Trig	Emulation stops when a trace trigger occurs as set by the **A** and **B** conditions and the **Trigger** logic setting. Note that due to internal logic, the break occurs one instruction after the actual instruction generating the trigger event.
On B Condition	Emulation stops when the **B** condition is true.

Copyright © 1997 Nohau Corporation

Trigger

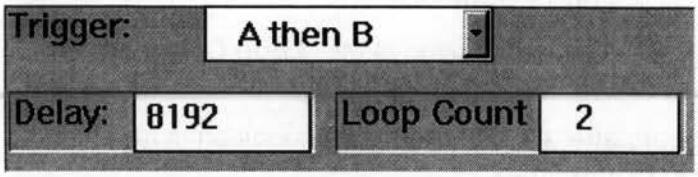


Figure 4-7: Trace trigger setup

The trace board stops capturing frames after the trigger condition is met and a programmable **Delay**. The **A** and **B** conditions and options are explained below. To disable triggering, select the **Trigger: No Trigger** option as explained below.

The **Delay** is the number of frames recorded after the trigger condition is met. The example shown (8192) is set to use half of the available buffer from a 16K board to record the events before the trigger, and half to record the events after the trigger. For a 4K (TR4) board, set this field to 2048 to achieve the same result. Set a higher number to record more frames after the trigger event, set a smaller number to record more pre-trigger event frames.

The **Loop Count** is the number of times the trigger condition must be met before the programmed delay begins. Note that if the **Loop Count** is set higher than the actual number of times the trigger event occurs, the board will not trigger.

Trigger Logic

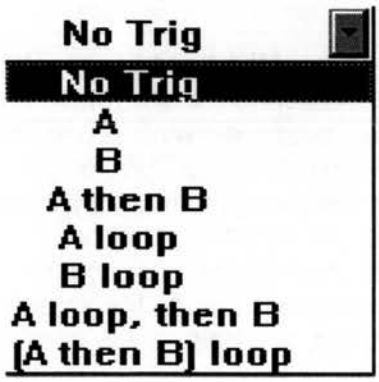


Figure 4-8: Trace trigger

This sub-menu selects the trigger logic. There are two programmable conditions **A** and **B** explained below, one or both of which must be met to cause a trace trigger. The selection **"No Trigger"** prevents trace triggering, and may be used as a quick enable / disable of triggering. This in combination with the complex conditions set in the **A** and **B** dialogs permits very complex, selective information capture. Note that triggering will occur based solely on the **A** or **B conditions** and the **Trigger Logic,** regardless of any filtering setup.

Condition	Action
No Trigger	Disables triggering.
A	Triggers when the **A** condition is met.
B	Triggers when the **B** condition is met.
A then B	Triggers after A condition, followed by B condition are each met in turn, in that order.
A loop	Triggers after **A** condition is met n (programmable) times as set in **Loop Count** field.

Copyright © 1997 Nohau Corporation

B loop	Triggers after **B** condition is met n (programmable) times as set in **Loop Count** field.
A loop, then B	Triggers after **A** condition is met **Loop Count** times followed by **B** condition (one time).
(A then B) loop	Triggers after **(A** followed by **B)** condition is met in turn **Loop Count** times.

Filter

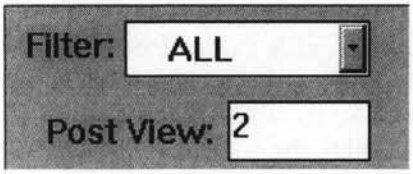


Figure 4-9: Trace filter setup

The **Filter** dialog box selects what frames are captured. Only frames meeting the conditions specified in the **Filter** field (see below) are recorded in the trace buffer.

The **Post View** field sets the number of extra bus cycles captured after each match of the trace filter. The frame meeting the filter parameter, and *n* more are recorded. This is useful for recording multi-byte instructions. It is also very useful for recording a few frames after a RTI (ReTurn from Interrupt) so that you may recognize the code returned to.

Filter Conditions:

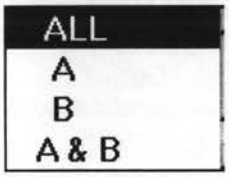


Figure 4-10: Filter conditions

Filtering is a key feature of the trace board. The **Filter Conditions** field controls what frames are recorded by the trace board, allowing you to limit the captured frames to only those in which you are interested. For instance you may only wish to see the characters read from the serial port. Setting a **condition** (**A** or **B**) and enabling the filter for that condition(s) will reduce the amount of information collected, and help you to pinpoint problems faster.

Filter:	**Action:**
All	All frames are captured. There is no filtering.
A	Only frames which match the **A** condition are recorded.
B	Only frames which match the **B** condition are recorded.
A & B	Frames which meet both **A** and the **B** conditions are captured.

Filtering is a key feature of the trace board. The Filter field controls what frames are recorded by the trace board. Record filtering means that the trace buffer will contain ONLY the records you are interested in. You may not need to visually scan through thousands of records to find the one you want.

Copyright © 1997 Nohau Corporation

Other Trace Setup Buttons

Use this button to add a new qualifier to the **A** or **B** condition boxes. Click in the **A** or **B** box, then click **Add** to open a dialog box. Alternately double click on the next available line in the **A** or **B** condition box. See Figure 4-11 for a description of the box to add or modify a trace qualifier.

Use this button to edit an existing qualifier. Highlight the qualifier, then click on **Edit**. Alternately, double click on the qualifier you want to edit.

Removes the selected qualifier. Highlight the qualifier, then click **Delete**.

Resets the trace hardware. Clears the trace buffer, resets the counter and prescaler to zero, and resets any pending loop counts to zero. You should use this if you are using a loop count for triggering and have stopped the trace or emulator prior to the trace triggering. This will clear the internal-loop-condition-found count to zero and prevent the trace from appearing to trigger early due to accumulated loop counts.

Conditions for Triggering and Filtering

Each machine cycle of the 68HC11's execution will present different address, data and port information to the trace board. We call the information from one such cycle a "frame." The largest features in the **Trace Setup** dialog box are the **A** and **B conditions**. These are lists of qualifiers (values) to look for in each frame as it is collected. The number of qualifiers that may be placed in each condition list is limited only by the memory available on your computer. Each qualifier in list **A** or **B** includes bits for the address and data buses, the external E0 and E1 bits, two groups of 8 bits for the probes, and cycle type. Each bit is either a "1", "0", or "don't care". For each machine cycle executed by the 'HC11, the trace board compares the recorded 64 bits for the current frame with the conditions in lists **A** and **B**. These comparisons produce boolean results - either a "true" or a "false".

Each qualifier in the condition list consists of seven separate fields: ADDRESS, E0, E1, CYCLE, DATA, PROBE0 and PROBE1. On all pods except POD-11S, PROBE0 is connected to PORTD and PROBE1 is connected to PORTA with jumpers, see the appropriate description for your pod in the EMUL68-PC User's Manual for further details. By removing the jumpers, the user can connect any

Copyright © 1997 Nohau Corporation

logic signals to the trace PROBE bits, and they will be sampled at the same time the other trace signals are captured.

Comparisons are made for each field of each qualifier within a given condition list (**A** or **B**) separately. Bits are either 0, 1, or x (don't care). The result of each comparison is by definition TRUE or FALSE. To be true, each field comparison has to match exactly with the current data from the active machine cycle. The result of the comparison within each field is OR'ed with the result of comparisons for that field in the other qualifiers in the given condition list (**A** or **B**). The result of this comparison is a TRUE or FALSE for each field. All fields must then also be TRUE for the condition to be met and result in a filter or trigger.

*Another way to think of this is to imagine each trace condition list (**A** and **B**) as a separate spreadsheet where each field in each qualifier is a cell. The cells are organized so that the seven fields are in columns, and the 8 rows represent qualifiers. First each cell is evaluated to TRUE or FALSE, blank cells are ignored. Then cells are OR'ed by columns. The result of this is a single row sum of TRUE or FALSE for each field. This row is then AND'ed across to give a single logic result of TRUE or FALSE for the entire **A** (or **B**) condition field. The subsequent trace actions then depend on the user's Trigger Logic (see page 4-9) and Filter Conditions (see page 4-10) selections.*

Adding a Trace Qualifier

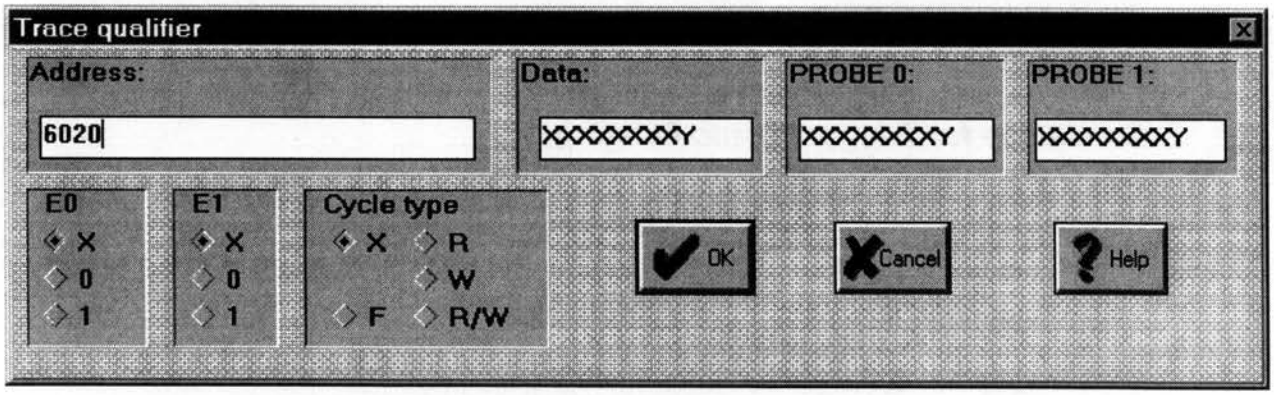


Figure 4-11: Trace qualifier dialog box

This dialog box is opened by clicking on the **Add** or **Edit** buttons in the **Trace setup** dialog box and allows you to configure one qualifier at a time. Each field is explained in detail below. The **'Y'** shown in some fields denotes binary notation is used for displaying the value. Values written into the fields without the ending **'Y'** are assumed to be hex. When this dialog box opens, the **Address** field is highlighted. Enter a suitable value, then use the <Tab> key or mouse to move to the **Data** field if you wish to also set a data value as a part of the qualifier. After entering a value, use the <Tab> key or mouse to move to each field in turn as needed. When all fields are set to your requirements, click on the **OK** button to return to the **Trace setup** dialog box. The number of qualifiers per **A** or **B condition** is limited only by memory available.

Copyright © 1997 Nohau Corporation

This dialog box can also be opened by double clicking inside the **A** or **B condition** sub-window of the **Trace setup** dialog box. If you double click on an existing qualifier, this box will open to modify that qualifier. If you instead double click on the next available line within the sub-window, the dialog box will open to enter a new qualifier.

The address sub-field can specify a single address or a range of addresses, and can be expressed in one of three acceptable forms as shown below. The terms "*expr1*" and "*expr2*" represent C expressions involving operators, symbols and/or hexadecimal constants. There are two other key words permitted; "to" is used to describe a from/to range between two numbers, and "len" used to describe a length. No spaces are permitted in the expressions between symbols, operators and constants, but spaces are required on each side of the words "to" or "len".

1. *expr1* a single address specified by *expr1*
2. *expr1* **to** *expr2* a range from *expr1* to *expr2* (inclusive)
3. *expr1* **len** *expr2* a range from *expr1* of length *expr2*

The rules for the use of symbols in address expressions is as follows.

1. The name of a C function represents the address of the first instruction that function generates.
2. If you enter the name of a C or assembly language variable, the ampersand symbol (&) must precede the name to obtain its address.
3. If a symbol name appears without a preceding ampersand, then the symbol's current value at the time of evaluation will be used.
4. Expressions are evaluated each time the trace setup dialog box is closed by clicking on the **OK** button.

Note *It is important to remember that if you use a symbol name in an expression, the value of the symbol used in the expression is evaluated each time the trace setup is changed. If you change the trace setup, it is possible for the value of a previously defined expression which evaluates to a range or location to also change.*

For example, suppose that *func1()* and *func2()* are functions in your program, and *foo* is a variable. The following examples will help to clarify the rules.

Example #	Expression in address field:	Definition of expression:
1	1000	a single address at $1000
2	func1 to &foo+1	a range of addresses from the first address of *func1()* to the address of (*foo* + 1 location)
3	func1 len 1000	a range of addresses from the first address of *func1()* to the address defined by (*func1()* + $FFF)

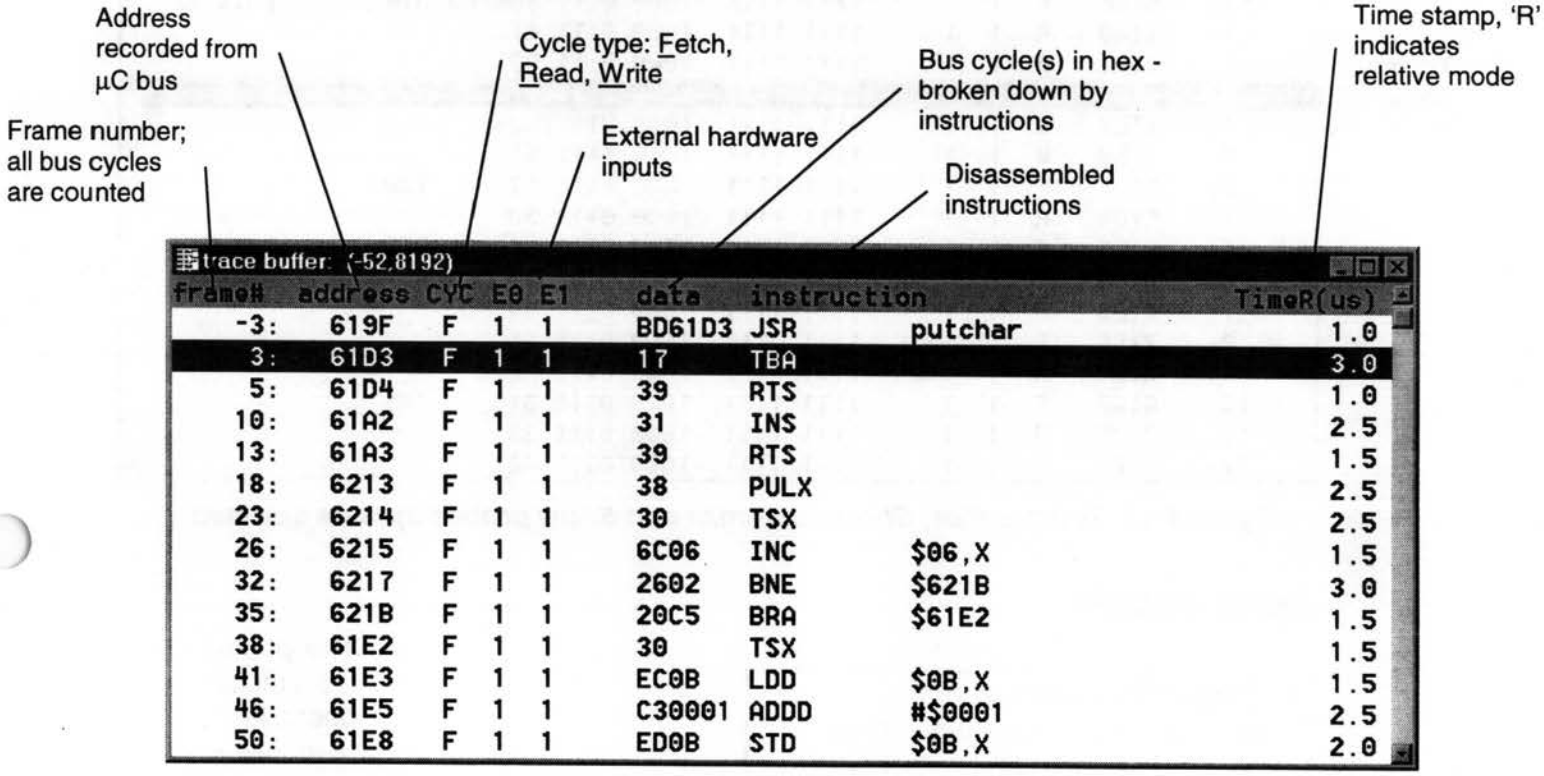
Copyright © 1997 Nohau Corporation

4	func1 len foo	a range of addresses from the first address of *func1()* to the address defined by (*func1()* + the value of *foo-1*)
5	&foo	a single address, the address of variable *foo*
6	func1 to func2-1	a range of addresses from the first address of *func1()* to the address one before *func2()*. Assuming that *func1()* and *func2()* were adjacent in memory (compiler dependent, see your map file) this would evaluate to contain all of the addresses of *func1()*

Copyright © 1997 Nohau Corporation

Trace Buffer Display

In the following two screens, the trace was set for no filtering, trigger on **A condition**. The **A condition** had a single qualifier with the address field set for the entry to the *putchar()* function at $61D3, the cycle type set to x (don't care) and all other fields set to x (don't care). The code is from the "*demo.a07*" example included with Win68. The time-stamping is set for relative mode with the prescaler set to 1, giving maximum resolution.



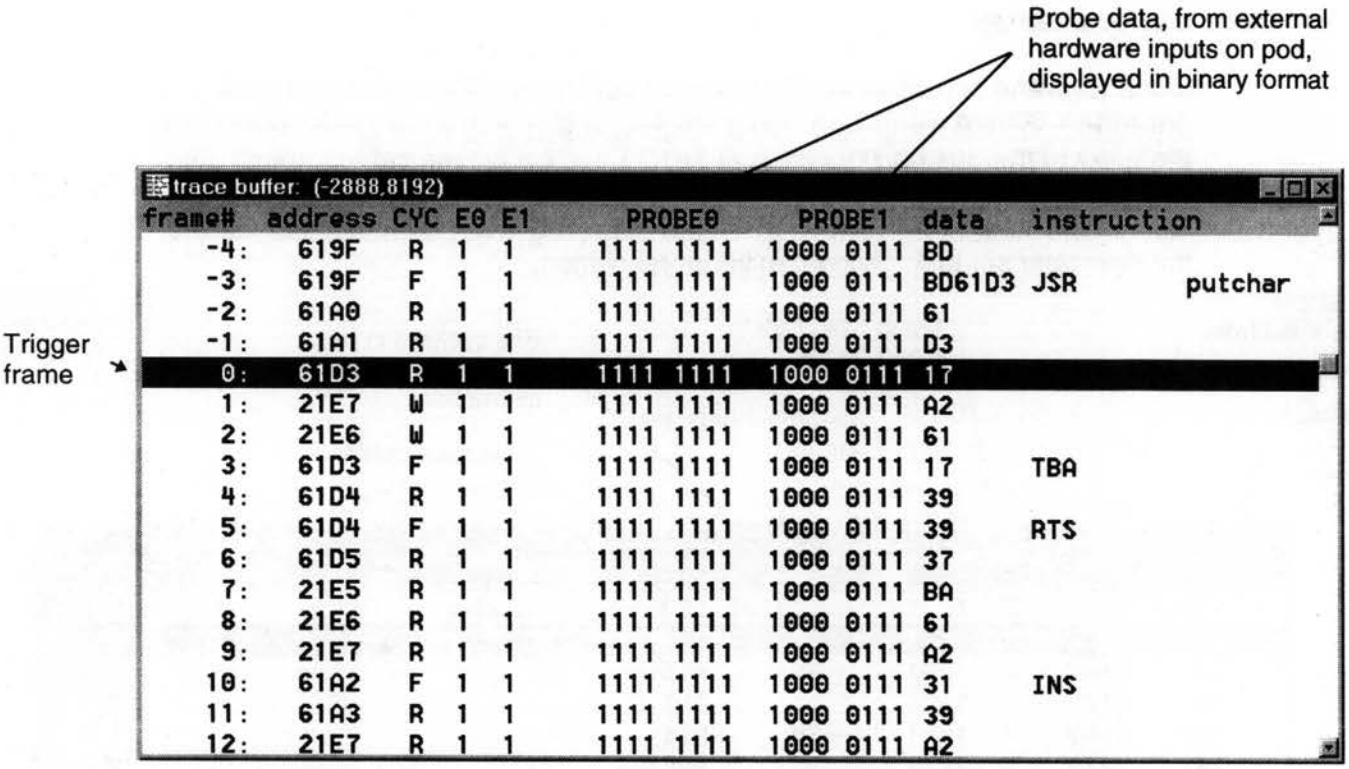
frame#	address	CYC	E0	E1	data	instruction		TimeR(us)
-3:	619F	F	1	1	BD61D3	JSR	putchar	1.0
3:	61D3	F	1	1	17	TBA		3.0
5:	61D4	F	1	1	39	RTS		1.0
10:	61A2	F	1	1	31	INS		2.5
13:	61A3	F	1	1	39	RTS		1.5
18:	6213	F	1	1	38	PULX		2.5
23:	6214	F	1	1	30	TSX		2.5
26:	6215	F	1	1	6C06	INC	$06,X	1.5
32:	6217	F	1	1	2602	BNE	$621B	3.0
35:	621B	F	1	1	20C5	BRA	$61E2	1.5
38:	61E2	F	1	1	30	TSX		1.5
41:	61E3	F	1	1	EC0B	LDD	$0B,X	1.5
46:	61E5	F	1	1	C30001	ADDD	#$0001	2.5
50:	61E8	F	1	1	ED0B	STD	$0B,X	2.0

Figure 4-12: Trace display window

This view shows the trace buffer after capturing the desired information. Note that frame numbers do not appear contiguous, this is because the **Show all frames** option, available by right clicking in the **Trace buffer** window, is *off*. All bus cycles were captured, but using this view reduces the clutter and allows focusing on instructions.

For more detailed timing and bus activity, turn **Show all frames** *on* as shown below. This is a portion of the same data as shown above in Figure 4-12 but broken down cycle-by-cycle. In this view, each bus cycle is shown on a separate line. Note that the instruction frames shown above as cycle type 'F' (opfetch) actually consist of a Fetch, often followed by Read or Write cycles. Also note the RTS instruction - the first 2 frames are internal 68HC11 cycles, during which the data bus is ignored; this is followed by popping the return address ($61A2) from the stack into the PC. Also shown in this view is the PROBE0/1 data from the hardware inputs available on many pods.

Copyright © 1997 Nohau Corporation



frame#	address	CYC	E0	E1	PROBE0	PROBE1	data	instruction
-4:	619F	R	1	1	1111 1111	1000 0111	BD	
-3:	619F	F	1	1	1111 1111	1000 0111	BD61D3	JSR putchar
-2:	61A0	R	1	1	1111 1111	1000 0111	61	
-1:	61A1	R	1	1	1111 1111	1000 0111	D3	
0:	61D3	R	1	1	1111 1111	1000 0111	17	
1:	21E7	W	1	1	1111 1111	1000 0111	A2	
2:	21E6	W	1	1	1111 1111	1000 0111	61	
3:	61D3	F	1	1	1111 1111	1000 0111	17	TBA
4:	61D4	R	1	1	1111 1111	1000 0111	39	
5:	61D4	F	1	1	1111 1111	1000 0111	39	RTS
6:	61D5	R	1	1	1111 1111	1000 0111	37	
7:	21E5	R	1	1	1111 1111	1000 0111	BA	
8:	21E6	R	1	1	1111 1111	1000 0111	61	
9:	21E7	R	1	1	1111 1111	1000 0111	A2	
10:	61A2	F	1	1	1111 1111	1000 0111	31	INS
11:	61A3	R	1	1	1111 1111	1000 0111	39	
12:	21E7	R	1	1	1111 1111	1000 0111	A2	

Figure 4-13: Trace buffer, *Show all frames* and *Show probes* options selected

Trace Local Menu

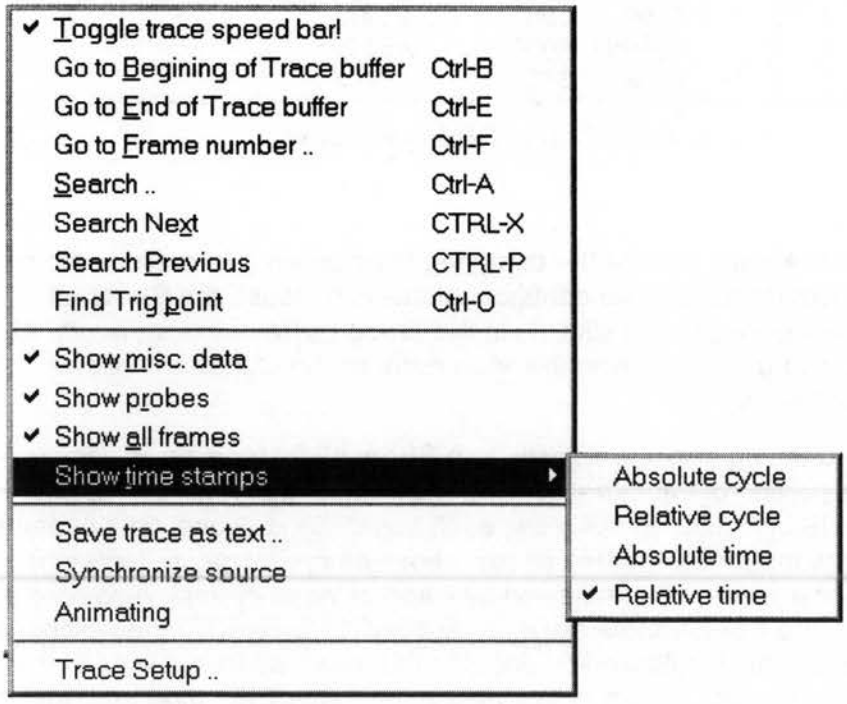


Figure 4-14: Trace buffer local menu

This pop-up local menu opens in response to a right mouse click in the trace buffer window, or from the main menu "**Trace**" when the trace buffer window is selected. It is used to navigate and control the trace buffer display window. Also shown is the **Show time stamps** sub-menu used to

Copyright © 1997 Nohau Corporation

control the display format for time stamping frames. The check marks indicate active display format controls. See the table below for details.

Trace Menu Commands	HotKey	Description
Toggle trace speed bar!		Turns the trace speed bar at the top of the window, below the button bar, on or off. The speed bar allows quick access to common trace functions and shows current trace status.
Go to Beginning of trace buffer	<Ctrl> B	Positions the cursor (highlighted line) at the first frame captured in the trace buffer. This is the oldest frame in the execution history.
Go to End of trace buffer	<Ctrl> E	Positions the cursor (highlighted line) at the last frame captured in the trace buffer. This is the newest frame in the execution history.
Go to Frame number..	<Ctrl> F	Positions the cursor (highlighted line) at the requested frame. Frames that are captured before the trigger event are negative, post-trigger frames are positive. The trigger frame is frame number 0. *If triggering is disabled or the trigger did not occur, all frames are pre-trigger and have negative frame numbers, except for the last frame recorded which is frame zero.*
Search..	<Ctrl> A	Opens a dialog box for searching in the trace buffer. The search is qualified by address, data, cycle type, etc. and can find the n^{th} occurrence by entering a loop count.
Search next	<Ctrl> X	Finds the next occurrence of the data meeting the search criteria by searching forwards in the buffer.
Search previous	<Ctrl> P	Finds the previous occurrence of the data meeting the search criteria by searching backwards (towards the beginning) in the buffer.
Find trigger point	<Ctrl> O	Positions the cursor on the frame containing the trigger event. If the trigger event did not occur, or triggering is disabled, the last frame in the buffer is displayed.
Show misc. Data		When checked, displays in the trace buffer window the cycle type (F, R, W) and E0/E1 pin data.
Show probes		When checked, displays the data for the hardware probe bits in binary. PROBE0/1 are connected to I/O ports A and D with jumpers on many pods, the jumpers may be removed and the pins connected to any valid logic signal for

Copyright © 1997 Nohau Corporation

	viewing in the trace buffer. See the detailed section for your pod for further information.
Show all frames	Shows each bus cycle on a separate line (frame) when checked. Instruction cycle type details are also broken out.
Show time stamps	Adds time stamp information on the far right of the trace buffer window. The information can be displayed in four different formats, as described below.
Save trace as text..	Opens a dialog box for saving the contents of the trace buffer to an ASCII text file. The user can choose a file name and starting and ending frame numbers. The information is saved in the same format as the current trace buffer display.
Synchronize source	Clicking this positions the Source and Program windows cursors to the highlighted frame in the trace buffer window.
Animating	Scrolls the trace buffer, highlighting one line at a time for review. The **Program window** cursor is updated to synchronize with the trace display.
Trace setup	Opens the **Trace setup** dialog box. The dialog box can also be opened from the **Trace speed bar**, or the main menu by clicking on **Config** I **Trace** .

Timestamping Sub-menu

This sub-menu, available within the trace-local menu by selecting **Show Time stamps**, controls how time stamps are displayed in the trace buffer window. There are four choices, the current selection is indicated by a check mark. Note that the values in the display are affected by the **Time stamp Prescaler** value (see page 4-7), both cycles and time are divided by the programmed value before being recorded in the trace buffer and displayed.

Menu Commands	HotKey	Description
Absolute cycle		Shows the number of bus cycles from the start of tracing.
Relative cycle		Shows the bus cycles between each displayed frame.
Absolute time		Shows the time from the start of tracing.
Relative time		Shows the elapsed time between frames.

Copyright © 1997 Nohau Corporation

Trace Speed Bar

Figure 4-15: Trace Speed Bar for rapid access to trace setup and display

The Trace Speed Bar allows quick access to the most common trace functions. This bar is enabled by using the local menu item "**Toggle trace speed bar**" from the Trace Buffer window's local menu.

At the far left of the bar the current trace status is shown (in the example shown, tracing is ***Stopped***). This indicates the status of the trace board (i.e. tracing, stopped) and not the emulator. With tracing started, you must also Run the emulator to collect information. In the rightmost box is shown the **frames** counter, this is the number of frames collected that meet the trace filter conditions. Next to this are the current values for setup parameters **trig delay** and **loop count**.

The **Trace** button starts and stops the trace feature. Tracing may be started and stopped without breaking emulation.

The **Setup** button brings up the **Trace Setup** dialog box which is covered in detail earlier in this chapter.

Program Performance Analyzer (PPA)

What part of your application uses most of the CPU cycles? Where should you concentrate optimization efforts? These are the type of questions that **Performance Analysis** is designed to answer by showing you where your program spends it's time. You set up "bins" which describe address ranges, run your program, and then watch the results to see how much relative time your program spends in each bin. Bins are often defined as single functions or subroutines and allow you to quickly see where your program is comparatively spending it's time. There is also a default "*Miss*" bin, which records cycles spent outside of your other defined bins. You must have the optional trace board installed to utilize the PPA features.

Performance Analysis is a statistical analysis of execution behavior. Once each second bus cycles are collected, sorted into their respective bins, and the results are displayed on the screen as percentages. The percentage of cycles that are collected depends upon the speed of your P.C. and what other tasks are running, etc. Nohau recommends running Win68 by itself to obtain the highest accuracy possible.

To produce more accurate results, run your program for longer periods of time (or at slower clock rates). If you watch the percentage values on the screen, you will see them change quickly at first, then more slowly. When the percentages of cycles in each bin change very slowly, you can assume that the percentages shown are as accurate as possible.

Copyright © 1997 Nohau Corporation

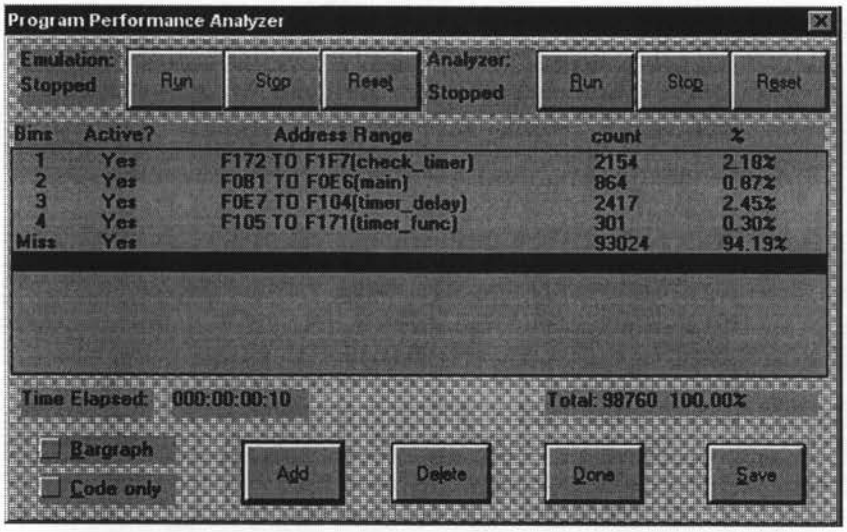


Figure 4-16: Program Performance Analysis window

When you select **Config | PPA..**, you will see a window similar to that shown above which is displaying results captured from the "*demo*"[4] example included with Win68. This window is used to both set up and display the results of the Program Performance Analyzer. The application and the data collection will automatically be started every time you open the PPA window. The six buttons at the top of the window control the running of the program and the statistics collection process separately.

Each bin is really a user defined address range. If the address range corresponds exactly to the address range of a function, that function name will be displayed next to the address. A bus cycle from an address that doesn't fall within any (user defined) address range will be counted in the *Miss* bin. A cycle within an existing but inactive (address range) bin will not be counted at all, that is it will not count in the inactive range and it also will not be counted within the Miss bin. Statistics will not be kept for that inactive bin at all.

PPA Fields

Bins This is the assigned "bin" number. A bin is a defined address range. You may define bins using the Add button - the number of bins is limited by your available memory. If you define more bins than can be displayed at one time, a scroll bar will appear for navigation of the list. There is also a default "Miss" bin, which records all bus cycles which fall outside the range of any user defined bins. The Miss bin cannot be deleted, edited, or made inactive.

[4] Load the file "*demo.a07*" (located in the *examples* directory) from the **File | Open** menu.

Copyright © 1997 Nohau Corporation

Active? This field shows whether or not a defined bin is counted in the statistics. When a bin is active (“**yes**”) it is counted as a part of the current statistics. When inactive (“**no**”), the bin is not counted. This allows the user to quickly see or ignore the effects of a bin without removing or adding it to the display.

Address Range This is the user defined address range for the bin and is displayed as hex in *FROM -> TO* format. If a complete function is exactly defined by the range, the function name is shown in parentheses'. This field is also used to display the histogram if the **Bargraph** box is checked.

Count Shows the bus cycle count for the bin during the current PPA run. A bus cycle whose address falls within the address range defined for the bin is counted. The count is cumulative across multiple **Run / Stop** commands, and may be reset to 0 by clicking the **Reset** (Analyzer) button.

% Shows the percentage of the counted cycles that the program spent accessing addresses within a defined bin. The sum of the percentages of all bins (user defined and the **Miss** bin) is always equal to 100%.

Time Elapsed: Displays the elapsed time since the start of the last PPA run, up to 1000 days total. The time is reset when the Analyzer **Run** or **Reset** buttons are clicked on. The time is not reset by starting and stopping the emulator.

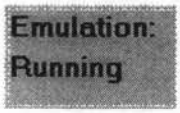
Total: Shows the total cycles counted in the user defined and **Miss** bins along with the percentage total, which is always equal to 100%.

PPA Controls

The PPA controls are divided into four sections: emulator control buttons and status, analyzer (PPA) control, controls for bins, and two miscellaneous check boxes as shown in Figure 4-16 above. The emulator control buttons are located at the top left of the PPA window, next to the **Emulation:** status. The analyzer controls are located at the top right of the PPA window, to the right of the **Analyzer:** status. The bin control buttons are located at the bottom of the window, with the miscellaneous check boxes to their left. The emulator and the PPA are controlled independently, that is the emulator can run with PPA stopped, and vice-versa, although running the PPA while the emulation is stopped has little practical utility. To record bus cycles for PPA analysis, you must have both the emulator and the PPA running.

Copyright © 1997 Nohau Corporation

Emulation Status and Controls



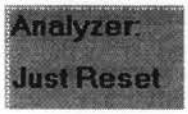
This box shows you the current emulator status: **Running** (your program, shown at left); **Stopped** (not executing your code); or **Just Reset** which indicates the **Reset** button was the last one clicked, resetting the microcontroller and waiting for you to begin emulation by clicking on **Run**.

Starts the emulator running your program. The emulator must be running to collect cycles for analysis. Does not effect the elapsed time or total cycle counters.

Halts program execution by stopping emulation. Does not effect the elapsed time or total cycle counters. If the Analyzer is running, the Time Elapsed counter will continue to run, but no bus cycle information is collected, and the count and percentage fields will not change.

Resets the microcontroller and waits for you to begin emulation by clicking on the **Run** button. Same effect as **Run | Reset Chip**. Emulation status will change to **Just Reset**.

Analyzer Status and Controls



This box shows you the current Program Performance Analyzer status: **Running** - the PPA is collecting bus cycles for analysis and updating the **Time Elapsed** and **Total:** fields; **Stopped** - the PPA is not collecting bus cycle information and bins and other fields are not being updated; or **Just Reset** (shown) which indicates the analyzer **Reset** button was the last one clicked which resets all bin counts, percentages, and other PPA fields to zero.

Starts the PPA running and collecting bus cycles. The collected information is analyzed and used to update the bin counts, percentages, and other PPA fields. Starting the PPA will reset the **Time Elapsed:** counter.

Copyright © 1997 Nohau Corporation

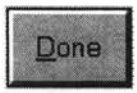
This button stops the PPA from collecting bus cycle information. The next time the (analyzer) **Run** button is clicked, the **Time Elapsed:** counter will be reset to zero. You must stop the PPA to modify the user defined bins. Other than stopping updates, this button does not affect any fields, and has no effect on emulation.

This button resets the PPA logic and zeros the bin counts, bin percentages, **Time Elapsed:** and **Total:** counters. The PPA status will change to **Just Reset**. This button has no effect on emulation, the microcontroller, or your program. You should generally use this button after you modify (add, delete, change addresses, change active status) the bin parameters so that the statistics reflect the current settings.

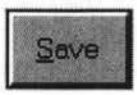
Bin Controls

Opens the **Address Range** dialog box shown in Figure 4-17. You may also open this dialog box by double clicking in the PPA window on the next available blank line. This dialog box is used to define the address ranges used for bins and place them in the PPA window. The PPA must be stopped to **Add** bins. You may also change a bin from active to inactive using this dialog box.

Deletes the bin highlighted in the PPA window from the window. To restore the bin (address range) to the window, use the **Add** button. To remove a bin from the current statistics without deleting it, change it's **Active** status from "**yes**" to "**no**" using the **Add** dialog box as explained above.



Quits the PPA window and ends the PPA session. When the PPA window is reopened, all counters, statistics, and fields are reset to zero.



Saves the current PPA bin definitions as part of the emulator startup file. This does not save the PPA statistics. Only one set of PPA definitions is saved, not one per project.

Miscellaneous Controls

When checked, this control displays a horizontal bar histogram of the relative bin percentages. It gives you a quick graphical feel for where your program is spending its

Copyright © 1997 Nohau Corporation

cycles. The bars are displayed in the **Address Range** field of the PPA window.

☐ Code only — This field is not currently operational.

Adding User Defined Bins:

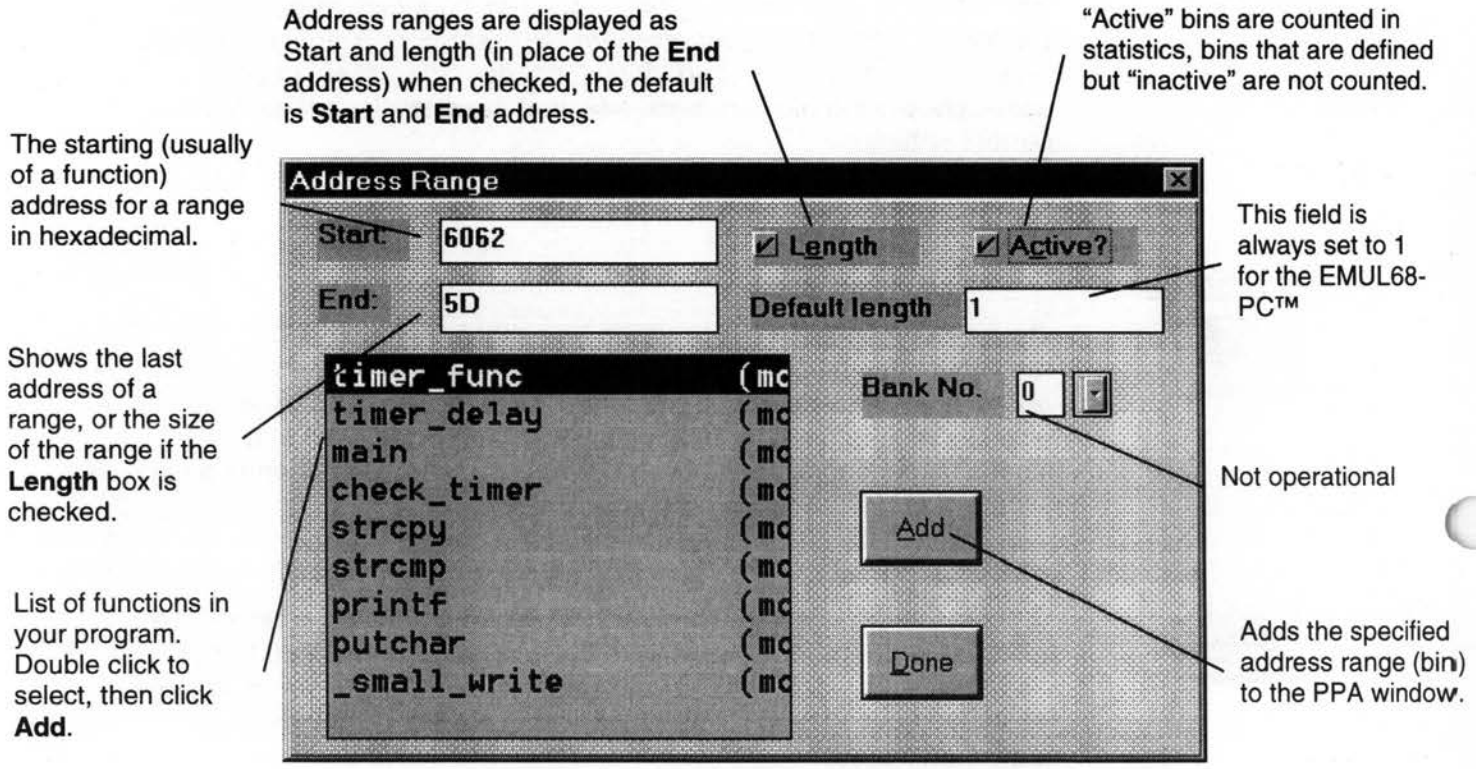


Figure 4-17: Adding bins to PPA display

This dialog box pops up when you select the **Add** button in the PPA window. It allows you to assign user selected address ranges to PPA bins, and specify whether the range is active or inactive. The specified address range is used for the next available PPA bin in the PPA display window. The example shown is from the "demo.a07" program. A list of the functions in your program is automatically displayed when this dialog box opens. Note that you must stop the PPA to **Add** or modify a bin in the PPA display.

The **Length** field controls how the **End** field is used. When checked, the **End** field displays the length. Without a check in the **Length** field, the **End** field displays the end address in hexadecimal notation.

The **Active** field determines whether or not a defined bin is counted in the statistics. By making a bin inactive, you can ignore it for a PPA run without

Copyright © 1997 Nohau Corporation

deleting (and then adding again later) it, permitting quick, temporary PPA configuration changes.

Example - Adding a PPA Bin

As an example, to add a bin corresponding to the main() function, double click on main in the function list, then click on the **Add** button. Note that clicking once on main will highlight it but not update the **Start** and **End** fields with the values for main(). Also note that double clicking does not actually add the bin. You must click on the **Add** button to add the bin to the PPA window display. With the **Address Range** dialog box open, you may add as many bins as you like before clicking on the **Done** button to close it.

Copyright © 1997 Nohau Corporation

Chapter 5: POD Boards

POD boards are used to adapt the EMUL-68™ in-circuit emulator card to specific 68HC11 target processors. Information on POD boards is not included in this manual at this time. Please see Chapter Six of the EMUL-68™ (DOS) User's Manual for information on pods for the 68HC11 processor.

Copyright © 1997 Nohau Corporation

Chapter 6: Accessories

This section is incomplete. Please refer to the EMUL68™-PC (DOS) manual for further details.

Copyright © 1997 Nohau Corporation

INDEX

Copyright © 1997 Nohau Corporation

D

E

F

G

Copyright © 1997 Nohau Corporation

H

I

J

L

M

Copyright © 1997 Nohau Corporation

N

O

P

Q

R

Copyright © 1997 Nohau Corporation

S

T

Copyright © 1997 Nohau Corporation

U

V

W

Z

Copyright © 1997 Nohau Corporation