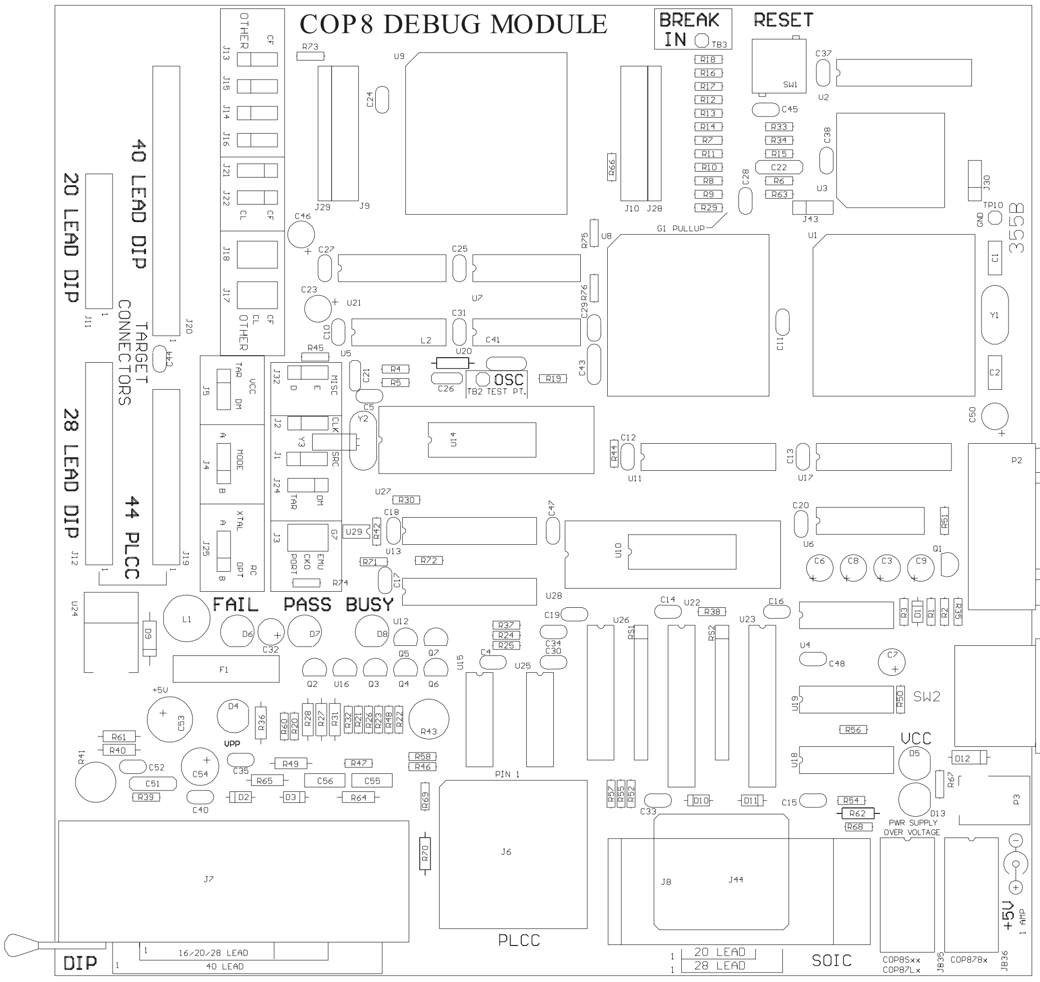
iceMASTER® Debug Module User's Manual for Microsoft® Windows®

MetaLink® Corporation



Document Version: 1.02

Table Of Contents

Chapter 1: Introduction

The Emulator

The iceMASTER COP8 Debug Module is a tool for designing, debugging, programming and evaluating COP8 Microcontroller Unit (MCU) devices. By providing all of the essential MCU timing and I/O circuitry, the Debug Module simplifies evaluation of the prototype hardware/software product.

The iceMASTER COP8 Debug Module is an in-circuit emulator controlled by an IBM PC (or compatible) running the Windows operating system. The iceMASTER COP8 Debug Module is an integral part of the development engineer's toolbox, with applications in device evaluation, software development and hardware integration.

The Debug Module can be connected to a target system in place of the microcontroller (using an optional Target Interface Cable), or operated independently in stand alone mode. Stand alone mode allows you to emulate hardware and/or execute code without a target system (provided no interaction with external devices is needed).

Hardware designers may use the Debug Module to develop and debug their designs, including programming their software into COP8 EPROM devices. All available features of a given device are accessible interactively, as well as through your application programs. Software designers have complete emulation capability as well.

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 iceMASTER COP8 Debug Module's capabilities; however, basic programming knowledge and familiarity with the microcontroller architecture are assumed.

Chapter 2: Hardware Installation

Before starting the hardware installation, verify which type of Debug Module that you have. The type of Debug Module (DM3, DM4 or DM5) can be determined from the paper label on the corner of the board nearest the Reset Switch.

If there is no label on the corner, the type of Debug Module may be determined by the presence (or absence) of the 44-pin PLCC LIF programming socket, the 4 position/post miniature terminal block, the yellow LED next to the power connector and the J1, J2, J24 jumber block group. If present, the 44-Pin PLCC socket is between the SOIC socket and the DIP socket, the terminal block is between the RS232 connector and the power switch and the yellow “PWR SUPPLY OVER VOLTAGE” LED is next to the power connector.

The following table shows how to determine which Debug Module you have. Note that these are not the only differences between the Debug Modules but are all that are needed to distinguish between them.

	Debug Module			
	DM3	DM4 (Type 1)	DM4 (Type 2)	DM5
4 Position Terminal Block	Yes	No	No	No
44-Pin PLCC LIF Socket	No	Yes	Yes	Yes
Yellow LED (next to power connector)	No	No	Yes	Yes
J1, J2, J24 Jumper Block Group	Yes	Yes	Yes	No*

* - If present, the jumper block group will be hardwired and will not be selectable.

Table 1. DM3/DM4/DM5 Determination

DM3 Installation

Connect one end of the RS-232 serial cable to a serial communication port on the Host Computer. Connect the other end of the RS-232 cable to the Debug Module.

Power may be supplied through the power receptacle or the 4 position/post terminal block.

To use the power receptacle, simply connect the power supply to the Debug Module by inserting the power supply’s connector into the power receptacle on the Debug Module.

Alternately, when using the 4 position/post terminal block, insert the ground wire into one of the terminals marked “GND” (the two “GND” terminals are connected) and then tighten the corresponding set screw. Insert the +5 volt source into the terminal marked “+5V” and then tighten the set screw.

If you are planning on programming COP8 EPROM devices see the PROM Programmer Chapter for a complete description on how to supply the necessary programming voltage.

NOTE: For safety, we recommend that all items in your system, including Debug Module, Host Computer and target, be connected to the same outlet. Different outlets (even though near one another) may be connected to different circuits in the building, resulting in large potential differences between grounds.

For a complete description of all hardware components of the Debug Module see the Hardware Description Chapter 3.

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

DM4, DM5 Installation

Connect one end of the RS-232 serial cable to a serial communication port on the Host Computer. Connect the other end of the RS-232 cable to the Debug Module.

Connect the power supply to the Debug Module by inserting the power supply’s connector into the power receptacle on the Debug Module.

NOTE: For safety, we recommend that all items in your system, including Debug Module, Host Computer and target, be connected to the same outlet. Different outlets (even though near one another) may be connected to different circuits in the building, resulting in large potential differences between grounds.

For a complete description of all hardware components of the Debug Module see the Hardware Description Chapter 3.

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

Chapter 3: Hardware Description

The chapter describes the different types of Debug Module. The type of Debug Module can be determined from the paper label on the corner of the board nearest the Reset Switch.

If there is no label on the corner and you are not sure which type you have then please see the table at the beginning of Chapter 2.

The rest of this chapter is broken into two major sections, the first section has a diagram of each type of Debug Module and the other contains a description of the components of the Debug Module board.

The major sections are:

1) Debug Module Diagrams (starting on page 11)

2) Debug Module Component Description (starting on page 14)

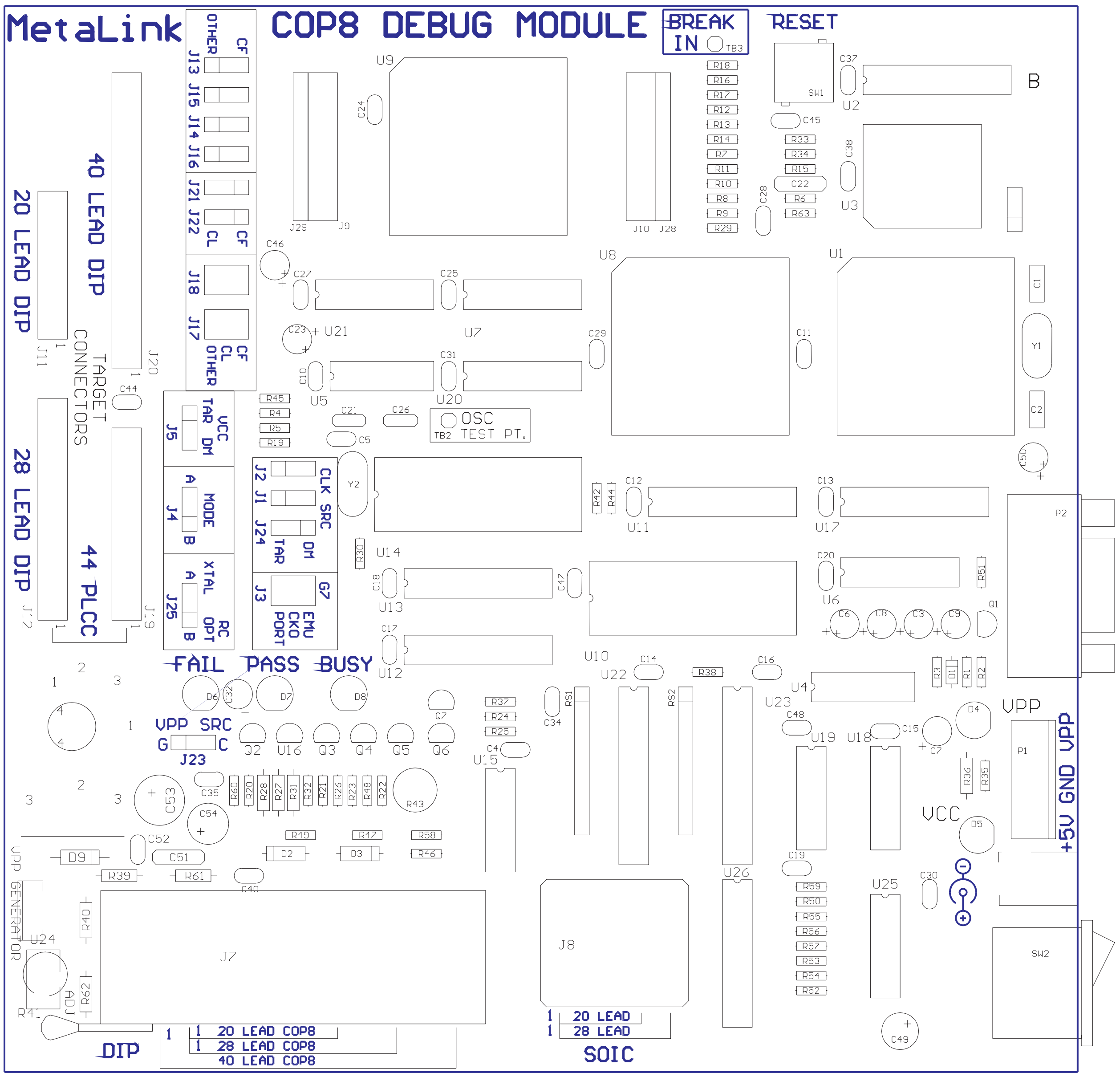


Figure 1. DM3 Debug Module

DM4 Debug Module

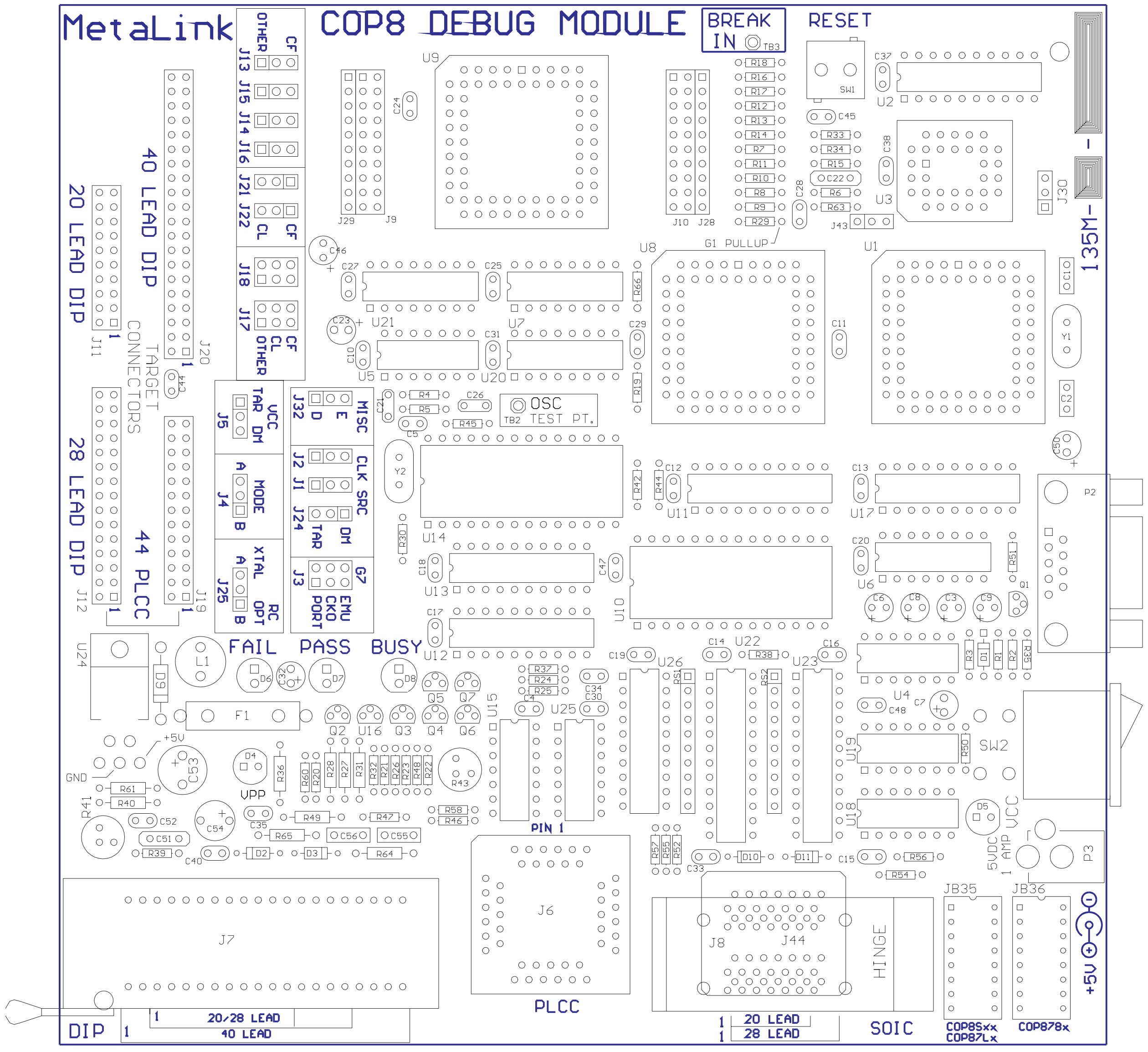


Figure 2. DM4 Debug Module

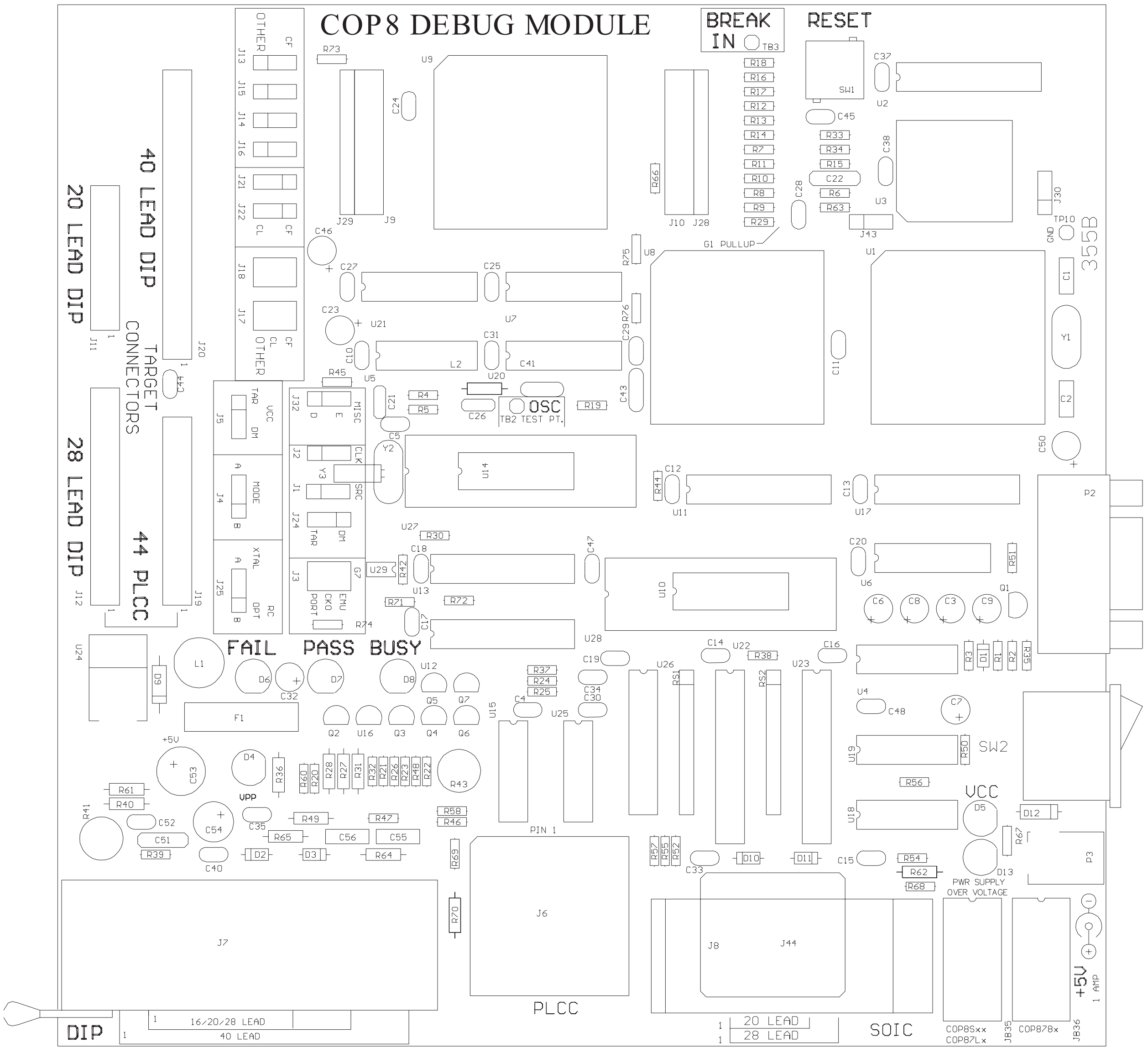


Figure 3. DM5 Debug Module

Debug Module Component Description

Buttons/Switches

SW1 RESET All Debug Modules. The RESET pushbutton allows you to reset the emulation microcontroller during emulation, independent of the target RESET pin.

SW2 POWER All Debug Modules. The POWER rocker switch controls power coming in on the power connector.

EPROM Programming Sockets

Three sockets are provided for programming COP8 EPROM devices

44-pin PLCC LIF DM3 Debug Modules. The 44-pin PLCC socket is not available on DM3 Debug Modules.

DM4 and DM5 Debug Modules. The 44-pin PLCC socket is available to program 44-pin PLCC devices.

40-pin DIP ZIF DM3 Debug Modules. The 40-pin DIP socket is available to program 20, 28 and 40-pin DIP devices.

DM4 and DM5 Debug Modules. The 40-pin DIP socket is available to program 16, 20, 28 and 40-pin DIP devices.

28-pin SOIC ZIF DM3 Debug Modules. The 28-pin SOIC socket is available to program 20 and 28-pin SOIC devices.

DM4 and DM5 Debug Modules. The 28-pin SOIC socket is available to program 16, 20 and 28-pin SOIC devices.

Note that some devices require programming adapters. The Host Software will notify you of any requirements when you select a device and when you select a programming command. For a list of all devices that may be programmed select the *Device* list box in the *PROM Programmer* Dialog Box. You may write this device list to an ASCII text file by pressing the *Write* button in the *PROM Programmer* Dialog Box. The device list written to disk will also include any special requirements for programming.

LED Indicators

VCC

All Debug Modules. The VCC LED indicates that V_{CC} is being applied to the Debug Module through the power connector.

VPP

DM3 Debug Modules. The VPP LED indicates that V_{PP} is present on the Debug Module. This can come from either the “VPP” post of the terminal block, if external V_{PP} is being applied, or from the On-Board V_{PP} Generator (see the PROM Programmer Chapter).

DM4 and DM5 Debug Modules. The VPP LED indicates that V_{PP} is present on the Debug Module.

PASS
BUSY
FAIL

All Debug Modules. The PASS LED, BUSY LED and FAIL LED indicate the current status of the EPROM Programmer.

PWR SUPPLY OVER VOLTAGE

DM3 and DM4 (Type 1) Debug Modules. There is no PWR SUPPLY OVER VOLTAGE LED on these types of Debug Modules.

DM4 (Type 2) and DM5 Debug Modules. When this LED is lit it indicates that V_{CC} is much greater than is allowed. Immediately power down the Debug Module.

Power

Terminal Block

DM3 Debug Modules. Power may be supplied through the power receptacle or the 4 position/post terminal block.

When using the power receptacle, it is supplied using a standard 2.5mm x 5.5mm x 9.5mm plug and jack, center positive (Switchcraft 760).

When using the 4 position/post terminal block, the minimum wire gauge for all connections is 24.

In either case, the power supply must provide +5 volts ±5%, at 1 ampere. The ripple voltage must be no greater than 50 millivolts, peak-to-peak.

If the J23 Jumper (see the Jumper Block Chapter) is set to C, and you plan on programming EPROM devices, the Debug Module requires that the programming voltage (V_{PP}) be supplied through the “VPP” Post of the terminal block (see the PROM Programmer Chapter).

DM4 and DM5 Debug Modules. There is no Terminal Block on the DM4 and DM5 Debug Modules, power is supplied via the power jack (described below).

Power Jack

DM3 Debug Modules. There is no power jack on the DM3 Debug Modules, power is supplied via the terminal block (described above).

DM4 and DM5 Debug Modules. Power is supplied with a standard 2.5mm x 5.5mm x 9.5mm plug and jack, center positive (Switchcraft 760). The power supply must provide +5 volts ±5%, at 1 ampere. The ripple voltage must be no greater than 50 millivolts, peak-to-peak.

Clock Generation and Selection

DM3 and DM4 Debug Modules. The figure below is a schematic diagram of the clock generation and selection circuit used on the DM3 and DM4 Debug Modules.

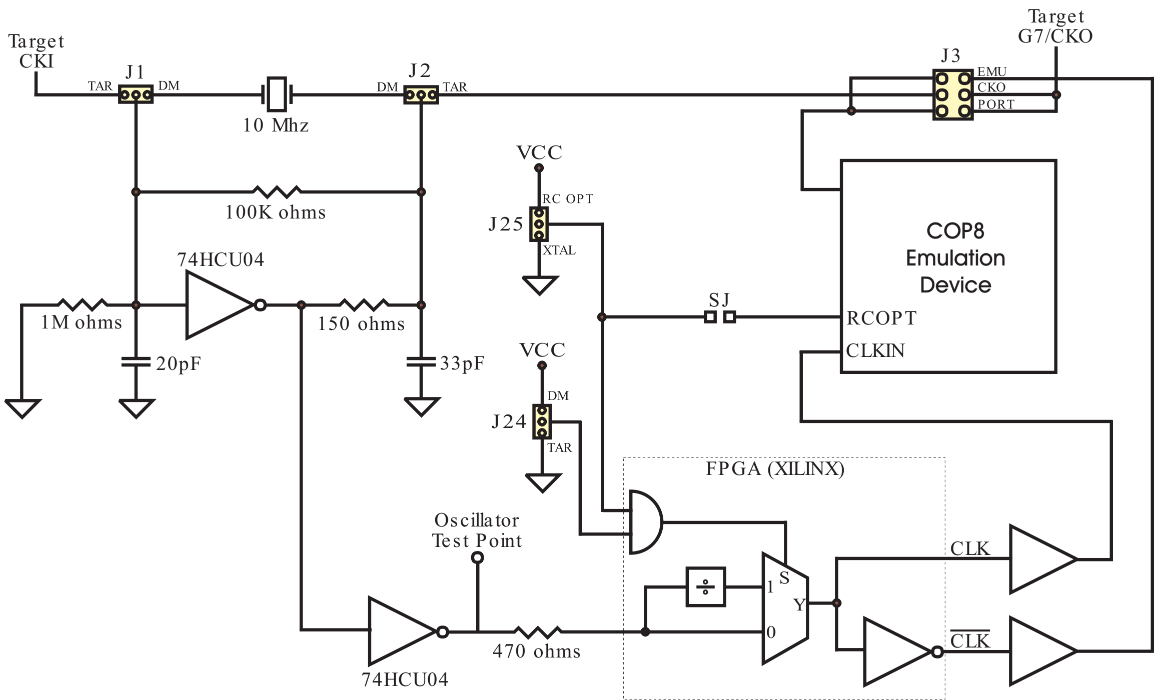


Figure 4. DM3, DM4 Clock Generation and Selection Circuit

As can be seen from the schematic diagram there are five jumper blocks (J1, J2, J3, J24 and J25) involved in the circuit. Using these jumper blocks, any Debug Module can operate in any Clock Mode (Crystal Oscillator, External Oscillator or R/C Oscillator Mode).

The five jumper blocks are required because the COP8 Emulation Device used in the Debug Module actually operates in External Oscillator Mode (for 820CJ, 840CJ, 880, 888CF, 888EG and 8SAx Debug Modules) or Crystal Oscillator Mode (for 888BC, 888CG, 888EB, 888EK, 888FH, 888GD, 888GG, 888GW, 888HG, 888KG and 8AC Debug Modules). The jumpers are used to configure the clock signals to emulate any Clock Mode.

Each jumper block used in this circuit is described in detail in the Jumper Blocks Chapter.

Each Clock Mode and the jumpers used to emulate each Clock Mode are described in detail in the Jumper Block Chapter.

<u>DM5 Debug Modules</u>. The figure below is a schematic diagram of the clock generation and selection circuit used on the DM5 Debug Module.

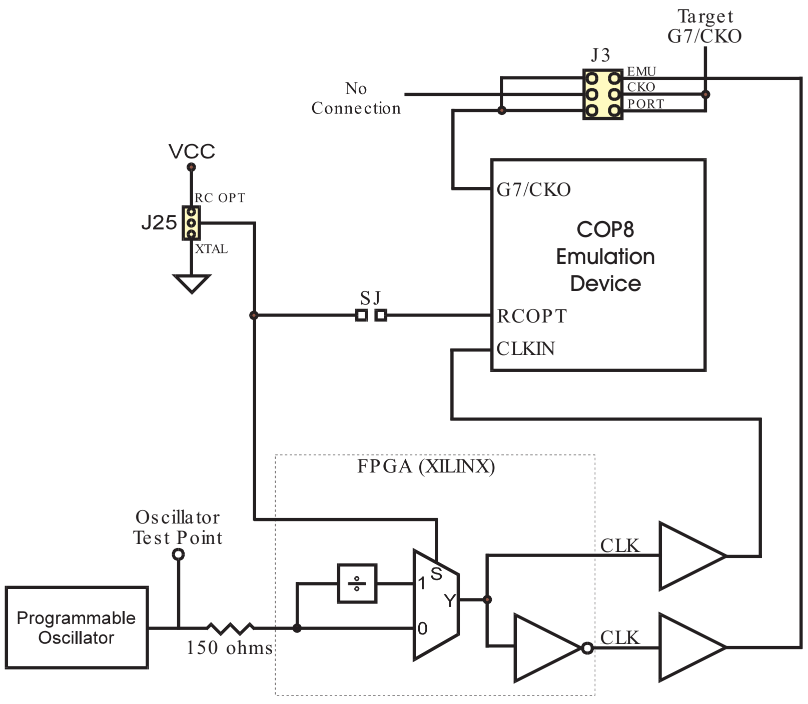


Figure 5. DM5 Clock Generation and Selection Circuit

As can be seen from the schematic diagram there are two jumper blocks (J3 and J25) involved in the circuit. Using these jumper blocks, any Debug Module can operate in any Clock Mode (Crystal Oscillator, External Oscillator or R/C Oscillator Mode).

The two jumper blocks are required because the COP8 Emulation Device used in the Debug Module actually operates in External Oscillator Mode (for 820CJ, 840CJ, 880, 888CF, 888EG, 8FGx, 8SAx and 8SGx Debug Modules) or Crystal Oscillator Mode (for 888BC, 888CG, 888EB, 888EK, 888FH, 888GD, 888GG, 888GW, 888HG, 888KG, 8AC and 8SEx Debug Modules). The jumpers are used to configure the clock signals to emulate any Clock Mode.

Each jumper block used in this circuit is described in detail in the Jumper Blocks Chapter.

Each Clock Mode and the jumpers used to emulate each Clock Mode are described in detail in the Jumper Block Chapter.

Break Input

All Debug Modules. The Break Input signal connector (labeled BREAK IN) is interfaced via a 74HC-type device on the Debug Module. The Break Input is pulled up with a 20K ohm pull-up resistor. When this input is pulled low, the current emulation cycle will break. The Break Input must remain active (LOW) for at least one machine cycle (10 clock cycles).

G0 Pull-Down Resistor

All Debug Modules. If G0 is configured as an external interrupt and the Debug Module is in stand-alone mode (not connected to a target system), noise on G0 may cause spurious interrupts. For this reason, a 47K pull-down resistor has been installed on G0.

Please note this resistor may have some effect on your target system. If you need to remove the resistor from the circuit, it is labeled R18 and is the closest resistor to the BREAK IN post.

G1 Pull-Up Resistor

Many of the devices that the Debug Module emulates require an external pull-up resistor on G1 for the WDOUT (Watchdog Out) signal to function properly when the Debug Module is being operated in stand-alone mode (not connected to a target system). For this reason, a 47K pull-up resistor has been installed on G1.

DM3 Debug Modules. Please note this resistor may have some effect on your target system. If you need to remove the resistor from the circuit, it is labeled R29 and is next to the XILINX chip in the line of resistors between the BREAK IN post and the XILINX chip.

DM4 and DM5 Debug Modules. Please note this resistor may have some effect on your target system. If you need to remove the resistor from the circuit, it is labeled G1 PULLUP (and R29) and is next to the XILINX chip.

RS-232 Interface

All Debug Modules. The communication link to the Host Computer is based on the serial

RS-232-C 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 manually.

Host PC Cable Connector			Cable		DM Cable Connector		
Signal	Pin		Function	Direction	Pin		Signal
	Female DB-25	Female DB-9			Male DB-9	Male DB-25	
TxD	2	3	Data to DM	$\rightarrow$	3	2	RxD
RxD	3	2	Data to Host	$\leftarrow$	2	3	TxD
RTS	4	7	Reset DM - active low	$\rightarrow$	7	4	RTS
CTS	5	8	ALE to HOST	$\leftarrow$	8	5	CTS
Ground	7	5	DC Ground	$\leftrightarrow$	5	7	Ground
DTR	20	4	Handshake	$\rightarrow$	4	20	DTR

Table 2. DM3, DM4, DM5 RS-232 Interface Connections

The cable mates via a 9-pin male DB-9 connector on the cable at the Debug Module end.

The table above specifies the pin assignments for a 25-pin male DB-25 connector on the Debug Module end. These assignments are compatible with standard DB-25 to DB-9 inline converters.

At the host end, the mating connector on the cable may be a 25-pin female DB-25 connector or a 9-pin female DB-9 connector.

Note that, in the 9-pin RS-232-C interface, pins 2 and 3 are reversed from their normal 25-pin D connector assignments.

The total length of the cable should not exceed 25 feet.

Target Interface Cables

All Debug Modules. One or more target interface cables are shipped with each Debug Module. Which cables are shipped depends on which emulation device your Debug Module supports. The following target interface cables are available:

1) 20-lead DIP

2) 28-lead DIP

3) 40-lead DIP

4) 44-lead PLCC

5) 68-lead PLCC

Shunt Blocks

DM3 Debug Modules. There are no shunt blocks on the DM3 Debug Module.

DM4 and DM5 Debug Modules. There are two shunt blocks on the DM4 Debug Module. The shunt blocks are labeled "COP8Sxx/COP87Lx" and "COP878x" and are used only during programming. You will be told to insert a shunt into one of the shunt blocks when a device to program is selected.

Jumper Blocks

Refer to the Jumper Blocks Chapter for a full description of all jumper blocks.

PROM Programmer

Refer to the PROM Programmer Chapter for information about using the PROM programming capabilities of the Debug Module.

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 foreword
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.

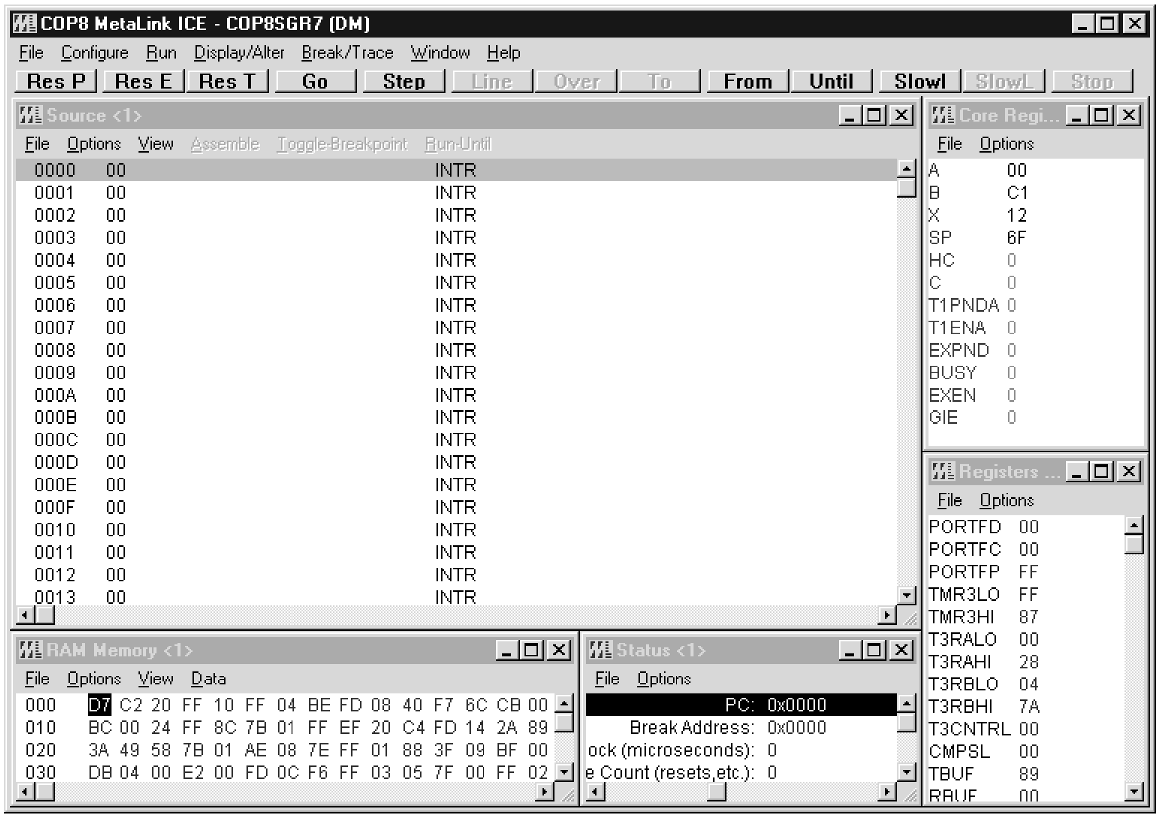
Help

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


COP8 MetaLink ICE - COP8SGR7 (DM)
File Configure Run Display/Alter Break/Trace Window Help
Res P Res E Res T Go Step Line Over To From Until Slowl SlowL Stop
Source <1>
File Options View Assemble Toggle-Breakpoint Run-Until
Core Regi...
File Options
Registers ...
File Options
RAM Memory <1>
File Options View Data
Status <1>
File Options

```
0000  00      INTR
0001  00      INTR
0002  00      INTR
0003  00      INTR
0004  00      INTR
0005  00      INTR
0006  00      INTR
0007  00      INTR
0008  00      INTR
0009  00      INTR
000A  00      INTR
000B  00      INTR
000C  00      INTR
000D  00      INTR
000E  00      INTR
000F  00      INTR
0010  00      INTR
0011  00      INTR
0012  00      INTR
0013  00      INTR
```

```
A        00
B        C1
X        12
SP       6F
HC       0
C        0
T1PNDA   0
T1ENA    0
EXPND    0
BUSY     0
EXEN     0
GIE      0
```

```
PORTFD   00
PORTFC   00
PORTFP   FF
TMR3LO   FF
TMR3HI   87
T3RALO   00
T3RAHI   28
T3RBLO   04
T3RBHI   7A
T3CNTRL  00
CMPSL    00
TBUF     89
RBUF     00
```

```
000   D7 C2 20 FF 10 FF 04 BE FD 08 40 F7 6C CB 00
010   BC 00 24 FF 8C 7B 01 FF EF 20 C4 FD 14 2A 89
020   3A 49 58 7B 01 AE 08 7E FF 01 88 3F 09 BF 00
030   DB 04 00 E2 00 FD 0C F6 FF 03 05 7F 00 FF 02
```

```
                    PC:  0x0000
         Break Address:  0x0000
ock (microseconds):  0
e Count (resets,etc.):  0
```

Figure 6. Default Screen Layout

The picture above shows the default screen layout. The windows shown (Source, RAM Memory, 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.

Core Register	The Core Register Window is used to display the core architecture registers and bit flags.
Emulating	The Emulating Window is displayed when an emulation is started using any run-type commands. It displays status information about the current emulation cycle.
Identification	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.
Program Structure	The Program Structure Window displays module, source line number or source scope information for the loaded program, if available.
RAM Memory	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.
Register (SFR)	The Register (SFR) Window is used to display the special function registers (SFRs).
Source	The Source Window is used to display code memory as assembly level instructions, optionally with HLL source images, if available.
Stack	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.
Status	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.
Symbols	The Symbols Window displays symbolic information for the loaded program, if available. Several display formats are available.
Trace	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.
Watch	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: Jumper Blocks

This chapter contains detailed descriptions of Debug Module Jumpers. The following describes the general layout of this chapter:

- As-Shipped Chart (page 26)
- Options By Jumper Block (page 28)
- Options By Function (page 35)

The “As-Shipped Chart” section is a chart of every jumper setting as shipped for each type of Debug Module sold.

The “Options By Jumper Block” section contains a description of each jumper block, where a jumper block is defined by each column heading in the “As-Shipped Chart”.

The “Options By Function” section contains a description of chip functions/features in which more than one jumper block are used, such as Clock Modes.

As-Shipped Chart

Debug Module Type	Emulation Devices Supported	Brown[1] Out	J1,J2,J24[2] (Clock Source)	J3 (G7 Option)	J4 (Mode)	J5 (VCC)	J13-J16 (CF Emul.)	J17,J18 (CF/CL Emul.)	J21,J22 (CF/CL Emul.)	J23[3] (VPP Source)	J25 (Clock Mode)	J26[4] (HALT)	J32[5] (Misc)	J35[6] (A/D Ref)
820CJ	82xCJ	DIS	DM[7]	CKO[7]	N/F	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
840CJ	84xCJ	DIS	DM[7]	CKO[7]	N/F	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
880	82x	N/A	DM[7]	CKO[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
880	84x	N/A	DM[7]	CKO[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
880	88x	N/A	DM[7]	CKO[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888BC	88xBC	N/A	DM[7]	CKO+EMU[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888CF	88xCF	N/A	DM[7]	CKO[7]	A	DM	‡CF	‡CF	‡CF	G	XTAL[7]	N/A	Open	N/A
888CF	88xCL	N/A	DM[7]	CKO[7]	A	DM	‡CF	‡CL	‡CL	G	XTAL[7]	N/A	Open	N/A
888CG	88xCG	N/A	DM[7]	CKO+EMU[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888EB	88xEB	N/A	DM[7]	CKO+EMU[7]	N/F	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	DM
888EG	88xCG	N/A	DM[7]	CKO[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888EG	88xCS	N/A	DM[7]	CKO[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888EG	88xEG	N/A	DM[7]	CKO[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888EK	88xEK	N/A	DM[7]	CKO+EMU[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888FH	88xFH	N/A	DM[7]	CKO+EMU[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888GD	88xGD	N/A	DM[7]	CKO+EMU[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888GG	88xGG	N/A	DM[7]	CKO+EMU[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888GW	88xGW	N/A	DM[7]	CKO+EMU[7]	N/F	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	Enabled	Open	N/A

Debug Module Type	Supports $Model Files	Brown[1] Out	J1,J2,J24[2] (Clock Source)	J3 (G7 Option)	J4 (Mode)	J5 (VCC)	J13-J16 (CF Emul.)	J17,J18 (CF/CL Emul.)	J21,J22 (CF/CL Emul.)	J23[3] (VPP Source)	J25 (Clock Mode)	J26[4] (HALT)	J32[5] (Misc)	J35[6] (A/D Ref)
888HG	88xHG	N/A	DM[7]	CKO+EMU[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
888KG	88xKG	N/A	DM[7]	CKO+EMU[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
8AC	8AC	N/A	DM[7]	CKO+EMU[7]	N/F	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	Open	N/A
8FGR	8FGx	N/A	DM[7]	CKO[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	D	N/A
8SAC	8SAx	N/A	DM[7]	CKO[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	D	N/A
8SER	8SEx	N/A	DM[7]	CKO+EMU[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	D	N/A
8SGR	8SGx	N/A	DM[7]	CKO[7]	A	DM	‡OTHER	‡OTHER	‡CF	G	XTAL[7]	N/A	D	N/A

NOTES:
‡ = Factory Setting (Do Not Change)
N/F = No Function
N/A = Not Applicable (does not exist on specified Debug Module)
1. Brown Out jumper exists only on 820CJ and 840CJ Debug Modules (on daughter card).
2. J1, J2 and J24 jumpers are hardwired to DM on DM5 Debug Modules.
3. J23 (VPP Source) jumper exists only on DM3 Debug Modules.
4. J26 (HALT) jumper exists only on 888GW Debug Module (on daughter card).
5. J32 (Misc) jumper exists only on DM4 and DM5 Debug Modules.
6. J35 (A/D Ref) jumper exists only on 888EB Debug Module (on daughter card).
7. See Clock Mode descriptions later in this chapter for a description of all clock modes.

Table 3. As-Shipped Jumper Chart

Options By Jumper Block

BROWN OUT

820CJ/840CJ Debug Modules. The **BROWN OUT** jumper is located on the 820CJ/840CJ daughter card next to the emulation microcontroller. It allows you to emulate the Brown Out option of the microcontroller. When enabled, the Brown Out feature will be emulated. When disabled, the Brown Out feature will NOT be emulated.

Brown Out Feature	BROWN OUT
Disabled	DIS
Enabled	EN

Table 4. BROWN OUT Jumper

Note that the nominal Brown Out voltage for the 820CJ is approximately 2.3 VDC and for the 840CJ is approximately 3 VDC. These voltage levels are due to operating characteristics of the Debug Module and may differ from final production parts.

Other Debug Modules. There is no **BROWN OUT** jumper on other Debug Modules.

J1,J2,J24 Clock Source

DM3 and DM4 Debug Modules. The **Clock Source** jumpers allow the clock source (CKI) to be supplied from either the Debug Module (DM) or the target system (TAR). The clock generation circuit is used on the Debug Module to emulate the oscillator circuit of the microcontroller.

Oscillator Source	J1	J2	J24
Debug Module	DM	DM	DM
Target System	TAR	TAR	TAR

Table 5. Clock Source Jumper

When the DM setting is selected for J1, J2 and J24, a 10 MHZ crystal is connected to the circuit. If J25 is set to XTAL than a fixed frequency of 10 MHz is supplied to the emulation microcontroller. If J25 is set to RC OPT, then the 10 MHz signal is routed through a divide circuit in the XILINX device to supply a fixed frequency of 5 MHz (for COP8SAC and COP8SGR microcontrollers) or 3.33 MHz (for all other COP8 microcontrollers) is supplied to the emulation microcontroller.

When the TAR setting is selected for J1, J2 and J24, the target system must provide either a crystal or clock driver input. The maximum frequency is 5 MHz (for COP8SAC and COP8SGR microcontrollers) or 3.33 MHz (for 888BC, 888CG, 888EB, 888EK, 888FH, 888GD, 888GG, 888GW, 888HG, 888KG, 8AC and 8SER microcontrollers) or 10 MHz (for all other microcontrollers). Note that in this mode, the Target Interface Cable must not exceed 6 inches in length.

For a complete description of Clock Modes see the Clock Modes description.

DM5 Debug Modules. The J1, J2 and J24 jumpers are hardwired to DM for the DM5 Debug Module. This is because the DM5 has a programmable clock that can supply any frequency from 390 Hz to 30 MHz (note that the operating range for specific devices is enforced in the software).

J3
G7 Option

The **G7 Option** jumper allows you to emulate the G7/CKO option of the target microcontroller. When emulating the CKO option, the Debug Module oscillator is driven out the target G7 pin and G7 is NOT available as an input pin and cannot be used to take the emulation microcontroller out of HALT mode (if available).

When emulating the INPUT/HALT bond out option, the target G7 pin is routed directly to the emulation microcontroller's G7 pin. This allows G7 to be used as an input pin or to take the emulation microcontroller out of HALT mode.

For a complete description of Clock Modes see the Clock Modes description.

820CJ, 840CJ, 880, 888CF, 888EG, 8FGR, 8SAC, 8SGR Debug Modules

G7 Option	J3
CKO	CKO[1]
INPUT/HALT	PORT

1. Warning: The EMU jumper position MUST be left OPEN for these Debug Modules.

Table 6. G7 Option Jumper

888BC, 888CG, 888EB, 888EK, 888FH, 888GD, 888GG, 888GW, 888HG, 888KG, 8AC, 8SER Debug Modules

G7 Option	J3
CKO	CKO+EMU
INPUT/HALT	PORT

Table 9. G7 Option Jumper

J4
Mode

The function of the **Mode** jumper depends on the type of Debug Module.

880, 8SAC Debug Modules. The **Mode** jumper determines how much on-chip internal RAM is available in the emulation microcontroller.

RAM Size (bytes)	J4 (Mode)	880 Supports	8SAx Supports
128	A	880,840	8SAC,8SAB
64	B	820	8SAA

Table 7. Mode Jumper (RAM Size)

820CJ, 840CJ, 888EB, 888GW, 8AC, 8FGR, 8SER, 8SGR Debug Modules. The **Mode** jumper has no function.

Other Debug Modules. The **Mode** jumper controls the HALT Enable mask option.

HALT	J4
Enabled	A
Disabled	B

Table 8. Mode Jumper (HALT Enable)

J5
VCC

The **VCC** Jumper allows the operating voltage of the Debug Module to be supplied from either the Debug Module (DM) or the target system (TAR). The **DM** setting supplies only 5.0V and is required for out-of-target emulation. The **TAR** setting uses the voltage supplied to the V_{CC} pin of the microcontroller socket on the target system through the Target Interface Cable (page). This voltage may range anywhere from 2.3VDC to 6.0VDC or 2.5VDC to 6.0VDC, depending on the emulation microcontroller. Attempting to operate the Debug Module outside the range specified for a particular device will cause unpredictable results.

When operating the Debug Module with a voltage source less than 4V, the maximum operating frequency must be reduced per the National Semiconductor specification for that device. Attempting to operate the Debug Module above the specified maximum frequency for that device will cause unpredictable results.

J13-J16
CF
Emulation

888CF Debug Module. The **J13-J16** jumpers are preset at the factory to **CF** for the 888CF Debug Module. These jumper settings should not be changed.

Other Debug Modules. The **J13-J16** jumpers are preset at the factory to **OTHER** for all other Debug Modules. These jumper settings should not be changed.

J17-J18
CF/CL
Emulation

888CF Debug Module. The **J17** and **J18** jumpers allow you to configure the 888CF Debug Module for either 88xCF or 88xCL operation. Both jumpers should be set to **CF** for 88xCF emulation, or to **CL** for 88xCL emulation.

Other Debug Modules. The **J17** and **J18** jumpers are preset at the factory to **OTHER**. These jumper settings should not be changed.

J21-J22
CF/CL
Emulation

888CF Debug Module. The **J21** and **J22** jumpers allow you to emulate the 88xCF and 88xCL parts with the 888CF Debug Module. Both jumpers must be set to **CF** if 88xCF emulation is desired. Both jumpers must be set to **CL** for 88xCL emulation.

J21 is used to connect a voltage to the VREF pin of the 888CF emulation microcontroller. **J22** is used to connect a ground to the AGND pin of the 888CF emulation microcontroller. These pins must be connected for the Port I/ACH functionality of the 888CF microcontroller to operate properly. As the table below shows, when J21 and J22 are in the CF position, VREF and AGND are assumed to be supplied from a target system through a Target Interface Cable.

Device to Emulate	J21	J22
88xCF (CF)	Target VREF	Target AGND
88xCL (CL)	Debug Module VCC	Debug Module GND

Table 10. CF/CL Emulation Jumpers

Other Debug Modules. The **J21** and **J22** jumpers are preset at the factory to **CF**. These jumper settings should not be changed.

J23
VPP SRC

DM3 Debug Modules. The **VPP SRC** jumper allows you to supply the V_{PP} (programming voltage) for EPROM programming from an external source, or to use the

On-board V_{PP} Generator. Refer to Chapter for more information on EPROM programming.

V_{PP} Source	J23
On-board V_{PP} Generator	G (for "G"enerator)
External (via VPP Post/Connector)	C (for "C"onnector)

Table 13. VPP SRC Jumper

DM4 and DM5 Debug Modules. There is no **VPP SRC** jumper on DM4 and DM5 Debug Modules, VPP is always generated on-board by the Debug Module. Refer to Chapter for more information on EPROM programming.

J25 Clock Mode

The **Clock Mode** jumper selects the clock mode for the emulation microcontroller. Note that other jumper blocks are involved in setting up the Debug Module to support Clock Modes. Therefore, see the Clock Modes description.

820CJ, 840CJ, 880, 888CF, 888EG, 8FGR, 8SAC, 8SGR Debug Modules

Clock Mode	J25
Crystal Oscillator	XTAL
External Oscillator	XTAL
R/C Oscillator	RC OPT

Table 11. Clock Mode Jumper

888BC, 888CG, 888EB, 888EK, 888FH, 888GD, 888GG, 888GW, 888HG, 888KG, 8AC, 8SER Debug Modules

Clock Mode	J25
Crystal Oscillator	XTAL
External Oscillator	RC OPT
R/C Oscillator	RC OPT

Table 12. Clock Mode Jumper

J26
HALT

888GW Debug Module. The **HALT** jumper is located on the 888GW daughter card next to the emulation microcontroller. It controls the HALT Enable mask option of the microcontroller.

HALT	J26
Enabled	ENABLED
Disabled	DISABLED

Table 14. HALT Jumper

Other Debug Modules. There is no **J26** jumper on other Debug Modules.

J32
Misc

DM3 Debug Modules

There is no **Misc** jumper on DM3 debug Modules.

DM4 and DM5 Debug Modules

8FGR, 8SAC, 8SER, 8SGR Debug Modules. The **Misc** jumper allows you to emulate the Watchdog option of the microcontroller.

Watchdog Option	J32
Enabled	E
Disabled	D

Table 15. Misc Jumper

Other Debug Modules. The **Misc** jumper MUST be left OPEN.

J35
A/D REF

888EB Debug Module. The **A/D REF** jumper is located on the 888EB daughter card next to the emulation microcontroller. It determines the source of the reference voltage for the A/D converter of the emulation microcontroller.

A/D Voltage Reference Source	J35
Debug Module	DM
Target System	TAR

Table 16. A/D Ref Jumper

When the DM setting is used, the Debug Module supplies the reference voltage of +5 VDC.

<u>Other Debug Modules</u>. There is no **J35** jumper on other Debug Modules.

Options By Function

Clock Modes

Crystal Oscillator Mode

In Crystal Oscillator Mode, G7 is used for CKO and is not available as a general purpose input.

Note that when emulating the CKO option, the Debug Module oscillator is driven out the target G7 pin and G7 is **not** available to take the emulation microcontroller out of HALT mode (if available).

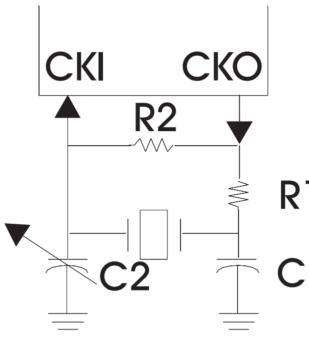


Figure 7. Crystal Oscillator Schematic

820CJ, 840CJ, 880, 888CF, 888EG, 8FGR, 8SAC, 8SGR Debug Modules

For these devices, the following table shows which jumper settings to use for the Crystal Oscillator Mode:

J1,J2,J24	J3	J25	MHz
DM	CKO[1]	XTAL	10[2]
TAR[3]	CKO[1]	XTAL	target[4]

1. Warning: The EMU jumper position MUST be left OPEN for these Debug Modules.
2. DM5 has a programmable clock (frequency set in host software).
3. TAR setting not available for DM5.
4. Target crystal can be up to 10 Mhz (N/A for DM5).

Table 17. Crystal Oscillator Mode Jumper Settings

888BC, 888CG, 888EB, 888EK, 888FH, 888GD, 888GG, 888GW, 888HG, 888KG, 8AC, 8SER Debug Modules

For these devices, the following table shows which jumper settings to use for the Crystal Oscillator Mode:

J1,J2,J24	J3	J25	MHz
DM	CKO+EMU	XTAL	10[1]
TAR[2]	CKO+EMU	XTAL	target[3]

1. DM5 has a programmable clock (frequency set in host software).
2. TAR setting not available for DM5.
3. Target crystal can be up to 10 Mhz (N/A for DM5).

Table 18. Crystal Oscillator Mode Jumper Settings

External Oscillator Mode

In External Oscillator Mode, G7 is available as a general purpose input and to take the emulation micro-controller out of HALT mode (if available).

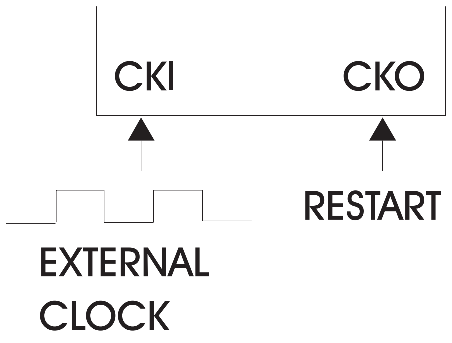


Figure 8. External Oscillator Schematic

820CJ, 840CJ, 880, 888CF, 888EG Debug Modules

For these devices, the following table shows which jumper settings to use for the External Oscillator Mode:

J1,J2,J24	J3	J25	MHz
DM	PORT[1]	XTAL	10[2]
TAR[3]	PORT[1]	XTAL	target[4]

1. Warning: The EMU jumper position MUST be left OPEN for these Debug Modules.
2. DM5 has a programmable clock (frequency set in host software).
3. TAR setting not available for DM5.
4. Target oscillator can be up to 10 Mhz (N/A for DM5).

Table 20. External Oscillator Mode Jumper Settings

888BC, 888CG, 888EB, 888EK, 888FH, 888GD, 888GG, 888GW, 888HG, 888KG, 8AC, 8SER Debug Modules

For these devices, the following table shows which jumper settings to use for the External Oscillator Mode:

J1,J2,J24	J3	J25	MHz
DM	PORT[1]	RC OPT	3.33[2]
TAR[3]	PORT[1]	RC OPT	target[4]

1. Warning: The EMU jumper position MUST be left OPEN for these Debug Modules.
2. DM5 has a programmable clock (frequency set in host software). On other Debug Modules this is a limitation of the emulation device.
3. TAR setting not available for DM5.
4. Target oscillator can be up to 3.33 Mhz (N/A for DM5).

Table 19. External Oscillator Mode Jumper Settings

8FGR, 8SAC, 8SGR Debug Modules

For these devices, the following table shows which jumper settings to use for the External Oscillator Mode:

J1,J2,J24	J3	J25	MHz
DM	PORT[1]	XTAL	5[2]
TAR[3]	PORT[1]	XTAL	target[4]

1. Warning: The EMU jumper position MUST be left OPEN for these Debug Modules.
2. DM5 has a programmable clock (frequency set in host software). On other Debug Modules this is a limitation of the emulation device.
3. TAR setting not available for DM5.
4. Target oscillator can be up to 5 Mhz (N/A for DM5).

Table 21. External Oscillator Mode Jumper Settings

R/C Oscillator Mode

In R/C Oscillator Mode, G7 is available as a general purpose input and to take the emulation microcontroller out of HALT mode (if available).

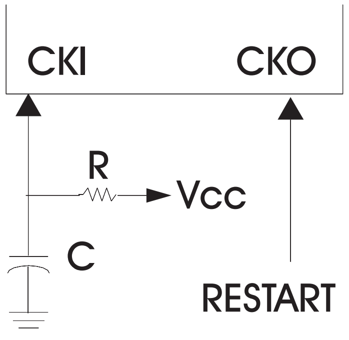


Figure 9. R/C Oscillator Schematic

Note that target R/C Oscillator Mode is not actually supported, the Debug Module will NOT oscillate with a target R/C circuit. It can be simulated, however, by driving the CKI pin (on the target connector) with a signal generator set at target CMOS level, 50% duty cycle. The maximum frequency supported depends on the emulation device in the Debug Module.

820CJ, 840CJ, 880, 888BC, 888CF, 888CG, 888EB, 888EG, 888EK, 888FH, 888GD, 888GG, 888GW, 888HG, 888KG, 8AC, 8SER Debug Modules

For these devices, the following table shows which jumper settings to use for the R/C Oscillator Mode:

J1,J2,J24	J3	J25	MHz
DM	PORT[1]	RC OPT	(1/3 * Frq)[2]
TAR[3]	PORT[1]	RC OPT	target[4]

1. Warning: The EMU jumper position MUST be left OPEN for these Debug Modules.
2. The frequency will be 1/3 of the crystal frequency, or 1/3 of the programmable clock on the DM5 (you must ensure the resulting frequency is legal).
3. TAR setting not available for DM5.
4. Can be simulated up to maximum frequency allowed for emulation device, as described above (N/A for DM5).

Table 22. R/C Oscillator Mode Jumper Settings

8FGR, 8SAC, 8SGR Debug Modules

For these devices, the following table shows which jumper settings to use for the R/C Oscillator Mode:

J1,J2,J24	J3	J25	MHz
DM	PORT[1]	RC OPT	(1/2 * Frq)[2]
TAR[3]	PORT[1]	RC OPT	target[4]

1. Warning: The EMU jumper position MUST be left OPEN for these Debug Modules.
2. The frequency will be 1/2 of the crystal frequency, or 1/2 of the programmable clock on the DM5 (you must ensure the resulting frequency is legal).
3. TAR setting not available for DM5.
4. Can be simulated up to maximum frequency allowed for emulation device, as described above (N/A for DM5).

Table 23. R/C Oscillator Mode Jumper Settings

Chapter 6: PROM Programmer

General Information

The COP8 Debug Module can program the code memory in any of the supported EPROM devices. Operation of the programmer hardware is under the control of the Host Software from the *PROM Programmer* Dialog Box.

All programming operations use the emulation code memory in the Debug Module. Code read from the device will be written into the emulation code memory in the Debug Module, and may not match the labels from a previously loaded program.

By default, areas of the emulation memory with no code will be filled with 0x00 hex bytes; however, you can use the *Inititial Code Value* control in the *Configure Emulator* Dialog Box to change the 0x00 hex byte to any other byte value.

Note that when a byte is programmed into the EPROM it is verified, therfore it is not strictly necessary to separately verify the parts after programming, although a *Verify* command is available.

When using the programmer, only one device should be in any of the ZIF sockets at one time. Do not insert a device into any of the sockets until prompted to do so by the Host Software, and always remove it before exiting the *PROM Programmer* Dialog Box or the device may be damaged.

DM3 - EPROM Programming Voltage (V_{PP}) Source

You can either supply your own external V_{PP} source or use the On-board V_{PP} Generator.

DM3 - External V_{PP} Source

In order to use an external source for V_{PP}, use the following procedure:

1) Connect the shorting block of the J23 jumper (VPP SRC) between the center post and the “C” post (“C” for Power “C”onnector).
2) Connect 13 volts to the terminal block labeled “VPP”.
3) Connect ground to either terminal labeled “GND” (of the 4 position/post terminal block).

Note that the V_{PP} voltage for all parts must be 13 volts and **it is not regulated by the Debug Module programming hardware**. The V_{PP} power source must supply at least 100mA of current at the specified voltage. Failure to supply correct and adequate power to the VPP terminal will result in erratic or possibly destructive programming.

Note that the rest of the Debug Module is powered from the “+5V” post on the power terminal block; the voltage supplied must be +5V ±5%.

DM3 - On-Board V_{PP} Generator

In order to use the On-Board V_{PP} Generator, connect the shorting block of the J23 jumper (VPP SRC) between the center post and the “G” post (“G” for On-Board “G”generator).

DM3 - On-Board V_{PP} Generator Adjustment

The On-Board V_{PP} Generator is set at the factory, no adjustment is required.

DM4, DM5 - EPROM Programming Voltage (V_{PP}) Source

The programming voltage (V_{PP}) is supplied by the On-Board V_{PP} Generator.

DM4, DM5 - External V_{PP} Source

If the On-Board V_{PP} Generator has become non-functional first check the fuse. If it is blown, replace it and try again. If it is still non-functional, you may supply the required 13 volts using the following procedure:

1) Remove the fuse.

2) Connect 13 volts to the fuse clip (on the side closest to the green PASS LED).

3) Connect ground to the middle lead (labeled GND) of the LM2577 in the TO-220 package (on the opposite side of the fuse from the 13 volt clip).

Note that the V_{PP} voltage for all parts must be 13 volts and **it is not regulated by the Debug Module programming hardware**. The V_{PP} power source must supply at least 100mA of current at the specified voltage. Failure to supply correct and adequate power to the VPP terminal will result in erratic or possibly destructive programming.

DM4, DM5 - On-Board V_{PP} Generator Adjustment

The On-Board V_{PP} Generator is set at the factory, no adjustment is required.

Chapter 7: Operational Considerations

The iceMASTER COP8 Debug Module is designed to be as close as is possible to an actual device. In most cases you will not be aware of any difference since the devices used to emulate are the actual microcontrollers. Since we need to know what is going on during emulation there are a few constraints placed upon us, and a few precautions you can take to prevent problems.

Emulation Notes

The following characteristics apply to all Debug Modules except where noted:

1) The INTR instruction is used to implement software breakpoints. There are no hardware breakpoints. When emulation starts, the program code at all breakpoints will be replaced by an INTR instruction and possibly NOPs to fill out a multi-byte instruction.

2) If your application code contains INTR instructions, they will never be executed. The Debug Module behaves as though a breakpoint occurred when emulation encounters an INTR, but will display the following message to alert you:

   Unsolicited breakpoint encountered at 0x????. (The program contains a non-breakpoint 'INTR' instruction.)

3) If a breakpoint is set on instruction which could potentially be skipped, emulation will break only when that instruction is actually executed. Emulation will never break on a skipped instruction.

   In addition, when a breakpoint is set on an instruction which could potentially be skipped and that instruction is skipped during emulation, the address and data values captured in the trace of execution will not reflect the program code at that address. Instead, the first cycle will be that of the skipped INTR instruction. If the original program code instruction was a multi-byte instruction, subsequent cycles will be the same as that of an executed NOP.

4) When a breakpoint is set on an instruction, emulation will break before that instruction executes. The two (2) bytes below the stack pointer will be overwritten when a breakpoint is reached because the INTR used for the breakpoint is actually executed.

5) Timer operation at breakpoint varies with emulator, as follows:

   880 Debug Modules. Upon reaching a breakpoint, the timer is shut off shortly after emulation stops. When emulation resumes (for a *Go* type emulation), the timer is restarted just before emulation in the program code begins.

888xx and 8Sxx Debug Modules. There is no delay in stopping or restarting running timers upon reaching a breakpoint or when emulation resumes.

6) HALT Mode. To allow clock re-synchronization in the COP8 microcontroller, it is necessary to program two NOPs immediately after the microcontroller comes out of HALT mode. When the multi-input wake-up interrupt is enabled, the first two instructions of the interrupt routine must be NOPs. If no interrupt is used, then two NOPs must follow the instruction that put the microcontroller into HALT mode.

7) IDLE Mode. Like HALT mode, it is necessary to program two NOPs to allow clock re-synchronization upon return from IDLE mode. The NOPs are place either at the beginning of the IDLE Timer interrupt routine or follow the instruction that put the microcontroller into IDLE mode.

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 blow every part you touch. The iceMASTER COP8 Debug Module can be as sensitive to static as any other circuit. The microcontroller devices in the emulators are especially vulnerable since adding extra protection would change response characteristics. This would be a step away from the real world. 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 troubles later.

Power

The Debug Module requires +5 VDC @1.0 amperes.

If you are going to be programming EPROM parts and your Debug Module does not have the on-board voltage generator see the PROM Programmer Chapter.

Clock Drivers

Many of the external clock drivers commonly used provide TTL level outputs. This can be a problem for not only the Debug Module but your target design as well. COP8 microcontrollers require a CMOS oscillator or clock levels to function reliably.

Microcontroller Serial Port

After breaking emulation, all interrupts are disabled. A serial port transmit or receive interrupt which occurs after the breakpoint has been encountered will be ignored.

IDLE and HALT Modes

During emulation, the status message

Emulation Processor Inactive

may be displayed. This status message indicates that the COP8 microcontroller is not executing instructions. Normally this can be due to an extended RESET pulse, or the microcontroller is in HALT or IDLE mode. If this condition occurs for any other reason turn to the Troubleshooting Chapter.

If the microcontroller is in HALT or IDLE mode and you execute the host-break command (by pressing either the *Stop* button or the ESC key), a confirmation box will prompt you with the following message:

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

A *Yes* response will reset the microcontroller and emulation will break at the reset address (0). A *No* or *Cancel* response will leave the microcontroller in its current state.

COP880 Single Step / Interrupt Interaction SW Fix

The host software for the COP880 Debug Module has been modified (as of version 3.6 Revision 11) to correct a problem involving interrupts while single stepping.

<u>Problem.</u> If an interrupt occurs at a breakpoint erratic behavior may result.

<u>Solution.</u> Interrupts are disabled for single steps by clearing the GIE bit just before a single step is performed. If GIE was set before the single step it will be restored (set) after the single step.

<u>Considerations.</u> This solution has two side effects.

1) If the instruction to be single stepped clears the GIE bit, and the GIE bit was set before the instruction, it will be restored (set) after the single step.

2) If the instruction to be single stepped sets the GIE bit and there is an interrupt pending, the interrupt will be lost.

Skipped Instruction / Interrupt Interaction Warning

Problem. Due to the implementation of software breakpoints on the Debug Module, if an interrupt occurs when an instruction is about to be skipped and there is a breakpoint at the instruction which will be skipped, then the INTR instruction used for the breakpoint will be skipped and the stack may be corrupt. If emulation continues out of the interrupt handler the situation will resolve itself. However, if emulation breaks while still executing the interrupt handler the stack may stay corrupted.

Solution. There are two solutions to this problem.

1) Do not put any breakpoints at instructions that can be skipped.

2) If you do have breakpoints at any instructions that can be skipped and you ever break while executing an interrupt handler, then do not remove any of those breakpoints at instructions that can be skipped. If you are not in an interrupt handler any breakpoints may be removed.

Chapter 8: Troubleshooting

Before starting any fault investigation, remove the emulator from the target system and configure it for stand-alone operation. Make sure that all connectors are in good condition and fully seated. Take note of any physical damage to the unit.

Several common problems are covered in this chapter. If this isn't enough to get the emulator back into operation, contact MetaLink.

Before Calling

Have the system near the phone so that problems may be "walked through". If the unit needs to be sent in for repair you will also need the following information:

1) the emulator's serial number (on the bottom of the emulator chassis)

2) the software revision level (see the Identification Window)

3) the address to which MetaLink will ship the repaired unit

Power

Ensure that there is +5 VDC on the correct pin of one of the target connectors on the Debug Module board. Also make sure that the output of the power supply is +5 VDC.

Communications Failure

First check the RS-232 connection. Compare the RS-232 cable with the illustration in the Hardware Description Chapter. Replace the cable if needed. A "break-out box" will facilitate a check on the presence and level of the RS-232 signals on the cable. Active signals will be at a nominal 10V and the polarity may be plus or minus depending on the hardware. If no activity is seen on Pin 2 (Transmit) the Host Computer has a fault in its interface card. If Pin 2 is active and Pin 3 is inactive (not toggling) the Debug Module has a fault.

A persistent fault may indicate that the emulation microcontroller has been damaged. For example, failure of the oscillator to start properly will result in a communication failure after, or during sequencer file loading. Therefore, make sure that the oscillator signal is present using the test point on the Debug Module at "OSC TEST PT.".

When running the oscillator from the crystal on the Debug Module, there should always be a signal. If there is no signal, the crystal or the 74HCU04 forming the oscillator may be damaged.

When running the oscillator from a target crystal, it may be necessary to adjust capacitors C1 and C2 (see figure below) in your target to ensure proper startup. See the data book for your emulation device to determine the proper capacitor values.

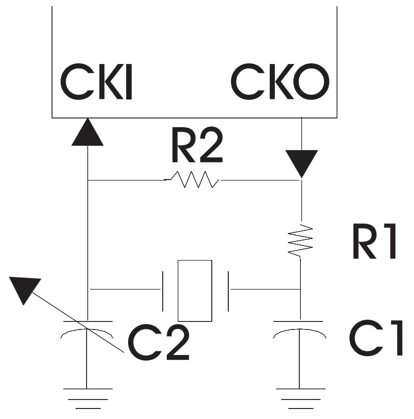


Figure 10. Crystal Oscillator Schematic

Emulation Problems

In stand-alone mode, load and run one of the demo programs that are supplied on the distribution diskettes. If the program runs correctly, the problem may not be with the emulator but with the target interface. Keep an open mind. Even in known good target systems, failures occur. It may even be possible for a "real" device to work in your target where the Debug Module has trouble. This is usually a problem with tolerances, not differences. If you encounter this, please call us so we can determine quickly where the problem lies.

Carefully consider the application:

1) verify that the Debug Module is configured properly

2) look for unexpected resets (e.g., watchdog timers)

3) check interrupt routines for proper returns to normal code execution

If these procedures restore operation in the stand-alone mode, or if the unit worked in the stand-alone mode without correction, it is necessary to determine if the target is causing the fault or if the Debug Module has a fault that doesn't manifest in stand-alone mode.

Troubles to watch for include:

1) bus contention

2) excessive loading

3) failed components in the target system (especially failures that are likely to damage the emulator)

If you have any questions contact MetaLink for technical assistance.

Chapter 9: 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[1]
Reset Counter during emulation	yes	yes	no[2]
Pass Counter during emulation	yes	no[3]	no[3]
Trace Triggers	yes	no	no
Probe Clips	yes	no[4]	no[4]
Break Button	yes	no[4]	no[4]
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[5]	yes[5]	yes[5]
Multiple Frequency Operation	yes[6]	yes[7]	no

Feature/Attribute/Command	Model 400	Debug Module	EPU
File|Store	yes	yes	yes
File|Restore	yes	yes	yes
File|Macro	no[5]	no[5]	no[5]
Run|From	yes	yes	yes
Run|Until	yes	yes	yes
Run|Slow Motion|Instruction	yes	yes	no
Run|Slow Motion|Line	yes	yes	no
Run|Step|Line	yes	yes	yes
Run|Step|Over	yes	yes	yes
Run|Step|To	yes	yes	yes
Trace|Search	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 24. Model 400, Debug Module, EPU Differences