



Copyright © 1999 by MetaLink Corporation

All rights are reserved.

This manual may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without the prior agreement and written permission of MetaLink Corporation.

iceMASTER<sup>®</sup>, MetaLink<sup>®</sup>, and the combination iceMASTER<sup>®</sup> and an alphabetic or numeric suffix, and the MetaLink logo are claimed trademarks of MetaLink Corporation and may only be used to describe MetaLink products.

National Semiconductor<sup>™</sup> is a trademark of National Semiconductor Corporation.

IBM<sup>®</sup> is a registered trademark of IBM Corporation.

Microsoft<sup>®</sup>, Windows<sup>®</sup>, and MS-DOS<sup>®</sup> are registered trademarks of Microsoft Corporation.

MetaLink Corporation reserves the right to make improvements in the products described in this manual as well as the manual itself at any time and without notice.

---

# Table Of Contents

- Chapter 1: Introduction** . . . . . 5
  - The Emulator . . . . . 5
  - Recommended References . . . . . 5
  - What You Need To Know . . . . . 6
- Chapter 2: Hardware Installation** . . . . . 7
- Chapter 3: Hardware Description** . . . . . 8
  - COP880C, COP888FH and COP888GG EPU Board . . . . . 8
  - COP8SAC and COP8SGR EPU Board . . . . . 9
  - Switches . . . . . 9
  - LED Indicators . . . . . 9
  - Power . . . . . 10
  - RS-232 Interface . . . . . 10
  - Target Interface Cable . . . . . 11
  - Programming Socket . . . . . 11
  - 16-Pin DIP Shunt . . . . . 11
- Chapter 4: Software Guide** . . . . . 12
  - Overview/Features . . . . . 12
  - Default Screen Layout . . . . . 14
  - Available Windows . . . . . 14
- Chapter 5: Operational Considerations** . . . . . 16
  - In-Circuit Simulation Limitations . . . . . 16
  - Memory Limitations . . . . . 17
  - Static . . . . . 17

<b>Chapter 6: Contacting National Semiconductor</b> . . . . .	<b>19</b>
Phone . . . . .	19
National's Microcontroller Bulletin Board . . . . .	19
Internet . . . . .	19
<b>Chapter 7: Differences</b> <b>(Model 400 vs. Debug Module vs. EPU)</b> . . . . .	<b>20</b>

---

# Chapter 1: Introduction

## The Emulator

---

The iceMASTER EPU-COP8 is an in-circuit simulator which allows you to debug code and hardware designs for the COP880C, COP888FH, COP888GG, COP8SAC and COP8SGR microcontrollers and to program some COP8 EPROM/OTP devices. Which devices can be programmed depend on which EPU you have (see the Programming Socket description in the Hardware Description Chapter).

**Warning: Only the EPROM/OTP devices listed in the Host Software may be programmed using the EPU-COP8. Attempts to program other EPROM/OTP devices may result in damage to the EPROM/OTP and the EPU-COP8 and will void the warranty.**

An in-circuit simulator combines features of a simulator and an in-circuit emulator. Like an emulator it plugs into your target system in place of the microcontroller and it executes code using the real microcontroller, which is on the EPU board. However, like a simulator it does not run real-time since the microcontroller is controlled cycle-by-cycle through software. In addition, break handling and trace are done completely through software. Throughout this document we will use the term simulation to describe what the in-circuit simulator is doing.

The EPU is controlled by an IBM PC (or compatible) running the Windows operating system.

The EPU can be operated in a target system in place of the COP880C, COP888FH, COP888GG, COP8SAC or COP8SGR microcontroller (depending on which EPU you have), or independently in stand alone mode. Stand alone mode allows you to execute code without a target system (provided no interaction with external devices is needed).

Designers may use the EPU to develop and debug their hardware and software designs. All available features of the microcontrollers are accessible interactively, as well as through your application programs.

For information about the COP880 or COP888 instruction set please refer to the COP880 or COP888 data sheets or the National assembler manual.

## Recommended References

---

Several additional references can be of help to you as you progress through the development process. The data book and programmer's guide for the microcontroller you are using provide essential information. You will also need the programmer's manual for the development language you are using.

## **What You Need To Know**

---

Throughout this manual it is presumed that you have a working knowledge of:

- 1) the family of microcontrollers you are emulating
- 2) the IBM PC (or compatible) as an engineering tool
- 3) a development language (e.g., Assembly Language or C)
- 4) Microsoft Windows 3.11, Windows 95 or Windows NT 4.0.

A few of these topics are discussed in this manual as a means of illustrating a particular feature or facet of the emulator's capabilities; however, basic programming knowledge and familiarity with the microcontroller architecture are assumed.

---

## Chapter 2: Hardware Installation

Connect one end of the RS-232 serial cable to a serial communication port on the Host Computer. Be sure you are using Communication Port 1 (COM1), 2 (COM2), 3 (COM3) or 4 (COM4). Connect the other end of the RS-232 serial cable to the EPU.

Connect the power supply to the EPU by inserting the power supply's connector into the EPU power receptacle. For safety, we recommend that all items in your system, including EPU, Host Computer and target board, be connected to the same outlet. Different outlets, though near one another, may be connected to different circuits resulting in large potential differences between grounds.

For a description of all hardware components of the EPU see the Hardware Description Chapter.

**Warning: The EPU is not designed to be “hot-plugged”. The EPU and target system must be turned off when attaching or removing the EPU from the target system. “Hot-plugging” may cause CMOS latch-up and can void the warranty.**

# Chapter 3: Hardware Description

## COP880C, COP888FH and COP888GG EPU Board

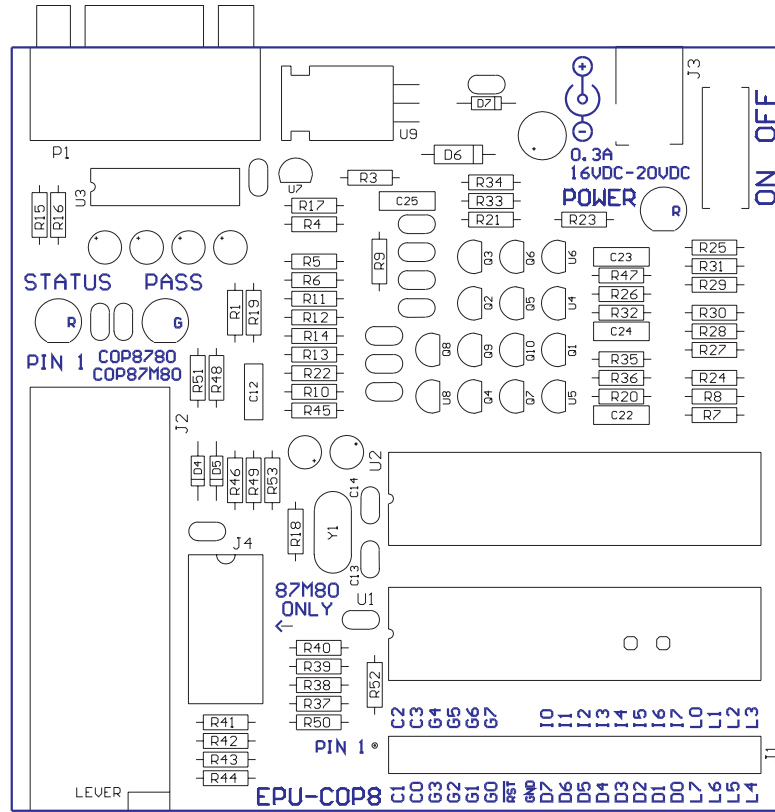


Figure 1. COP880C, COP888FH and COP888GG EPU Board

## COP8SAC and COP8SGR EPU Board

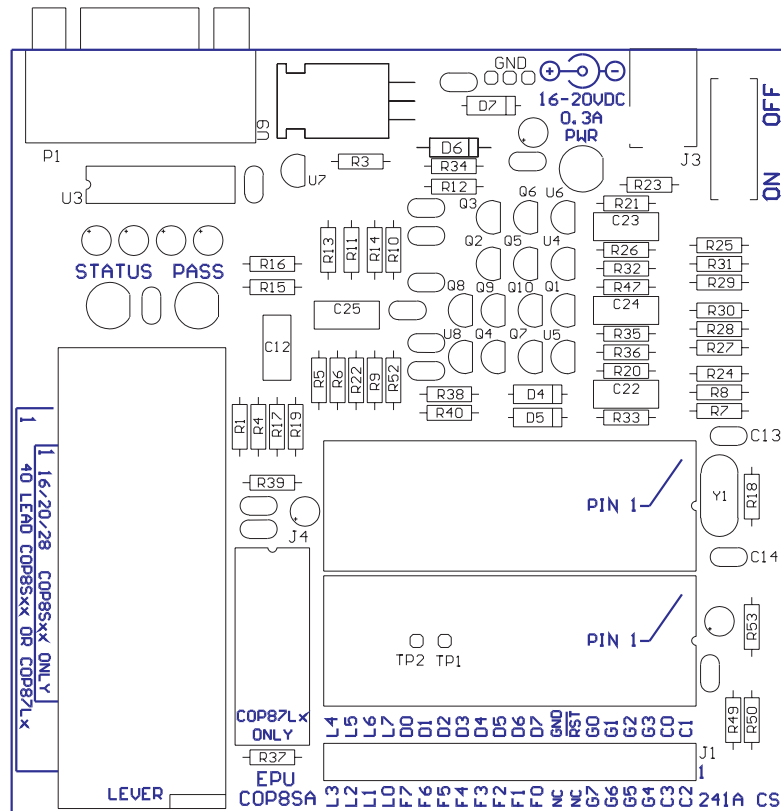


Figure 2. COP8SAC and COP8SGR EPU Board

### Switches

One switch is provided on the EPU:

- 1) The slide switch controls **Power** coming in on the power connector.

### LED Indicators

<b>POWER LED</b>	The POWER LED (red) indicates that power is being supplied to the EPU board through the power connector.
<b>PASS LED</b>	The PASS LED (green) is used to indicate the final status of a programming operation. When lit, the previous programming operation was successful. When dark, the STATUS LED is used to determine the status of the programming operation.
<b>STATUS LED</b>	The STATUS LED (red) is used to indicate the status of a programming operation. A flashing STATUS LED (red) indicates that a programming operation is in progress. When lit (steady), the previous programming operation failed.

## Power

The power supply supplies both the logic and programming voltage of the EPU. It is supplied using a standard 2.5mm X 5.5mm X 9.5mm plug and jack, center positive (Switchcraft Type 760). The power supply must provide +16 to +20 volts DC at 300 mA. The ripple voltage must be no greater than 50 millivolts, peak-to-peak.

Note that there is no power connection to the target.

## RS-232 Interface

Host PC Cable Connector		Cable			EPU Cable Connector (Male DB-9)	
Signal	Pin		Function	Direction	Pin	Signal
	PC/XT (Female DB-25)	PC/AT (Female DB-9)				
TxD	2	3	Data to EPU	→	2	RxD
RxD	3	2	Data to Host	←	3	TxD
RTS	4	7	Reset EPU - active high	→	4	RTS
Ground	7	5	DC Ground	↔	7	Ground

**Table 1. RS-232 Interface**

The communication link to the Host Computer is based on the serial RS-232C specification. The serial baud rates are established entirely under Host Software control. Therefore, you do not need to adjust your serial port's baud rate using, for example, the DOS MODE command.

The cable mates via a 9-pin male DB-9 connector on the cable at the EPU end. At the host end, the mating connector on the cable is a 9-pin female DB-9 connector. A 9-pin cable is provided with the EPU. Adapters are available to connect the DB-9 connector (at the PC end) to a DB-25 connector (DB-25 female to DB-9 male). Note that Pins 2 and 3 are reversed from their normal 25-pin D connector assignments in the 9-pin RS-232C interface of the PC/AT.

## **Target Interface Cable**

---

The target interface cable is a 40-pin wiremount connector (40 position 3M 3417 series or equivalent) through a 40-pin flat cable to a 40-pin DIP plug (40 position 3M 3508 series or equivalent).

**WARNING: The target interface cable must be removed from the EPU-COP8 board in order to program any EPROM/OTP devices.**

## **Programming Socket**

---

A 40-pin DIP ZIF socket is provided for programming COP8 EPROM/OTP devices. Which devices can be programmed depend on which EPU you have, as follows:

- COP880C**    The COP880C, COP888FH and COP888GG can program 40-pin DIP COP8780  
**COP888FH**    and 40-pin DIP COP87Lx devices using the 40-pin DIP ZIF socket. Other COP878x  
**COP888GG**    and COP87Lx devices can be programmed using MetaLink Programming Adapters.
- COP8SAC**    The COP8SAC and COP8SGR can program 16-pin, 20-pin, 28-pin and 40-pin DIP  
**COP8SGR**    COP8SAx and COP8SGx devices and 40-pin DIP COP87Lx devices using the 40-  
pin DIP ZIF socket. Other COP8SAx, COP8SGx and COP87Lx devices can be pro-  
grammed using MetaLink Programming Adapters.

For a list of all supported devices along with information required for programming each device (such as whether an adapter is required or if shunt blocks need to be installed), press the *Write* button in the *PROM Programmer* Dialog Box. The device list will be written to a text file, in a format suitable for printing.

Note that when a new device is selected, the Host Software will tell you if a Programming Adapter is required and whether to remove or install shunt blocks.

## **16-Pin DIP Shunt**

---

**WARNING: The 16-pin DIP shunt provided with the EPU is for programming support of COP87Lx devices and should not be installed except when programming a COP87Lx device. Care should be taken to store the shunt since it may be needed for future use. When programming any COP87Lx device make sure the shunt is installed in the jumper block labeled J4, which is also labeled:**

**87M80 ONLY**(on COP880C, COP888FH and COP888GG EPUs)  
**COP87Lx ONLY**    (on the COP8SAC and COP8SGR EPUs)

---

# Chapter 4: Software Guide

## Overview/Features

---

This chapter is an overview of some of the features of the software and a guide to help you understand the layout of the screen. Note that context sensitive help is available for every command.

**Windows** You can have any number of windows of any type, including multiple windows of the same type, open at the same time.

All windows are updated after an emulation cycle or after something changes (e.g., after you explicitly change the value in a register or memory location during Break Condition).

To cycle through the existing windows:

CTRL+F6: cycle forward  
CTRL+SHIFT\_F6: cycle backward

This is the Microsoft Windows convention for cycling through MDI (Multi-Document Interface) child windows.

The software implementation “model” most closely resembles an MDI application, with the following enhancements:

- 1) The “child” windows (all windows except the Main Window) are not constrained to be within (“be clipped by”) the Main Window. They can be anywhere on the screen. (For comparison: Open several documents in a word processor and move the document windows around.)
- 2) You can have as many “child” windows as you want, of a given type, open at the same time (e.g., 5 Code Memory windows, each viewing the same/different/overlapping memory addresses).
- 3) If you turn off the ‘Configure|Preferences|Move Windows with Main’ option, the Main Window can be moved independently of any “child” windows. Currently, re-sizing the Main Window is always done independently of any “child” windows.

Additionally, the Main Menu window will never be “on top of” any of the “child” windows.

You can also pick a specific existing window by selecting it from the bottom of the “Windows” pull-down menu.

**Double Click** In general, double-click (left mouse button) on a symbol name, register name or memory-location to pop-up the “Display/Alter|Expression” dialog box, which will allow you to view, change or browse the value.

NOTE: Currently, right-clicking in a Browse Window (browse data structure) does pop-up a properties menu showing everything you can do to the selected item (e.g., change (modify), or browse further (if it is a pointer, structure or array).

**Save Settings On Exit** Currently, the "Save Settings on Exit" function saves the preferences, the name of the previously loaded program file and the position, size and other information for each window.

These settings can optionally be restored (re-established) whenever you restart the software based on the "Restore Settings at Startup" and "Reload Code File at Startup" settings.

A detailed list of exactly what settings are saved is displayed in the on-line help for the "Save Settings on Exit" command.

**Host Break** There are several ways to stop emulation if no breakpoint is set or reached during emulation:

- 1) Click on the ‘Stop’ toolbar button.
- 2) Select the ‘Stop(Esc)’ command in the ‘Emulating’ window.
- 3) Select the ‘Run|Stop’ command from the Main Menu window.
- 4) Press the ‘Esc’ key.

### **Automatic Configuration**

If the emulator is powered up when the software is invoked, communication between the emulator and the software will be established automatically.

**Help** Detailed and context sensitive **Help** is available on-line using the Help Key (F1).

**Dynamically Annotated Code** When emulating using single steps (*Step* command) or slow motion mode (*Slow Motion* command), the right side of the Source Window is used to display a history of execution for each instruction executed. This history contains the value (before execution of the instruction) of any address, register and Bit used by the instruction. In addition, if the instruction is an unconditional jump instruction or a conditional jump, where the condition is met (TRUE), the Target Address and an arrow indicating direction of program flow will be displayed. If the instruction is a conditional jump, where the condition is not met (FALSE), a \* is displayed.

## Default Screen Layout

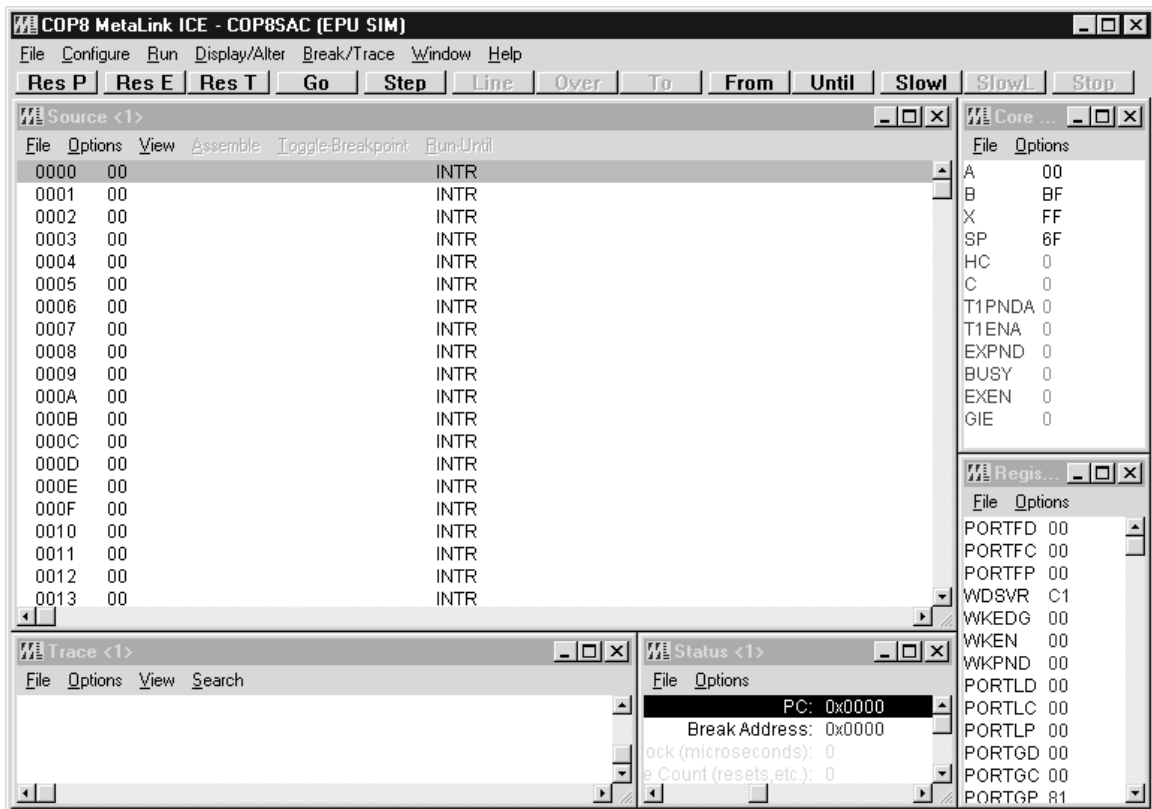


Figure 3. Default Screen Layout

The picture above shows the default screen layout. The windows shown (Source, Trace, Core Registers, Registers (SFR), and Status) are just a subset of the available windows. The available windows are described below.

### Available Windows

- Break** The Break Window is used to display, add, remove or edit break-points. Break-points are evaluated and transmitted to the emulator as they are created or edited.
- Browse** The Browse Window is used to inspect or change structures, unions, arrays, pointers and bit-fields. For each element of the object being browsed, at least the address, data type and value are displayed and where meaningful, array subscript values and member names are also displayed.
- Code Memory** The Code Memory Window displays memory as hexadecimal bytes and their ASCII equivalent. In addition, from the Code Memory Window you may fill or copy blocks of memory, compare two blocks of memory and search memory for a value (or values) match/mismatch.

<b>Core Register</b>	The Core Register Window is used to display the core architecture registers and bit flags.
<b>Emulating</b>	The Emulating Window is displayed when an emulation is started using any run-time commands. It displays status information about the current emulation cycle.
<b>Identification</b>	The Identification Window describes the properties of your emulator system. The information displayed includes such things as hardware, firmware and software versions numbers, memory sizes, model numbers and option configurations. Such information is useful, for example, when you call for technical support.
<b>Program Structure</b>	The Program Structure Window displays module, source line number or source scope information for the loaded program, if available.
<b>RAM Memory</b>	The RAM Memory Window displays memory as hexadecimal bytes and their ASCII equivalent. In addition, from the RAM Memory Window you may fill or copy blocks of memory, compare two blocks of memory and search memory for a value (or values) match/mismatch.
<b>Register (SFR) Source</b>	The Register (SFR) Window is used to display the special function registers (SFRs). The Source Window is used to display code memory as assembly level instructions, optionally with HLL source images, if available.
<b>Stack</b>	The Stack Window displays memory at the stack as hexadecimal bytes and their ASCII equivalent. In addition, from the Stack Window you may fill or move blocks of memory, compare two blocks of memory and search memory for a value (or values) match/mismatch.
<b>Status</b>	The Status Window displays status information such as the PC address, break address, real-time clock, inactive count, pass count, repetition count, emulation status, trace status, trace read percentage and trace trigger.
<b>Symbols</b>	The Symbols Window displays symbolic information for the loaded program, if available. Several display formats are available.
<b>Trace</b>	The Trace Window is used to display a trace of the most recent emulation. Several display formats are available. Note that the Trace Window is available only on those emulator systems that have trace memory.
<b>Watch</b>	The Watch Window is used to display information about watch expressions. You can think of watch expressions as peepholes into memory where you specify the starting address symbolically (the name of a program variable, register or bit) or numerically (an expression) and where the displayed values are interpreted according to the data type of the expression (if available).

---

# Chapter 5: Operational Considerations

## In-Circuit Simulation Limitations

---

The iceMASTER EPU-COP8 actually uses a COP880C, a COP888FH, a COP888GG, COP8SAC or a COP8SGR for simulation. However, since the EPU is controlled through the software, it has the following constraints:

- 1) The INTR instruction is used to implement software break-points.
- 2) If your application code contains INTR instructions, **they will never be executed**. Simulation will stop when an INTR is encountered.
- 3) If a break-point is set on an instruction which could potentially be skipped, simulation will break only when that instruction is actually executed. Simulation will never break on a skipped instruction.

When a break-point is set on an instruction which could potentially be skipped and that instruction is skipped during simulation, the address and data values captured in the trace will be different from the full featured iceMASTER COP8 (Model 400) emulator. The first cycle will be that of a skipped INTR instruction. For multi-byte instructions, subsequent cycles will be the same as that of executed NOP instructions.

- 4) When a break-point is set on an instruction, simulation breaks before that instruction executes. The two (2) bytes beyond the top of the stack will be overwritten when a break-point is reached. This is due to the fact that INTR instructions are used to implement break-points.
- 5) Upon reaching a break-point, the timer is shut off shortly after simulation stops. When simulation resumes, the timer is restarted just before simulation in the target application program actually begins.
- 6) HALT Mode. To allow clock re-synchronization in the COP8 microcontroller, it is necessary to program two NOP's immediately after the processor comes out of HALT mode. If the multi-input wake-up interrupt is enabled, the first two instructions of the interrupt routine must be NOP's. If no interrupt is used, then two NOP's must follow the "enter HALT mode" (set G7 data bit) instruction.

Note that on the COP888FH and COP888GG the microcontroller is in crystal oscillator mode, therefore the G7 pin is not available for I/O or HALT restart.

- 7) IDLE Mode (COP888FH, COP888GG, COP8SAC and COP8SGR only). To allow clock re-synchronization in the COP8 microcontroller, it is necessary to program two NOPs immediately after the processor comes out of IDLE mode. If the IDLE timer

interrupt is enabled, the first two instructions of the interrupt routine must be NOP's. If no interrupt is used, then two NOPs must follow the "enter IDLE mode" (set G6 data bit) instruction.

- 8) During simulation, the status message box

#### **Emulation Processor Inactive**

may be displayed. This status message indicates that the COP8 simulation processor is not executing instructions. Normally this will be due to an extended RESET pulse, or to the entry of HALT or IDLE mode. If you execute a Host-break (i.e., manually stopping simulation by pressing **Esc**), a Confirmation Box will prompt you with the following:

#### **Processor not running, do you want to RESET to break emulation?**

A **Y** response will reset the system and simulation will break at address 0 (zero). An **N** or **Esc** response will leave the device in its current state. If the device is in HALT or IDLE mode it will return to HALT or IDLE mode, therefore no further instructions are executed and no break-points will be encountered.

- 9) Watchdog Timer/Clock Monitor (COP888FH, COP888GG, COP8SAC and COP8SGR only). Due to the method of in-circuit simulation used on the COP888FH, COP888GG, COP8SAC and COP8SGR EPU's the Clock Monitor cannot be defeated. This means that G1 will always be asserted and therefore the COP888FH, COP888GG, COP8SAC and COP8SGR EPU's cannot support the Watchdog Timer or Clock Monitor feature of the microcontrollers.

## **Memory Limitations**

---

**The iceMASTER EPU-COP8 allows up 32K bytes of memory for programming. However, do not do code simulation in the last 256 bytes of the 32K byte memory space (0x7F00 to 0x7FFF) as unpredictable operation may occur.**

## **Static**

---

Perhaps the most difficult problem anyone who uses MOS devices will face is static. You may go for years with no fault traceable to static, or you may damage devices frequently. The iceMASTER EPU-COP8 can be as sensitive to static as any other circuit. The microcontroller devices used in the EPU are especially vulnerable since adding extra protection would change response characteristics. The built-in protections internal to the devices are operative.

We do still recommend, however, that you take every precaution regarding static. The use of grounding straps, static free workstations, and a little extra care in handling the emulator (and any MOS part) can prevent most problems.

---

## Chapter 6: Contacting National Semiconductor

### Phone

---

North America (Canada and United States)

1-800-272-9959

Germany

49-8141-103-0

Hong Kong

852-737-1800

### National's Microcontroller Bulletin Board

---

In North America (Canada and United States)

1-800-NSC-MICRO

1-800-672-6427

Use the following communication parameters:

8 data bits

1 stop bit

No parity

### Internet

---

Telnet: nscmicro.nsc.com

user: anonymous

password: username@host

FTP: nscmicro.nsc.com

user: anonymous

password: username@host

## Chapter 7: Differences (Model 400 vs. Debug Module vs. EPU)

The following table is a comparison of the features in the iceMASTER COP8 Model 400 In-Circuit Emulator, the iceMASTER COP8 Debug Module and the iceMASTER EPU-COP8.

Feature/Attribute/Command	Model 400	Debug Module	EPU
PROM Programmer	no	yes	yes
Interchangeable Probe Cards	yes	no	no
Number of Trace Frames	4K	100	100
Number of Break-points	64K	32K	32K
Type of Break-point	HW	SW (INTR instruction)	SW (INTR instruction)
Number of Trace-On-points	64K	0	0
Number of Trace-Off-points	64K	0	0
Number of Increment-Pass-Count-points	64K	0	0
Real-Time Clock during emulation	yes	yes	no <sup>1</sup>
Reset Counter during emulation	yes	yes	no <sup>2</sup>
Pass Counter during emulation	yes	no <sup>3</sup>	no <sup>3</sup>
Trace Triggers	yes	no	no
Probe Clips	yes	no <sup>4</sup>	no <sup>4</sup>
Break Button	yes	no <sup>4</sup>	no <sup>4</sup>
Maximum RS-232 Baud Rate	115,200 baud	57,600 baud	115,200 baud
Multiple Watch Windows	yes	yes	yes
Multiple Internal Data Windows (hex “dump” style)	yes	yes	yes
Multiple Code Windows (hex “dump” style)	yes	yes	yes
Multiple Command Windows	yes <sup>5</sup>	yes <sup>5</sup>	yes <sup>5</sup>
Multiple Frequency Operation	yes <sup>6</sup>	yes <sup>7</sup>	no

<b>Feature/Attribute/Command</b>	<b>Model 400</b>	<b>Debug Module</b>	<b>EPU</b>
<i>File Store</i>	yes	yes	yes
<i>File Restore</i>	yes	yes	yes
<i>File Macro</i>	no <sup>5</sup>	no <sup>5</sup>	no <sup>5</sup>
<i>Run From</i>	yes	yes	yes
<i>Run Until</i>	yes	yes	yes
<i>Run Slow Motion Instruction</i>	yes	yes	no
<i>Run Slow Motion Line</i>	yes	yes	no
<i>Run Step Line</i>	yes	yes	yes
<i>Run Step Over</i>	yes	yes	yes
<i>Run Step To</i>	yes	yes	yes
<i>Trace Search</i>	yes	yes	yes

1. Real-Time Clock: The “Real-Time Clock:” field in the Status Window is blank. However, the “heartbeat” wheel spins during emulation. The same is true of the pop-up “Emulating:” box during emulation.
2. Reset Counter: The “Inactive Count:” field in the Status Window is blank.
3. Pass Counter: The “Pass Count:” field in the Status Window is blank.
4. The Debug Module and EPU have no probe clips. The Debug Module does provides a single “Break Input” point.
5. The Macro feature of the emulator software for DOS is not available in the Windows-based emulator software. Instead, a command window is available.
6. Multiple frequency operation is available when using the crystal from the target system.
7. On DM3 and DM4 Debug Modules, multiple frequency operation is available when using the crystal from the target system. Multiple frequency operation is provided via a programmable clock on the DM5, and is controllable using the emulator software.

**Table 2. Model 400, Debug Module, EPU Differences**