OSBDM User's Design & Troubleshooting Guide

June 2011

How to Contact Us

Corporate Headquarters	Freescale Semiconductor, Inc. 6501 William Cannon Drive West Austin, Texas 78735 U.S.A. P&E Microcomputer Systems, Inc, 98 Galen St., 2nd Floor Watertown, MA 02472-4502
World Wide Web	`http://www.freescale.com/codewarrior` `http://www.pemicro.com/osbdm`

1 Introduction to Open Source BDM

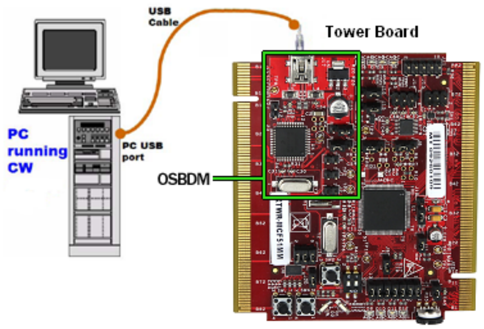
This document describes an Open Source programming and debugging development tool designed to work with Freescale HCS08, RS08, ColdFire V1,V2, V3 and V4, Kinetis ARM, HC(S)12(X), DSC, and Qorivva MPC55xx/56xx microcontrollers.

This chapter contains the following sections:

1.1 About the Open Source BDM Interface

The Open Source BDM design includes both software and embedded hardware components which allow connectivity between an application on the PC and a Freescale microcontroller resident on an evaluation board, development board, or Freescale Tower card. The Open Source BDM is often referred to as OSBDM or OSJTAG depending upon the target processor it is designed to work with. In this manual, the names "Open Source BDM", "Open Source Jtag", "OSBDM" and "OSJTAG" are often used interchangeably. There is no support for Open Source BDM from Freescale or P&E Microcomputer Systems; the Open Source BDM is provided with all the required source code, hardware schematics, and libusb drivers for both hardware and software components. Because it is open source, the source code can be used and/or modified from its original design at the user's own risk. Figure 1.1 provides a pictorial overview of the typical connections required for programming and debugging using the Open Source BDM interface. A PC connects to the Open Source BDM via the mini-USB port on the tower board. In this example, the ColdFireV1 MM256 demonstration tower board is being programmed and debugged.

Figure 1.1 Debugging with OSBDM Interface



The Open Source BDM (OSBDM) hardware design can be embedded into a development board which will provide two major capabilities: (1) it provides power to the board from the USB bus, and (2) it allows the board to be debugged and programmed from the PC with only a USB connection. There is a separate schematic available for each of the different architectures supported by the OSBDM (or OSJTAG) design.

The Open Source BDM software library on the PC (often referred to as OSDLL) is a dynamically linked library which provides very basic run control and debug capabilities to an application on the PC. Freescale's CodeWarrior, P&E Microcomputer Systems Software, and various third-party software use the OSDLL software library to provide debug and programming capabilities for the processor resident in the development board.

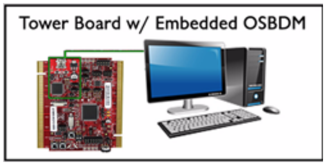
Many commercial tools and customer test applications use the P&E commercial UNIT library connection to support the OSBDM design as it has more capabilities than the open source library (OSDLL) and also supports the USB Multilink and Cyclone Pro/Max commercial hardware interfaces.

The current version of the Open Source BDM design supports the HCS08, RS08, ColdFire V1, ColdFire V2/V3/V4, Kinetis ARM, HC(S)12(X), DSC, and Qorivva MPC55xx/56xx architectures. Visit http://www.pemicro.com/osbdm for the latest OSBDM design iterations.

1.2 OSBDM In The Product Cycle

OSBDM is ideal for the beginning of the product development cycle. Its cost-effective nature makes it well-suited to planning, analysis, and device evaluation. Many of Freescale's evaluation boards come with the Open Source BDM design built-in, which requires just a USB connection to the PC. The main trade-off in using the Open Source BDM is the performance and feature set limitations.

Figure 1.2 Evaluation Phase: OSBDM Tower

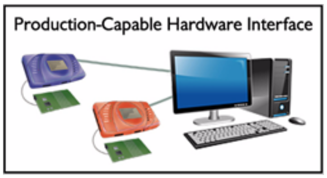


During the actual design phase, most companies will produce their own application-specific hardware. This hardware does not contain the OSBDM circuitry and instead will include a very inexpensive debug connector. This debug connector will allow a hardware debug interface, such as P&E's Multilink Universal, to connect to and debug/program the board. This interface has a much higher performance and more robust feature set which is ideal for the development phase.

Figure 1.3 Development Phase: Debug Interface

For production/manufacturing, particularly at high volumes, a PC-based or stand-alone automated programmer, such as P&E's Cyclone PRO and Cyclone MAX, is recommended.

Figure 1.4 Production Phase: Programming Hardware



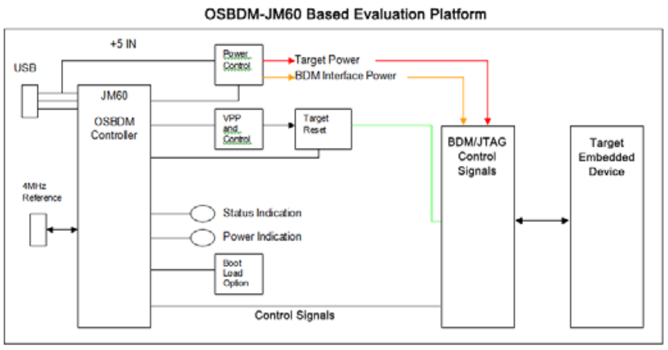
1.3 OSBDM Microcontroller

Open Source BDM (OSBDM) is based on the MC9S08JM60 microcontroller with embedded support for USB 2.0 full-speed communications. All communication and control for BDM/JTAG operations are performed by the JM60. The USB port provides the host communication and power source for the JM60 and BDM internal circuits. Peripheral devices and JM60 I/O ports are used to control communication to the target device. The JM60 internal flash memory contains all operating firmware for the BDM application. The OSBDM firmware is designed to support communication to the target device via the OS DLL as well as CodeWarrior, P&E and other third party PC software. The latest hardware interface drivers from P&E include support for OSBDM USB interfaces. To download the OSBDM design/driver package, please go to www.pemicro.com/osbdm.

1.4 Hardware Block Diagram

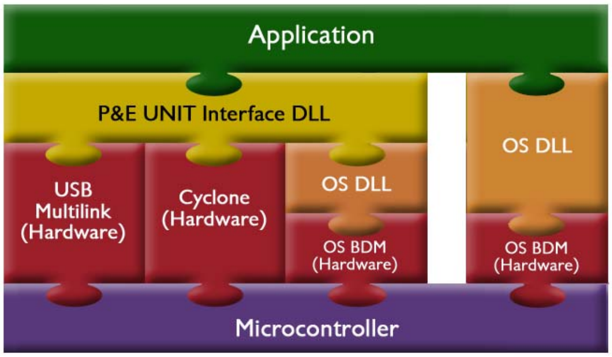
The Open Source BDM (OSBDM) hardware design can be embedded into a development board which will provide two major capabilities; (1) it provides power to the board from the USB bus; and (2) it allow the board to be debugged and programmed from the PC with only a USB connection. There is a separate schematic available for each of the different architectures supported by the OSBDM (or OSJTAG) design.

Figure 1.5 Hardware Block Diagram



1.5 PC Software Block Diagram

Figure 1.6 OSBDM Layers



The figure above shows how an application on the PC has two different ways to control and debug a Freescale processor.

One option is to use the Open Source BDM library (OSDLL) directly to communicate to the OSBDM hardware. Since OSBDM is ideal for the evaluation stage of production, this path has the disadvantage of being stuck in evaluation because it can communicate only to the OSBDM hardware. The advantage of this route is that all components necessary are open source components. This approach is included in the open source archive with the exception of support for the Qorivva which is currently only available through P&E's UNIT library.

The other option is to go through the P&E UNIT Interface DLL which is designed to simplify the user’s communication to OSBDM and have multiple hardware options. The advantages of using the P&E UNIT Interface DLL are extra features such as breakpoints and file load to RAM, and the ability to communicate with OSBDM and P&E’s hardware like the USB Multilink Universal and the Cyclone Pro/Max. This path is ideal for users who intend to move from evaluation and into development and production stages. This option is used in Freescale’s CodeWarrior and many other tool vendor applications. P&E's UNIT library is available to tool vendors at no cost.

With the current release of the OSBDM firmware, the Qorivva MPC55xx/56xx firmware doesn't have a full complement of debug calls to allow the OSDLL to easily debug a target. It does provide an ability to shift JTAG data to the target microcontroller. This support will be added to the OSBDM for future releases. P&E's UNIT library interface does currently support the OSBDM with standard debug calls (run, step, go, load, etc.) and can be used to easily integrate OSBDM support for the MPC55xx/MPC56xx devices.

1.6 OSBDM-JM60 Features

The open source BDM is implemented with the following features:

- USB 2.0 full speed
- Provides power to the rest of the development board. Power to target is limited to USB requirements and cannot exceed 0.5A of combined OSBDM and target circuit current.
- Different schematics and firmware provided to support different Freescale architectures, including the HCS08, RS08, HC(S)12(X), DSC, and ColdFire V1, V2, V3, and V4 targets, as well as Kinetis ARM, and Qorivva MPC55xx/56xx targets
- Optional high voltage circuitry allows +12Vpp generation for RS08 targets
- Enumeration Status and Target Power Indicators
- Boot load option to update firmware using USB connection. Many commercial debuggers will use this interface (often via P&E's UNIT library) to update the firmware of the OSBDM. There is also a firmware update utility available from P&E. Refer to Installation and Operation of the OSBDM.
- Low cost design, easy to embed on target boards
- Optional USB virtual serial port application to pass target serial data to host Program Counter (PC) with USB connection

1.7 Open Source BDM Package

This section describes the content of the Open Source BDM package. The package is distributed in a .zip file and includes development folders.

Table 1.1 Open Source BDM Components

Component	Description/Interfaces	Available/Comments
Windows USB drivers	LibUSB drivers used to connect to the Open Source BDM debug port. These files are required when the PC detects the Open Source BDM hardware when plugged into the PC as a new USB device.	The distributed files do not include the virtual serial port drivers. An installer which installs all the OSBDM drivers, including debug and virtual serial drivers, as well as P&E Multilink and Cyclone drivers can be downloaded and installed from http://www.pemicro.com/osbdm.
JM60 USB/BDM firmware	Firmware running on the JM60 that receives commands via USB from the PC and converts them into commands as defined by the BDM or JTAG communications protocol. These commands are serial outputted - “bit - banged” - by the JM60 using port pins to drive the communication pins on the user’s target. Required CodeWarrior 6.3.	This file is provided as an S-record and needs to be programmed into JM60 on-chip flash. Firmware update files for different device architectures are also provided with OSBDM design package 28 and higher. These files can be used with the automatic firmware update capability that is a part of CodeWarrior 10.1 or higher, as well as an OSBDM firmware update utility from P&E Microcomputer Systems: www.pemicro.com/osbdm

Table 1.1 Open Source BDM Components

Component	**Description/Interfaces**	**Available/Comments**
Open Source BDM Schematic and BOMs	PCB: OpenSourceBDM Revision B schematics for each individual OSBDM per each device architecture. This hardware contains the JM60 and circuitry, clock, and power to provide the interface to the program or debug target	All schematic files for the Open Source BDM PCB are provided along with a Bill Of Materials (BOM).
Open Source BDM Library (OSDLL)	The Open Source BDM library (OSDLL) is a PC-based dynamically linked library which uses libusb to connect to the Open Source BDM hardware.	Full source for this library is included. The library source is in the form of a Microsoft Visual Studio 2008 project and is located in the osbdm-sw\osbdm-pc\osbdmusb\src directory of the source distribution archive. A commercial library is also available from P&E Microcomputer Systems which additionally supports USB Multilink and Cyclone Pro/Max hardware as well.

1.8 Support and Licensing

Open Source BDM is not supported by Freescale or P&E Microcomputer Systems, Inc.; it is open source. Any bugs, enhancements, or support questions should be addressed through the Open Source BDM forum or Freescale’s website. Open Source BDM has been thoroughly tested, but there is no guarantee of error-free operation. All the source files are available to anyone under the GNU LESSER GENERAL PUBLIC LICENSE. For more information on licensing, refer to the *COPYING_LGPL.txt* file located at the root directory of the OSBDM-JM60 package. An OSBDM design package can be downloaded from www.pemicro.com/osbdm.

1.9 History

- TBDML - Turbo BDM Light — A low cost, open source debugging interface compatible with the CodeWarrior environment for the HCS12 family, versions 5.0 and lower. It is based on the MC908JB08 MCU and updated to the MC908JB16 device. Versions of CodeWarrior for HC12 5.1 and higher only support OSBDM JM60 designs. Support for TBDML hardware interfaces has also been deprecated in CodeWarrior 10.x software products.
- TBLCF - Turbo BDM Light ColdFire — Provides an interface for the open source BDM project, and also to the ColdFire V2, V3, and V4 microprocessors and microcontrollers. It is based on the MC908JB16 MCU. This feature is no longer supported under CodeWarrior 10.x versions.
- OSBDM-JM60 — It supports the HCS08, RS08, and ColdFire V1, V2, V3 and V4, HC(S)12(X), Kinetis ARM, and Qorivva MPC55xx/56xx Freescale architectures.

2 Installation and Operation of the OSBDM

This chapter contains the following sections:

2.1 Installing Windows OSBDM and USB Drivers

The OSBDM circuitry uses two different USB drivers on the PC, one for debug and the other for virtual serial. The debug driver is open source and is based upon the libUSB. The virtual serial driver is based upon the Jungo USB driver and is not currently open source. One of the long term goals of the OSBDM design is to move the virtual serial driver either to the libUSB driver or to use the custom CDC communications class to implement a serial port.

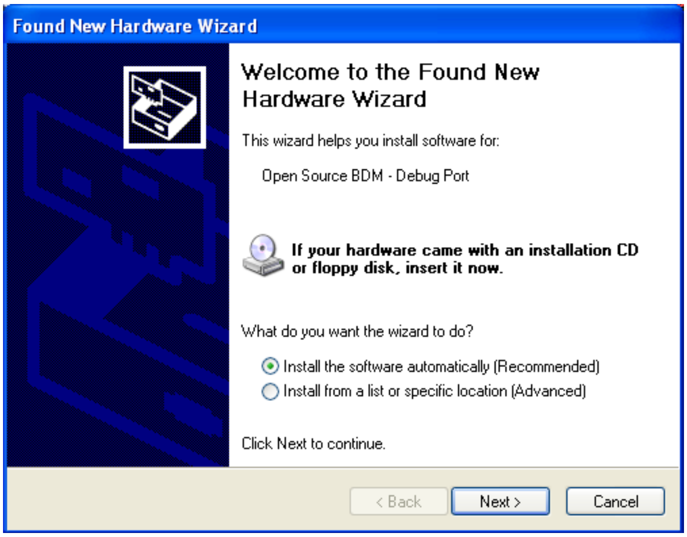
The OSBDM source repository includes a layout of files which allow the OSBDM debug interface to be installed on a Windows 32 or 64 bit machine.

P&E provides an installer which automatically installs OSBDM debug and virtual serial drivers on both Windows 32 and 64 bit machines. This may be downloaded from http://www.pemicro.com/osbdm. In the root install directory, the OSDLL can be found encase existing software packages need to be updated with this DLL.

The following procedure specifies how to install the Open Source BDM USB hardware drivers on the Windows operating system using the P&E installer. For this procedure, it is assumed that the Open Source BDM Windows USB driver package is already unpacked and being installed onto the development PC via the standalone installation of P&E hardware device drivers or when drivers are automatically installed as part of CodeWarrior 10.1 or a more recent layout installation.

When OSBDM is connected to the PC for the first time, the Windows operating system will recognize a new composite USB device. Each OSBDM interface running firmware version 29.00 or greater will enumerate two USB devices. The OSBDM core device will enumerate as the primary USB device. At the same time, the virtual serial port will also be enumerated as a secondary USB device Figure 2.1 illustrates that the initial connection starts the Windows driver installation and displays the Windows New Hardware Wizard dialog box. Both OSBDM and virtual serial port will enumerate as two different USB devices - therefore, each will have to be installed separately.

Figure 2.1 Found New Hardware Wizard


Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

This wizard helps you install software for:

Open Source BDM - Debug Port

If your hardware came with an installation CD or floppy disk, insert it now.

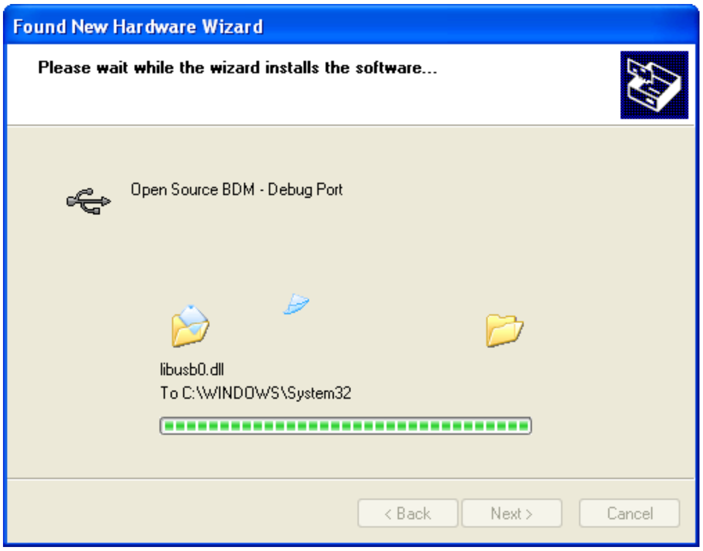
What do you want the wizard to do?

(●) Install the software automatically (Recommended)
( ) Install from a list or specific location (Advanced)

Click Next to continue.

1. For this installation, select the **Install the software automatically** option and click **Next**.
2. This will initiate the installation of the Open Source BDM USB driver and DLL file, as shown in Figure 2.2.

Figure 2.2 Driver Installation



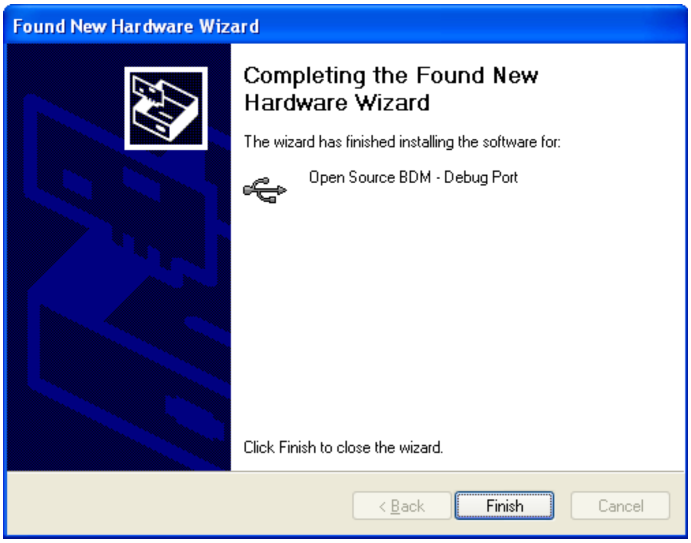
Please wait while the wizard installs the software...

Open Source BDM - Debug Port

libusb0.dll
To C:\WINDOWS\System32

3. Once the installation procedure is completed, as shown in Figure 2.3, click **Finish**. Because of the plug-and-play nature of USB, a reboot of Windows is not required.

Figure 2.3 Finish Installation Open Source BDM Windows USB Drive



Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

Open Source BDM - Debug Port

Click Finish to close the wizard.

4. Once the OSBDM Debug port is installed, repeat steps 1-3 when installing the Virtual Serial port. Once the OSBDM Debug port and Virtual Serial port are installed correctly, the OSBDM device will be ready to use.

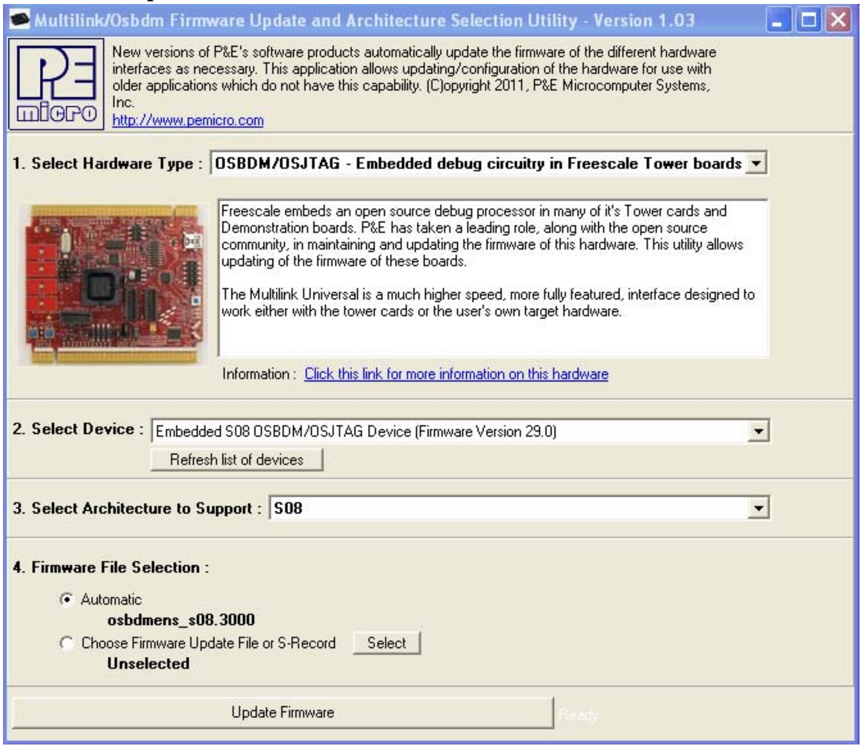
2.2 Firmware Update Utility

For development tools or custom applications which use the P&E UNIT library interface (which relies on OSDLL), the firmware of the OSBDM will automatically be updated if it is old. This requires the user's intervention to boot the device into bootloader mode.

For development tools or custom applications which use the OSDLL directly, the firmware is not automatically updated (although the OSDLL does have some provisions for firmware updating). In this case, P&E provides a firmware update utility which may be downloaded from http://www.pemicro.com/osbdm. This utility can be used to update the firmware automatically to the latest version (the default), or the user may specify a firmware update file or S19 to program into the OSBDM 9S08JM60 processor. The following directions demonstrate a typical update scenario using this utility.

1. Run PEFirmwareUpdater.exe (See Figure 2.4)
2. From the drop box labeled “1. Select Hardware Type”, select “OSBDM/OSJTAG – Embedded debug circuitry in Freescale Tower boards”.
3. From the drop box labeled “2. Select Device”, select the board that needs the firmware updated. If the device is not showing up, use the “Refresh list of devices” to repopulate the device list.
4. From the drop labeled “3. Select Architecture to Support”, select the OSBDM board design type. This field may be auto populated after selecting the device in step 3.
5. From the field “4. Firmware file selection”, indicate the firmware version by either having the utility automatically select the firmware, or by manually selecting the firmware by clicking on the “Select” button. The version number is labeled without the decimal period. Example: Selecting *osbdmens_s08.3007* means the OSBDM firmware version 30.07 for the HCS08 design type is selected.
6. Click on the “Update Firmware” button and follow the instructions to update the firmware.

Figure 2.4 PEFirmwareUpdater



Multilink/Osbdm Firmware Update and Architecture Selection Utility - Version 1.03

New versions of P&E's software products automatically update the firmware of the different hardware interfaces as necessary. This application allows updating/configuration of the hardware for use with older applications which do not have this capability. (C)opyright 2011, P&E Microcomputer Systems, Inc.
http://www.pemicro.com

1. Select Hardware Type : OSBDM/OSJTAG - Embedded debug circuitry in Freescale Tower boards

Freescale embeds an open source debug processor in many of it's Tower cards and Demonstration boards. P&E has taken a leading role, along with the open source community, in maintaining and updating the firmware of this hardware. This utility allows updating of the firmware of these boards.

The Multilink Universal is a much higher speed, more fully featured, interface designed to work either with the tower cards or the user's own target hardware.

Information : Click this link for more information on this hardware

2. Select Device : Embedded S08 OSBDM/OSJTAG Device (Firmware Version 29.0)

3. Select Architecture to Support : S08

4. Firmware File Selection :

(●) Automatic
osbdmens_s08.3000
( ) Choose Firmware Update File or S-Record
Unselected

A problem may occur where the user is making use of an application (Such as CodeWarrior 6.3) which relies on an older version of OSDLL (named either osbdm-jm60.dll or peosbdmv1.dll). The latest version of OSDLL is backwards API compatible with previous version. In the root directory of the firmware Update Utility install is both latest copies of OSDLL, which are identical to one another (osbdm-jm60.dll/ peosbdmv1.dll).

If this occurs, both the Device manager in Windows and P&E's Firmware Information Utility will properly display the hardware, but older development software will not recognize it. In this case, the user should update the copies of the DLL in those distributions, as well as the [Windows\System32] folder ([Windows\SysWow64] on 64-bit systems).

2.3 Firmware Recovery Utility

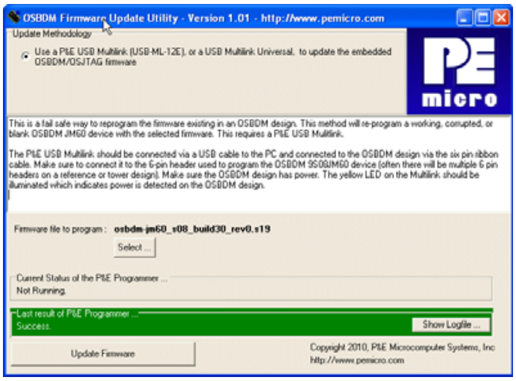
The firmware update utility described in the previous section requires that the OSBDM hardware be booted into bootloader mode. If for any reason the firmware is corrupt to the point where it cannot boot into bootloader mode, P&E provides a Firmware Recovery Utility. This utility requires either a USB Multilink Universal or a USB-ML-12 Multilink in order to reprogram the OSBDM circuitry via the 6-pin OSBDM debug connector. This is a fail-safe way to reprogram the firmware in an existing OSBDM design. This method will re-program a working, corrupted, or blank OSBDM 9S08JM60 device with the selected firmware.

The P&E USB Multilink (USB-ML-12E) should be connected via a USB cable to the PC and connected to the OSBDM design via the 6-pin ribbon cable. Make sure to connect it to the 6-pin header used to program the OSBDM 9S08JM60 device (often there will be multiple 6-pin headers on a reference or tower design). Make sure the OSBDM design has power. The yellow LED on the Multilink should be illuminated which indicates power is detected on the OSBDM design.

This utility may be downloaded from http://www.pemicro.com/osbdm. The following are typical steps to getting the firmware recovery utility to work:

1. Run the P&E Firmware Recovery Utility
2. Click the "Select" button and choose the firmware to be programming into the unit. Default firmware for each architecture configuration is included with the utility.
3. Click the "Update Firmware" button. A window should pop up that displays the status of the programming process, and when complete the Firmware Recovery Utility should indicate a positive result.

Figure 2.5 Firmware Recovery Utility


OSBDM Firmware Update Utility - Version 1.01 - http://www.pemicro.com
Update Methodology
Select ...
Show Logfile ...
Update Firmware

Use a P&E USB Multilink (USB-ML-12E), or a USB Multilink Universal, to update the embedded OSBDM/OSJTAG firmware

This is a fail safe way to reprogram the firmware existing in an OSBDM design. This method will re-program a working, corrupted, or blank OSBDM JM60 device with the selected firmware. This requires a P&E USB Multilink.

The P&E USB Multilink should be connected via a USB cable to the PC and connected to the OSBDM design via the six pin ribbon cable. Make sure to connect it to the 6-pin header used to program the OSBDM 9S08JM60 device (often there will be multiple 6 pin headers on a reference or tower design). Make sure the OSBDM design has power. The yellow LED on the Multilink should be illuminated which indicates power is detected on the OSBDM design.

Firmware file to program : **osbdm-jm60_s08_build30_rev0.s19**

Current Status of the P&E Programmer ...
Not Running.

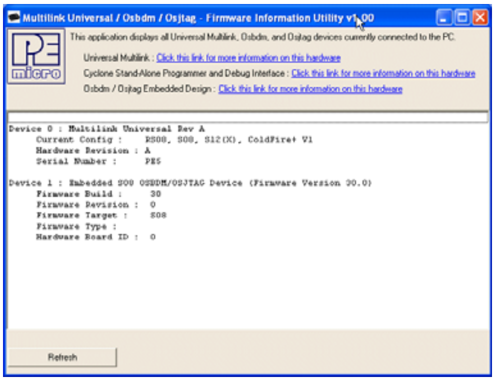
Last result of P&E Programmer ...
Success.

Copyright 2010, P&E Microcomputer Systems, Inc
http://www.pemicro.com

2.4 Firmware Information Utility

This utility will query the USB bus for any attached OSBDM and USB Multilink devices. It will then display information about these devices, such as configuration and version number information. This is very useful to understand which USB devices are properly configured (driver-wise) and what their current state is. An image of the utility is shown here:

Figure 2.6 Firmware Information Utility


Multilink Universal / Osbdm / Osjtag - Firmware Information Utility v1.00
Refresh

This application displays all Universal Multilink, Osbdm, and Osjtag devices currently connected to the PC.

Universal Multilink : Click this link for more information on this hardware

Cyclone Stand-Alone Programmer and Debug Interface : Click this link for more information on this hardware

Osbdm / Osjtag Embedded Design : Click this link for more information on this hardware

```
Device 0 : Multilink Universal Rev A
      Current Config :     RS08, S08, S12(X), ColdFire+ V1
      Hardware Revision : A
      Serial Number :      PE5

Device 1 : Embedded S08 OSBDM/OSJTAG Device (Firmware Version 30.0)
      Firmware Build :      30
      Firmware Revision :   0
      Firmware Target :     S08
      Firmware Type :
      Hardware Board ID :   0
```

2.5 Troubleshooting Debug Driver

If you are having trouble connecting to an OSBDM-based device, the first step is to verify in the Windows Device Manager that the OSBDM device has enumerated properly. For example, on Windows XP the user would click the START button, select Control Panel, double click System, select the Hardware Tab, and click the Device Manager button.

Under the LibUSB-Win32 Devices, the OSBDM device should be shown. If it is not, the user should follow these steps:

1. Install the latest drivers as per Installing Windows OSBDM and USB Drivers.
2. Unplug and Replug the OSBDM device. If it is shown with a yellow exclamation mark, select it, right mouse click, and select re-install driver.
3. See if the device is shown properly in the Windows Device Manager
4. If the device still does not show up in the Device Manager, place the OSBDM device into bootloader mode and run the Firmware Update Utility as shown in Firmware Update Utility. Place the device backing into normal mode (remove IRQ jumper).
5. Unplug and Replug the OSBDM device. If it is shown with a yellow exclamation mark, select it, right mouse click, and select re-install driver.

2.6 Troubleshooting Virtual Serial Driver

Before attempting to fix any virtual serial drivers, make sure that the device debug function shows up properly as detailed in Troubleshooting Debug Driver. Once this is complete, refer to Figure 2.7 for an example of what the drivers should look like for both the debug and virtual serial functions.

If the Jungo tab does not show up, reinstall the drivers as detailed in Installing Windows OSBDM and USB Drivers. If the PEMICRO USB Serial Port does not show up under the Jungo tab, select the OSBDM device listed in the LibUSB-Win32 Devices list, right-click, and choose reinstall driver.

If the driver still does not show up, then most likely the OSBDM-based devleopment board has older firmware. The firmware should be updated to 30.07 or newer.

In firmware version 30.07, the USB device ID (PID) was changed so that Windows would always recognize both interfaces of the USB device. A corresponding change was made to the OSDLL to recognize both the older and new PID devices. The firmware is available in the P&E Firmware Update Utility. Run the Firmware Update Utility as detailed in Firmware Update Utility to update the firmware to version 30.07. An older version of the firmware, from before the PID change, may be found in the Archive subfolder of the firmware update utility.

After the OSBDM Drivers are installed properly on the PC and/or after the target firmware has been successfully updated, the device manager will properly enumerate the virtual serial and the utilities will be able to use it. Refer to Firmware Update Utility for instructions on how to update the target's firmware.

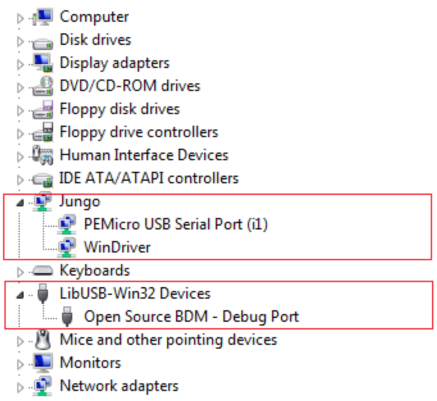
The device manager should show the following devices (see Figure 2.7):

Jungo -> PEMicro USB Serial Port (i1)

Jungo -> WinDriver

LibUSB-Win32 Devices -> Open Source BDM – Debug Port

Figure 2.7 Device Manager - Virtual Serial and Debug Ports Enumerated



```
▷ Computer
▷ Disk drives
▷ Display adapters
▷ DVD/CD-ROM drives
▷ Floppy disk drives
▷ Floppy drive controllers
▷ Human Interface Devices
▷ IDE ATA/ATAPI controllers
◢ Jungo
    PEMicro USB Serial Port (i1)
    WinDriver
▷ Keyboards
◢ LibUSB-Win32 Devices
    Open Source BDM - Debug Port
▷ Mice and other pointing devices
▷ Monitors
▷ Network adapters
```

A problem may occur where the user is making use of an application (Such as CodeWarrior 6.3) which relies on an older version of OSDLL (named either osbdmjm60.dll or peosbdmv1.dll). The latest version of OSDLL is backwards API compatible with previous version. In the root directory of the firmware Update Utility install is both latest copies of OSDLL, which are identical to one another (osbdm-jm60.dll/peosbdmv1.dll).

If this occurs, both the Device manager in Windows and P&E's Firmware Information Utility will properly display the hardware, but older development software will not recognize it. In this case, the user should update the copies of the DLL in those distributions, as well as the [Windows\System32] folder ([Windows\SysWow64] on 64-bit systems).

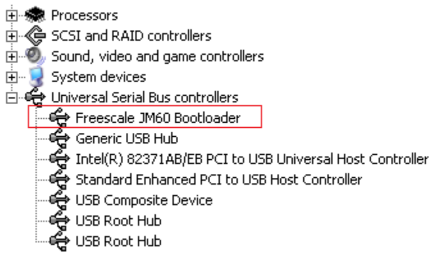
2.7 Troubleshooting Bootloader Drivers

Bootloader mode is a firmware mode that an OSBDM design can be placed in to allow it to be updated. In newer software applications, such as CodeWarrior 10.1 or P&E's firmware update utility, you may be prompter to boot the device into bootloader mode in order to update the firmware. Entering bootloader mode is as simple as unplugging the USB cable from the OSBDM design, placing a two pin jumper on the bootloader header (connecting IRQ to GND), and plugging the USB cable back in. If all goes well, the device will boot properly into bootloader mode.

If however, the update software doesn't recognize the USB device, there may be a driver issue on the PC. This is generally due to older utilities and drivers being installed on the PC. The simplest way to diagnose this is to open the Windows Device Manager. If the device appears under the "libUSB-Win32 devices" tab as "Open Source BDM - Bootloader Port", then all is well with the drivers. This is shown in Figure 2.14.

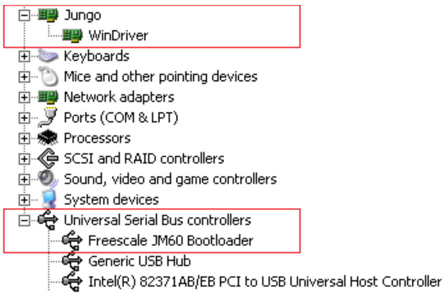
If there is a previous driver installation from Freescale JM60 Demo Applications and the JM60 board is enumerated in bootloader, the device may be using older drivers and the device manager will look as shown in Figure 2.8 below.

Figure 2.8 Bootloader drivers after Freescale JM60 Demo Application


Processors
SCSI and RAID controllers
Sound, video and game controllers
System devices
Universal Serial Bus controllers
Freescale JM60 Bootloader
Generic USB Hub
Intel(R) 82371AB/EB PCI to USB Universal Host Controller
Standard Enhanced PCI to USB Host Controller
USB Composite Device
USB Root Hub
USB Root Hub

The first step is to install the drivers install on P&E's website at http://www.pemicro.com/osbdm to make sure the latest system drivers are installed on the machine. After installing the drivers and re-plugging the board in bootloader mode, the user will notice it is still using the Freescale JM60 Bootloader drivers (Figure 2.8), which makes it unrecognizable by applications (See Figure 2.9 and).

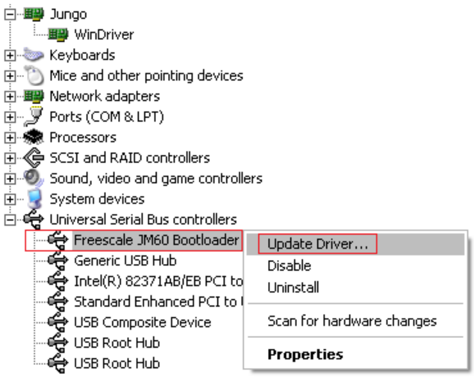
Figure 2.9 OSBDM Bootloader drives NOT installed


Jungo
WinDriver
Keyboards
Mice and other pointing devices
Network adapters
Ports (COM & LPT)
Processors
SCSI and RAID controllers
Sound, video and game controllers
System devices
Universal Serial Bus controllers
Freescale JM60 Bootloader
Generic USB Hub
Intel(R) 82371AB/EB PCI to USB Universal Host Controller

In order to get the OSBDM Booloader to enumerate properly and be able to use the Firmware Update utility, the user will need to update the drivers by following the steps below:

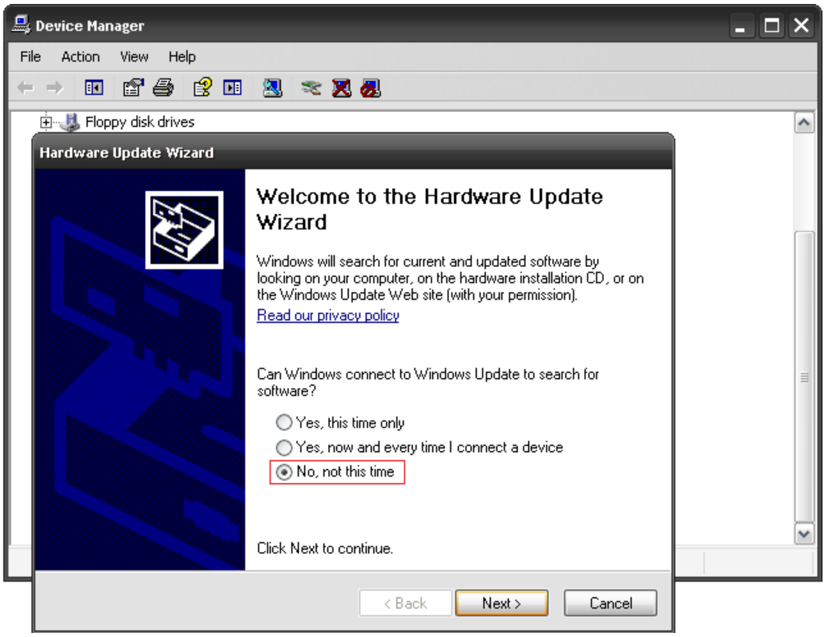
1. On the device manager, right click on Freescale JM60 Bootloader and select "Update Driver..."

Figure 2.10 Step 1 - Select “Update Driver”



2. When prompted if Windows can connect to Windows Update to search for software, select "No, not this time” option and press Next.

Figure 2.11 Step 2 - Skip Software Search



Welcome to the Hardware Update Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

Read our privacy policy

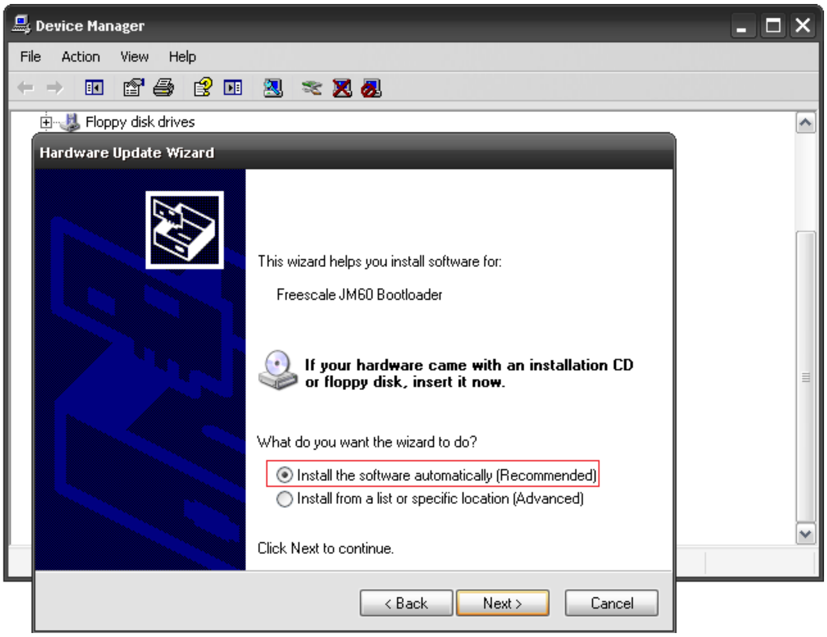
Can Windows connect to Windows Update to search for software?

- Yes, this time only
- Yes, now and every time I connect a device
- [✓ No, not this time]

Click Next to continue.

3. When asked "What do you want to the wizard to do?", select Install software automatically and press Next.

Figure 2.12 Step 3 - Select “Install Software Automatically”



This wizard helps you install software for:

Freescale JM60 Bootloader

If your hardware came with an installation CD or floppy disk, insert it now.

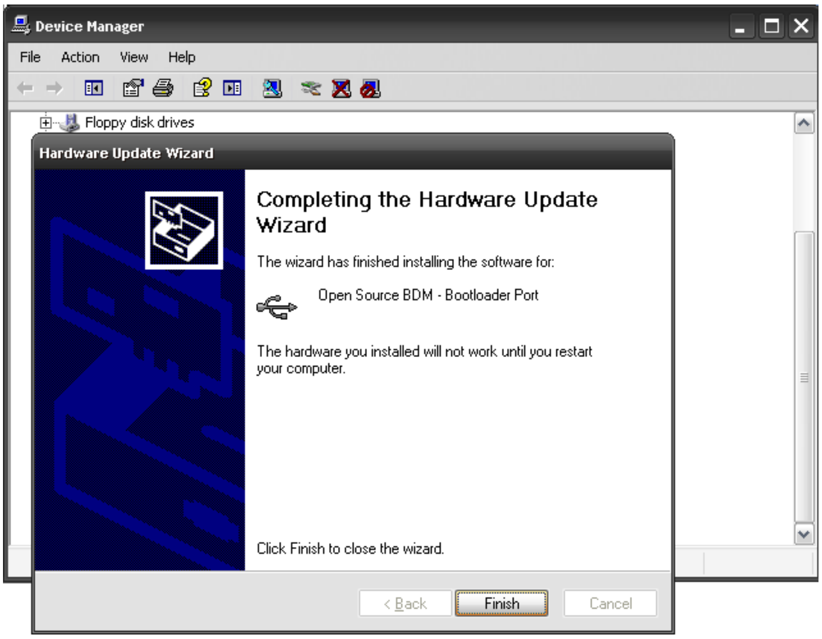
What do you want the wizard to do?

[✓ Install the software automatically (Recommended)]
Install from a list or specific location (Advanced)

Click Next to continue.

4. Complete the installation and the OSBDM Bootloader drivers will be successfully updated to work with P&E applications.

Figure 2.13 Step 4 - Complete Installation



Completing the Hardware Update Wizard

The wizard has finished installing the software for:

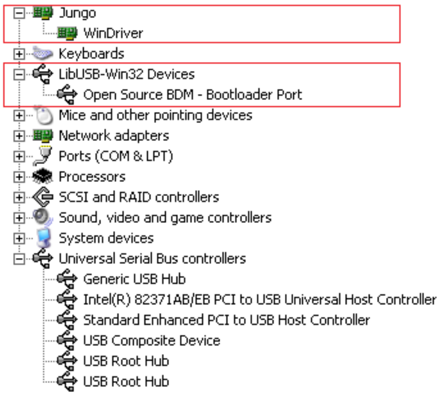
Open Source BDM - Bootloader Port

The hardware you installed will not work until you restart your computer.

Click Finish to close the wizard.

5. The device manager should now enumerate properly

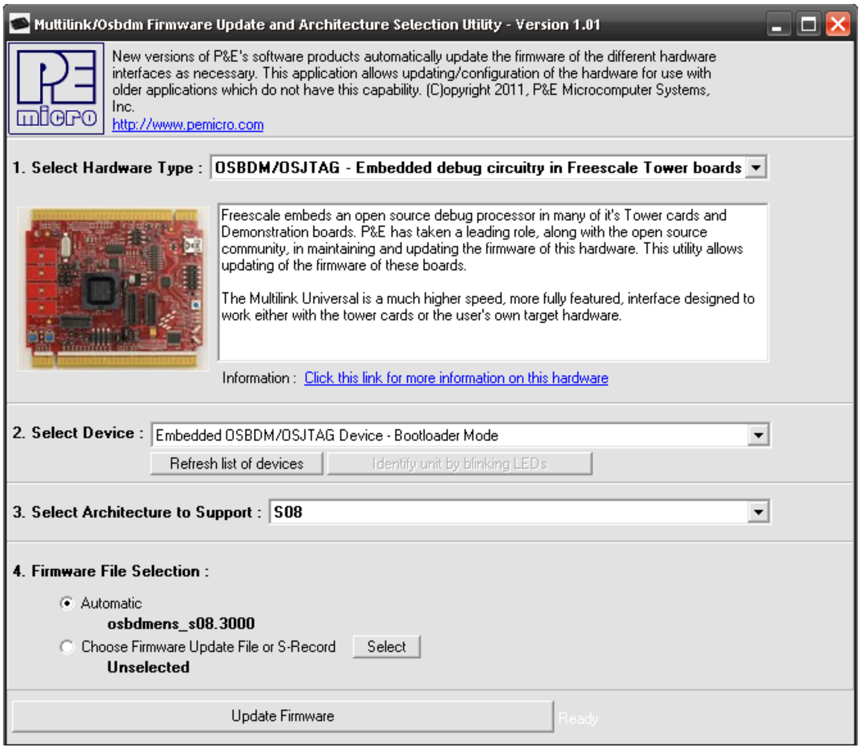
Figure 2.14 Properly Enumerated Device Manager



```
Jungo
    WinDriver
Keyboards
LibUSB-Win32 Devices
    Open Source BDM - Bootloader Port
Mice and other pointing devices
Network adapters
Ports (COM & LPT)
Processors
SCSI and RAID controllers
Sound, video and game controllers
System devices
Universal Serial Bus controllers
    Generic USB Hub
    Intel(R) 82371AB/EB PCI to USB Universal Host Controller
    Standard Enhanced PCI to USB Host Controller
    USB Composite Device
    USB Root Hub
    USB Root Hub
```

6. The firmware updater utility (or the update mechanism in many development tools) will now recognize the OSBDM board in bootloader mode.

Figure 2.15 OSBDM Boatloader Mode Recognized In Firmware Updater



Multilink/Osbdm Firmware Update and Architecture Selection Utility - Version 1.01

New versions of P&E's software products automatically update the firmware of the different hardware interfaces as necessary. This application allows updating/configuration of the hardware for use with older applications which do not have this capability. (C)opyright 2011, P&E Microcomputer Systems, Inc.
http://www.pemicro.com

1. Select Hardware Type : OSBDM/OSJTAG - Embedded debug circuitry in Freescale Tower boards

Freescale embeds an open source debug processor in many of it's Tower cards and Demonstration boards. P&E has taken a leading role, along with the open source community, in maintaining and updating the firmware of this hardware. This utility allows updating of the firmware of these boards.

The Multilink Universal is a much higher speed, more fully featured, interface designed to work either with the tower cards or the user's own target hardware.

Information : Click this link for more information on this hardware

2. Select Device : Embedded OSBDM/OSJTAG Device - Bootloader Mode

3. Select Architecture to Support : S08

4. Firmware File Selection :

(●) Automatic
osbdmens_s08.3000
( ) Choose Firmware Update File or S-Record
Unselected

3 OSBDM Firmware

The OSBDM-JM60 firmware application supports various target types, such as: HCS08, RS08, ColdFire V1, V2, V3 and V4, HC(S)12(X), DSC, Kinetis ARM and Qorivva MPC55xx/56xx.

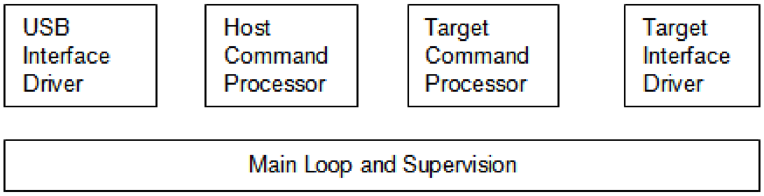
This chapter contains the following sections:

- Firmware Block Diagram
- Composite Device
- Firmware Source Table

3.1 Firmware Block Diagram

The OSBDM firmware source code has different conditional defines to determine which architecture it will support for a specific binary build. The firmware only will support a single architecture at a time. The firmware must be loaded on hardware which supports that specific firmware type. There are different schematics depending upon the hardware architecture chosen. For example, the OSBDM firmware that supports the CFV234 device architecture is designed to run on CFV234 the OSBDM reference design but it cannot be used on OSBDM design hardware for ARM Kinetis.

Figure 3.1 Firmware Block Diagram



A common USB interface driver and host command processor are applied for all target types. Target command processor and target interface driver are selected to support the specific target device. Common utility source files are also included.

3.2 Composite Device

Previous to firmware version 27, the OSBDM design was a single function device. The OSBDM supported only a debug function. With version 27 and beyond, the device became what is known as a composite device. This type of device registers with the operating system as two separate logical devices (implemented in firmware as two interfaces of a single device). So when enumerating on the PC, the device will show up as an OSBDM debug function and an OSBDM virtual serial function. USB Endpoints 1 and 2 correspond to the debug function and Endpoints 3 and 4 correspond to the virtual serial function.

3.3 Firmware Source Table

Table 3.1 describes the association of various targets with the source files.

Table 3.1 Description of Source Files

Firmware Source File	Target Association	Function Summary
Start08.c	JM60	Reset Vector and start-up
main.c	All Targets	Application start, main loop and management
cmd_processing.c	All Targets	Host command processor
USB_User_API.c	All Targets	USB driver I/O, USB driver is interrupt driven
MCU.c	JM60	JM60 initialization and hardware support
bdm_9S08.c	HCS08	HCS08 target specific command processor
bdm_9RS08.c	RS08	RS08 target specific command processor
bdm_9S12.c	HC(S)12(X)	HC(S)12(X) target specific command processor
bdm_cfv1.c	ColdFire V1	ColdFire V1/Flexis target specific command processor

Table 3.1 Description of Source Files

Firmware Source File	Target Association	Function Summary
bdm_bgnd_driver.c	S08	Common BGND type interface driver for higher speed targets (Edge capture type, 25Mhz maximum target BDM clock)
bdm_cf.c	ColdFire V2/3/4	ColdFire V2/3/4 Target Specific Command Processor and Interface Driver
jtag_dsc.c	DSC	DSC Target Specific Command Processor and Interface Driver
jtag_kinetis.c	ARM	ARM Target Specific Command Processor and Interface Driver
jtag_eppc.c	Qorivva MPC55xx/56xx	Qorivva Target Specific Command Processor and Interface Driver
timer.c	All Targets	Timer and timing support
util.c	All Targets	Utility functions and support
serial_io.c	All Targets	Target Serial channel/ COM port support

NOTE `Main.c` determines if the JM60 USB bootloader application or primary application starts. Each source file listed has one or more associated header files (.h extension) to provide the necessary definitions. Select the Header tab of the CodeWarrior project to edit these files.

4 OSBDM PC Software Library (OSDLL)

This chapter describes the Open Source BDM solution PC software components.

This chapter contains the following sections:

- Overview
- OSBDM-JM60.DLL and LIBUSB

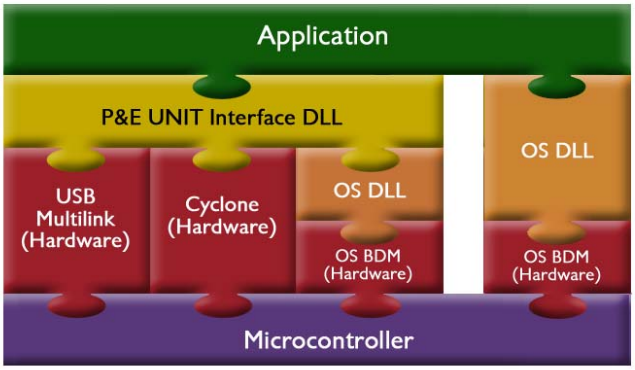
4.1 Overview

This section describes the Open Source BDM solution PC software components. These components include:

- Open Source BDM interface DLL
- USB driver (libusb.lib)

Figure 4.1 illustrates these software components and their interfaces.

Figure 4.1 OSBDM Windows PC Software and Interface



The figure above shows how an application on the PC has two different ways to control and debug a Freescale processor.

One option is to use the Open Source BDM library (OSDLL) directly to communicate to the OSBDM hardware. Since OSBDM is ideal for the evaluation stage of production, this path has the disadvantage of being stuck in evaluation because it can communicate only to the OSBDM hardware. The advantage of this route is that all components necessary are open source components. This approach is included in the open source archive with the exception of support for the Qorivva which is currently only available through P&E's UNIT library.

The other option is to go through the P&E UNIT Interface DLL which is designed to simplify the user's communication to OSBDM and have multiple hardware options. The advantages of using the P&E UNIT Interface DLL are extra features such as breakpoints and file load to RAM, and the ability to communicate with OSBDM and P&E's hardware like the USB Multilink Universal and the Cyclone Pro/Max. This path is ideal for users who intend to move from evaluation and into development and production stages. This option is used in Freescale's CodeWarrior and many other tool vendor applications. P&E's UNIT library is available to tool vendors at no cost.

With the current release of the OSBDM firmware, the Qorivva MPC55xx/56xx firmware doesn't have a full complement of debug calls to allow the OSDLL to easily debug a target. It does provide an ability to shift JTAG data to the target microcontroller. This support will be added to the OSBDM for future releases. P&E's UNIT library interface does currently support the OSBDM with standard debug calls (run, step, go, load, etc.) and can be used to easily integrate OSBDM support for the MPC55xx/MPC56xx devices.

NOTE P&E UNIT Interface DLL and USB Multilink/Cyclone PRO/Cyclone MAX hardware interfaces are commercial products. They are designed and distributed by P&E Microcomputer Systems (www.pemicro.com). The P&E UNIT Interface DLL is a free library for third-party tool vendors.

NOTE For all Freescale architectures, except for Qorivva MPC55xx/56xx, OSBDM can be used in standalone with the OSDLL without relying on the P&E UNIT Interface Libraries. Future OSBDM releases will natively support Qorivva as well.

4.2 OSBDM-JM60.DLL and LIBUSB

The OSBDM-JM60.DLL provides an interface between CodeWarrior, P&E Unit Interface .dlls or other possible third party software products, and the Open Source BDM firmware. This section describes the API of the Open Source BDM DLL including a Windows Open

Source USB drivers library, LIBUSB. The source code for the Open Source BDM DLL is available.

4.2.1 OSBDM USB Base Driver API

Software applications interface to the OSBDM-JM60 firmware using the `osbdm_dll.h`. The header files *`commands.h`* and *`osbdm_dll.h`* contain all the definition used by these APIs that should be included in your project.

The Open Source BDM DLL functions are listed and briefly described below:

unsigned char osbdm_dll_version(void)

Returns the version number of the `osbdm_dll.h` in BCD format (major in upper nibble and minor in lower nibble).

OsbdmErrT osbdm_get_version(unsigned char *version_info)

Returns the hardware (MSB) and firmware (LSB) version of the open OSBDM-JM60 device.

unsigned char osbdm_init()

Initializes a connection to the USB driver and returns the number of OSBDM-JM60 devices found. If there is a problem communicating with the USB driver or other error then -1 is returned.

This function must be called before a device can be opened.

OsbdmErrT osbdm_open(unsigned char device_no)

Opens communication with an OSBDM-JM60 specified by `device_no`. The first device is number 0.

Returns 0 on success and non-zero on failure. This function must be called before any further communication with the device can take place.

void osbdm_close()

Closes communication with currently opened device.

OsbdmErrT osbdm_target_reset(ResetT resetmode)

Resets the MCU target using the `resetmode` value. Returns 0 on success and non-zero on failure.

OsbdmErrT osbdm_target_go()

Starts target code execution from current PC address. Returns 0 on success and non-zero on failure.

OsbdmErrT osbdm_target_step()

Steps over a single target instruction. Returns 0 on success and non-zero on failure.

OsbdmErrT osbdm_target_halt(void)

Brings the target into active background (debug) mode with user code execution halted. Returns 0 on success and non-zero on failure.

OsbdmErrT osbdm_get_speed(unsigned long *speed)

Returns the crystal (or external source) frequency of the target in KHz.

OsbdmErrT osbdm_set_speed(unsigned long speed)

Sets the BDM communication speed which is the crystal or external source frequency in KHz. Returns `osbdm_error_ok` on success or the error is defined in `OsbdmErrT`.

OsbdmErrT osbdm_init_hardware(void)

Initialize the OSBDM-JM60 BDM hardware and establish a connection between the OSBDM-JM60 probe. Returns `osbdm_error_ok` on success or the error is defined in `OsbdmErrT`.

OsbdmErrT osbdm_status(unsigned char *data)

Retrieve the target BDM communication status information to `data`. Returns `osbdm_error_ok` on success or the error is defined in `OsbdmErrT`.

OsbdmErrT osbdm_config(unsigned char config_type, unsigned char config_param, unsigned long param_value)

Configure the OSBDM-JM60 and target. The configuration type `config_type`, defined in `ConfigT` selects the entity to be configured. The configuration parameter `config_param`, is set to the value specified in `param_value`. Returns `osbdm_error_ok` on success or the error is defined in `OsbdmErrT`.

OsbdmErrT osbdm_write_fill(unsigned char type, unsigned char width, unsigned long addr, unsigned char *data, unsigned long count)

Fill a single value to a contiguous section of target's memory with an amount of `count`. The value to fill is written to `data` and `addr` is starting address in memory, `type` specifies the memory type for fill operation, which is defined in `OsbdmMemT` and `width` is the number of bits for the memory access size. Returns `osbdm_error_ok` on success or the error is defined in `OsbdmErrT`.

unsigned char osbdm_write_block(unsigned char type, unsigned char width, unsigned long addr, unsigned char *data, unsigned long size)

Writes a block of memory to the target.

unsigned char osbdm_read_block(unsigned char type, unsigned char width, unsigned long addr, unsigned char *data, unsigned long size)

Reads a block of memory from the target. Both `osbdm_read_block` and `osbdm_write_block` share the same parameters. Some of these parameters are also used by other memory functions.

Table 4.1 describes the parameters of read and write memory functions if applicable to specific device architectures.

NOTE For Qorivva MPC55xx/56xx devices, the only way to use OSBDM outside of CodeWarrior is through P&E's Unit Interface Libraries. The Libraries are required to handle complex JTAG communication which often differs between devices within the Qorivva family. The Libraries are not available as open source. Please contact support at http://www.pemicro.com/support/index.cfm with a request for a unit_ppc dll product release.

Table 4.1 Description of Parameters

Parameters	**Description**
type	Type of memory to be read or written: MEM_RAM - Normal memory MEM_REG - Normal register MEM_CREG - Control register MEM_DREG - Debug register MEM_P - Special Program Memory MEM_P_FLASH - Special Program Flash Memory MEM_X - Special Data Memory MEM_X_FLASH - Special Data Flash Memory
width	Number of bits to be read or written at a time (8, 16 or 32)
addr	Start address
data	Pointer to the data to be written or to hold the data read
size	Number of 8-bit data bytes to be written or read

4.2.2 OSBDM Bootloader API

OsbdmErrT osbdm_bootloader_open_device(unsigned long device_no);

Opens communication with an OSBDM-JM60 in bootloader mode specified by device_no. The first device is number 0.

Returns 0 on success and non-zero on failure. This function must be called before any further communication with the device can take place in bootloader mode. Use the standard osbdm_close() to close communication with currently opened device.

OsbdmErrT osbdm_bootloader_erase_flash_block(unsigned long start_address, unsigned long end_address);

Erase a single block of flash memory of the target in bootloader mode.

OsbdmErrT osbdm_bootloader_program_flash_block(unsigned long data_address, unsigned long data_length, char *data_array);

Program a block of flash memory of the target in bootloader mode. The maximum length of the block is 8 bytes per API call.

OsbdmErrT osbdm_bootloader_verify_flash_block(unsigned long data_address, unsigned long data_length, char *data_array)

Verify a block of flash memory of the target in bootloader mode. The maximum length is 8-bit data bytes per API call.

4.2.3 OSBDM USB Driver API

OsbdmErrT osbdmAPI_connect(CoreT core_type)

Connect to the device specified by `core_type` and collect the BDM communication status information. Returns `osbdm_error_ok` on success or the error defined in `OsbdmErrT`.

OsbdmErrT osbdmAPI_get_status(ConnectStateT *status)

Get the target BDM communication status information, which contains the target reset state, the target connection state, the OSBDM-JM60 version, and the flash state. The `ConnectStateT` is defined in a common header file, `osbdm_def.h`. Returns `osbdm_error_ok` on success or the error defined in `OsbdmErrT`.

OsbdmErrT osbdmAPI_run(void)

Start the execution of a core. Returns `osbdm_error_ok` on success or the error defined in `OsbdmErrT`.

OsbdmErrT osbdmAPI_step(void)

Step over a single target instruction. Returns `osbdm_error_ok` on success or the error defined in `OsbdmErrT`.

OsbdmErrT osbdmAPI_read_mem(unsigned char mem_space, unsigned int addr, unsigned int byte_count, SizeT access_size, unsigned char *buffer)

Read `byte_count` of data from the memory. The data is read into buffer and `addr` is the starting address in memory to begin read. Memory access attribute can be specified with `access_size` whose type is defined in `SizeT`. The memory space `mem_space`, is an optional target dependent parameter and it is defined in the target specific header. Returns `osbdm_error_ok` on success or the error defined in `OsbdmErrT`.

OsbdmErrT osbdmAPI_write_mem(unsigned char mem_space, unsigned int addr, unsigned int byte_count, SizeT access_size, unsigned char *buffer)

Write `byte_count` of data to the memory. The data is written to buffer and address is the starting address in memory to begin write. Memory access attribute can be specified with `access_size` whose type is defined in `SizeT`. The memory space `mem_space`, is an optional target dependent parameter and it is defined in the target specific header. Returns `osbdm_error_ok` on success or the error defined in `OsbdmErrT`.

OsbdmErrT osbdmAPI_config(ConfigT config_type, unsigned char config_param, unsigned long param_value)

Configure the OSBDM-JM60 and target. The configuration type `config_type`, defined in `ConfigT` selects the entity to be configured. The configuration parameter `config_param`, is set to the value specified in `param_value`. Returns `osbdm_error_ok` on success or the error defined in `OsbdmErrT`.

OsbdmErrT osbdmAPI_core_mode(CoreModeT *core_mode)

Poll the execution mode of the core and returns the mode defined in `CoreModeT`. Returns `osbdm_error_ok` on success or the error defined in `OsbdmErrT`.

OsbdmErrT osbdmAPI_secure_mode(SecureModeT *secure_mode)

Poll the target secure mode and return the mode defined in `SecureModeT`. Returns `osbdm_error_ok` on success or the error defined in `OsbdmErrT`.

4.2.4 Target Specific Header Files

osbdm_cfv1.h

The header file contains Coldfire V1 target specific definitions that are used in the OSBDM-JM60 API. Some ColdFire V1 internal control signals are defined in this header file.

osbdm_cfv234.h

The header file contains Coldfire V2/3/4 target specific definitions that are used in the OSBDM-JM60 API. Some ColdFire V2/3/4 internal control signals are defined in this header file.

osbdm_s08.h

The header file contains HCS08 target specific definitions that are used in the OSBDM-JM60 driver API. Some HCS08 internal control signals are defined in this header file.

jtag_dsc.h

The header file contains DSC target specific definitions that are used in the OSBDM-JM60 driver API. Some DSC internal control signals are defined in this header file.

jtag_eppc.h

The header file contains Qorivva MPC55xx/56xx target specific definitions that are used in the OSBDM-JM60 driver API. Some Qorivva MPC55xx/56xx internal control signals are defined in this header file.

jtag_kinetis.h

The header file contains ARM Kinetis target specific definitions that are used in the OSBDM-JM60 driver API. Some ARM Kinetis internal control signals are defined in this header file.

osbdm_s12.h

The header file contains HCS(S)12(X) target specific definitions that are used in the OSBDM-JM60 driver API. Some HCS(S)12(X) internal control signals are defined in this header file.

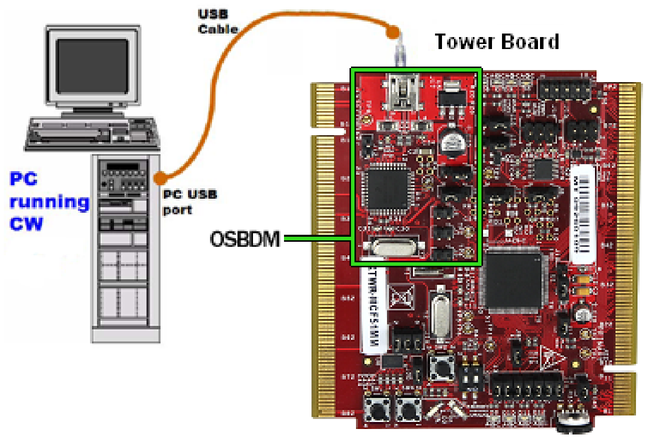
5 OSBDM Embedded Hardware

This chapter explains about the OSBDM-JM60 hardware and its connections. It contains the following sections:

5.1 OSBDM Options and Connections

The OSBDM design for each Freescale architecture has a set of communication and power signals that should be connected to the corresponding signals on the embedded evaluation board. The design also provides jumper options for feature and target settings, such as the bootloader mode enable jumper, for consequent firmware update. Jumper shunts are applied to enable a feature or option when installed on 2 pins. The option is disabled when the jumper shunt is placed on one pin or it is removed. In some cases, shunt is placed between two pins of a 3 pin header i.e. between pins 1 and 2 or between pins 2 and 3. This allows selection between two different functions.

Figure 5.1 Option and Connector Diagram

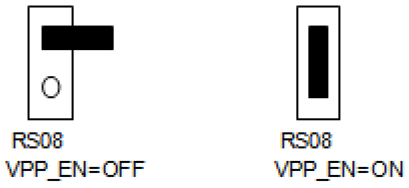


5.1.1 Options

5.1.1.1RS08 VPP Enable

RS08 VPP Enable provides Flash programming voltage (VPP) support for RS08 type targets. When enabled, the OSBDM will provide the VPP voltage to the RESET signal under the application control.

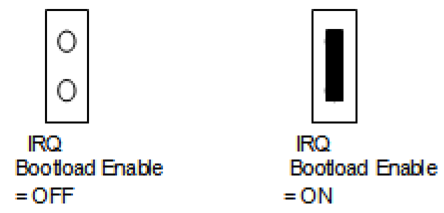
Figure 5.2 RS08 VPP Enable Options



5.1.1.2IRQ/Bootload Mode Enable

IRQ/Bootload Mode Enable provides an option to enable the JM60 USB bootloader application. This application should reside in the JM60 flash firmware. Bootloader application is included in the OSBDM projects by default. The jumper must be set prior to connecting the OSBDM USB connector to the host PC. USB drivers must be installed on the host PC for proper operation. OSBDM firmware can be updated via an automatic firmware update mechanism in CW10.1 or higher, or by using P&E's OSBDM firmware update utility. Refer to the Installation and Operation of the OSBDM section for instructions on how to use the utility.

Figure 5.3 IRQ/Bootload Mode Enable Options



5.1.2 HCS08/RS08/HC(S)12(X)/CFV1 OSBDM Power and Control Signals

As a part of including the OSBDM design in your evaluation board, a set of outlined power and control signals has to be connected to the corresponding signals of your embedded on-board device. Below is a list of these signals for devices that belong to

HCS08/RS08/HC(S)12(X)/CFV1 Freescale families. A full schematic of this OSBDM design can be downloaded from www.pemicro.com/OSBDM.

Table 5.1 HCS08/RS08/HC(S)12(X)/CFV1 Power and Control Signals

Signal	Description
5V_TRG	This 5V regulated output signal provides a power output connection to the rest of the evaluation board. The power for this rail comes from the USB Bus and is controlled by the Reference Design. After the Reference Design enumerates on the PC's USB Bus, this voltage output is enabled. The evaluation board should draw no more than 250mA of current from this connection. The evaluation board should provide sufficient storage and bypass capacitance on this line.
3V_TRG	This 3.3V regulated output signal provides a power output connection to the rest of the evaluation board. The power for this rail comes from the USB Bus and is controlled by the Reference Design. After the Reference Design enumerates on the PC's USB Bus, this voltage output is enabled. The evaluation board should draw no more than 250mA of current from this connection. The evaluation board should provide sufficient storage and bypass capacitance on this line. The 3V_TRG voltage source and the 3V on-board regulator are both optional.
V_TRG	This input signal should be connected to the regulated voltage at which target Freescale device is running. OSBDM in turn uses this voltage source to power communication circuitry and to ensure that OSBDM is talking to on-board device at the correct voltage level.

Table 5.1 HCS08/RS08/HC(S)12(X)/CFV1 Power and Control Signals

Signal	Description
TXD_VIRTUAL_TGT	This line should be connected to the TXD signal of the SCI module on the target microprocessor. Through this line OSBDM will be receiving traffic from the serial module of the target MCU. This connection will be used to accommodate sending Serial Data to the Virtual Serial Port through the Reference Design. It will also allow the user to use the P&E toolset to capture the traffic coming from the serial port of the user's target microprocessor.
RXD_VIRTUAL_TGT	This line should be connected to the RXD signal of the SCI module on the target microprocessor. Through this line OSBDM will be transmitting traffic to the serial module of the target MCU. This connection will be used to accommodate sending Serial Data from the Virtual Serial Port through the Reference Design. It will also allow the user to use the P&E toolset to capture the traffic coming from the serial port of the user's target microprocessor.
TBGND	This bidirectional signal should be connected to the evaluation board processor's BGND pin. This signal allows the reference design to communicate with the evaluation board processor via the background debug module.
TGT_RST	This bi-directional signal should be connected to the RESET line of the microprocessor. This signal allows the reference design to reset the evaluation board processor and also detect resets of the evaluation board processor.

Table 5.1 HCS08/RS08/HC(S)12(X)/CFV1 Power and Control Signals

Signal	**Description**
GND	This is the common ground signal. It should be connected to the ground plane of the target microprocessor board. This ground is connected in the Reference Design to the PC ground via the USB port.
ELE_PS_SENSE	The ELE_PS_SENSE is an optional input signal that allows for better integration of the OSBDM design within Freescale's Tower Cards and Elevator evaluation framework. Please refer to ELE_PS_SENSE Signal for more details.

5.1.3 CFV234 OSBDM Power and Control Signals

As a part of including an OSBDM design in your evaluation board, a set of outlined power and control signals has to be connected to the corresponding signals of your embedded on-board device. Below is a list of these signals for devices that belong to CFV234 Freescale families. A full schematic of this OSBDM design can be downloaded from www.pemicro.com/OSBDM.

Table 5.2 CFV2/3/4 OSBDM Power and Control Signals

Signal	**Description**
TXD_VIRTUAL_TGT	This line should be connected to the TXD signal of the SCI module on the target microprocessor. Through this line OSBDM will be receiving traffic from the serial module of the target MCU. This connection will be used to accommodate sending Serial Data to the Virtual Serial Port through the Reference Design. It will also allow the user to use the P&E toolset to capture the traffic coming from the serial port of the user’s target microprocessor.
RXD_VIRTUAL_TGT	This line should be connected to the RXD signal of the SCI module on the target microprocessor. Through this line OSBDM will be transmitting traffic to the serial module of the target MCU. This connection will be used to accommodate sending Serial Data from the Virtual Serial Port through the Reference Design. It will also allow the user to use the P&E toolset to capture the traffic coming from the serial port of the user’s target microprocessor.
CF_BKPT	This output signal should be connected to the evaluation board processor’s BKPT pin. BKPT is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.
CF_DCLK	This output signal should be connected to the evaluation board processor’s DCLK pin. DCLK is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.

Table 5.2 CFV2/3/4 OSBDM Power and Control Signals

Signal	Description
CF_TCLK	This bi-directional signal should be connected to the evaluation board processor’s TCLK pin. TCLK is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.
5V_TRG	This 5V regulated output signal provides a power output connection to the rest of the evaluation board. The power for this rail comes from the USB Bus and is controlled by the Reference Design. After the Reference Design enumerates on the PC’s USB Bus, this voltage output is enabled. The evaluation board should draw no more than 250mA of current from this connection. The evaluation board should provide sufficient storage and bypass capacitance on this line.
3V_TRG	This 3.3V regulated output signal provides a power output connection to the rest of the evaluation board. The power for this rail comes from the USB Bus and is controlled by the Reference Design. After the Reference Design enumerates on the PC’s USB Bus, this voltage output is enabled. The evaluation board should draw no more than 250mA of current from this connection. The evaluation board should provide sufficient storage and bypass capacitance on this line. The 3V_TRG voltage source and the 3V on-board regulator are both optional.
V_TRG	This input signal should be connected to the regulated voltage at which target Freescale device is running. OSBDM in turn uses this voltage source to power communication circuitry and to ensure that OSBDM is talking to on-board device at the correct voltage level.

Table 5.2 CFV2/3/4 OSBDM Power and Control Signals

Signal	Description
GND	This is the common ground signal. It should be connected to the ground plane of the target microprocessor board. This ground is connected in the Reference Design to the PC ground via the USB port.
CF_DSI	This output signal should be connected to the evaluation board processor's DSI pin. DSI is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.
CF_DSO	This input signal should be connected to the evaluation board processor's DSO pin. DSO is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.
CF_PST0	This input signal should be connected to the evaluation board processor's PST0 pin. PST0 is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module. NOTE: If the ColdFire processor has the ALLPST signal, then there is the option to connect all of the CF_PST0-3 signals to the evaluation board processor's ALLPST pin.
CF_PST1	This input signal should be connected to the evaluation board processor's PST1 pin. PST1 is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module. NOTE: If the ColdFire processor has the ALLPST signal, then there is the option to connect all of the CF_PST0-3 signals to the evaluation board processor's ALLPST pin.

Table 5.2 CFV2/3/4 OSBDM Power and Control Signals

Signal	Description
CF_PST2	This input signal should be connected to the evaluation board processor's PST2 pin. PST2 is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module. NOTE: If the ColdFire processor has the ALLPST signal, then there is the option to connect all of the CF_PST0-3 signals to the evaluation board processor's ALLPST pin.
CF_PST3	This input signal should be connected to the evaluation board processor's PST3 pin. PST3 is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module. NOTE: If the ColdFire processor has the ALLPST signal, then there is the option to connect all of the CF_PST0-3 signals to the evaluation board processor's ALLPST pin.
TGT_RST	This bi-directional signal should be connected to the RESET line of the microprocessor. This signal allows the reference design to reset the evaluation board processor and also detect resets of the evaluation board processor.

Table 5.2 CFV2/3/4 OSBDM Power and Control Signals

Signal	Description
CF_TA	This output signal should be connected to the evaluation board processor's TEA or TA pin. TA is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module. If the ColdFire processor that the user is using has the TEA signal, always choose it over the TA signal. If the processor does not possess either TEA or TA signal, please leave this line unconnected.
ELE_PS_SENSE	The ELE_PS_SENSE is an optional input signal that allows for better integration of the OSBDM design within Freescale's Tower Cards and Elevator evaluation framework. Please refer to ELE_PS_SENSE Signal for more details.

5.1.4 Qorivva MPC55xx/56xx & DSC Power and Control Signals

As a part of including an OSBDM design in your evaluation board, a set of outlined power and control signals has to be connected to the corresponding signals of your embedded on-board device. Below is a list of these signals for devices that belong to Qorivva MPC55xx/56xx Freescale families. A full schematic of this OSBDM design can be downloaded from www.pemicro.com/OSBDM.

Table 5.3 Qorivva MPC55xx/56xx & DSC Power and Control Signals

Signal	Description
GND	This is the common ground signal. It should be connected to the ground place of the target microprocessor board. This ground is connected in the Reference Design to the PC ground via the USB port.
5V_TRG	This 5V regulated output signal provides a power output connection to the rest of the evaluation board. The power for this rail comes from the USB Bus and is controlled by the Reference Design. After the Reference Design enumerates on the PC’s USB Bus, this voltage output is enabled. The evaluation board should draw no more than 250mA of current from this connection. The evaluation board should provide sufficient storage and bypass capacitance on this line.
3V_TRG	This 3.3V regulated output signal provides a power output connection to the rest of the evaluation board. The power for this rail comes from the USB Bus and is controlled by the Reference Design. After the Reference Design enumerates on the PC’s USB Bus, this voltage output is enabled. The evaluation board should draw no more than 250mA of current from this connection. The evaluation board should provide sufficient storage and bypass capacitance on this line. The 3V_TRG voltage source and the 3V on-board regulator are both optional.
V_TRG	This input signal should be connected to the regulated voltage at which target Freescale device is running. OSBDM in turn uses this voltage source to power communication circuitry and to ensure that OSBDM is talking to on-board device at the correct voltage level.

Table 5.3 Qorivva MPC55xx/56xx & DSC Power and Control Signals

Signal	**Description**
TXD_VIRTUAL_TGT	This line should be connected to the TXD signal of the SCI module on the target microprocessor. Through this line OSBDM will be receiving traffic from the serial module of the target MCU. This connection will be used to accommodate sending Serial Data to the Virtual Serial Port through the Reference Design. It will also allow the user to use the P&E toolset to capture the traffic coming from the serial port of the user's target microprocessor.
RXD_VIRTUAL_TGT	This line should be connected to the RXD signal of the SCI module on the target microprocessor. Through this line OSBDM will be transmitting traffic to the serial module of the target MCU. This connection will be used to accommodate sending Serial Data from the Virtual Serial Port through the Reference Design. It will also allow the user to use the P&E toolset to capture the traffic coming from the serial port of the user's target microprocessor.
TDO	This input signal should be connected to the evaluation board processor's TDO pin. TDO is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.
TMS	This input signal should be connected to the evaluation board processor's TMS pin. TMS is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.
TDI	This output signal should be connected to the evaluation board processor's TDI pin. TDI is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.

Table 5.3 Qorivva MPC55xx/56xx & DSC Power and Control Signals

Signal	Description
TCK	This bi-directional signal should be connected to the evaluation board processor's TCK pin. TCK is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.
RESET_B	This bi-directional signal should be connected to the RESET line of the microprocessor. This signal allows the reference design to reset the evaluation board processor and also detect resets of the evaluation board processor.
RDY	This input signal should be connected to the RDY line of the microprocessor. This signal is used to indicate to the development tools that data is ready to be read from or write to the Qorivva's read/ write access registers.
JCOMP	This output signal should be connected to the JCOMP line of the microprocessor. This signal is used to enable the JTAG controller.
ELE_PS_SENSE	The ELE_PS_SENSE is an optional input signal that allows for better integration of the OSBDM design within Freescale's Tower Cards and Elevator evaluation framework. Please refer to ELE_PS_SENSE Signal for more details.

5.1.5 Kinetis ARM Power and Control Signals

As a part of including an OSBDM design in your evaluation board, a set of outlined power and control signals has to be connected to the corresponding signals of your embedded on-board device. Below is a list of these signals for devices that belong to Kinetis ARM

Freescale family. A full schematic of this OSBDM design can be downloaded from www.pemicro.com/OSBDM.

Table 5.4 Kinetis ARM Power and Control Signals

Signal	**Description**
GND	This is the common ground signal. It should be connected to the ground plane of the target microprocessor board. This ground is connected in the Reference Design to the PC ground via the USB port.
5V_TRG	This 5V regulated output signal provides a power output connection to the rest of the evaluation board. The power for this rail comes from the USB Bus and is controlled by the Reference Design. After the Reference Design enumerates on the PC’s USB Bus, this voltage output is enabled. The evaluation board should draw no more than 250mA of current from this connection. The evaluation board should provide sufficient storage and bypass capacitance on this line.
3V_TRG	This 3.3V regulated output signal provides a power output connection to the rest of the evaluation board. The power for this rail comes from the USB Bus and is controlled by the Reference Design. After the Reference Design enumerates on the PC’s USB Bus, this voltage output is enabled. The evaluation board should draw no more than 250mA of current from this connection. The evaluation board should provide sufficient storage and bypass capacitance on this line. The 3V_TRG voltage source and the 3V on-board regulator are both optional.
V_TRG	This input signal should be connected to the regulated voltage at which target Freescale device is running. OSBDM in turn uses this voltage source to power communication circuitry and to ensure that OSBDM is talking to on-board device at the correct voltage level.

Table 5.4 Kinetis ARM Power and Control Signals

Signal	Description
TXD_VIRTUAL_TGT	This line should be connected to the TXD signal of the SCI module on the target microprocessor. Through this line OSBDM will be receiving traffic from the serial module of the target MCU. This connection will be used to accommodate sending Serial Data to the Virtual Serial Port through the Reference Design. It will also allow the user to use the P&E toolset to capture the traffic coming from the serial port of the user's target microprocessor.
RXD_VIRTUAL_TGT	This line should be connected to the RXD signal of the SCI module on the target microprocessor. Through this line OSBDM will be transmitting traffic to the serial module of the target MCU. This connection will be used to accommodate sending Serial Data from the Virtual Serial Port through the Reference Design. It will also allow the user to use the P&E toolset to capture the traffic coming from the serial port of the user's target microprocessor.
TDO	This input signal should be connected to the evaluation board processor's TDO pin. TDO is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.
TMS	This input signal should be connected to the evaluation board processor's TMS pin. TMS is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.
TDI	This output signal should be connected to the evaluation board processor's TDI pin. TDI is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.

Table 5.4 Kinetis ARM Power and Control Signals

Signal	Description
TCK	This bi-directional signal should be connected to the evaluation board processor's TCK pin. TCK is one of the signals that allow the reference design to communicate with the evaluation board processor via the background debug module.
RESET_B	This bi-directional signal should be connected to the RESET line of the microprocessor. This signal allows the reference design to reset the evaluation board processor and also detect resets of the evaluation board processor.
ELE_PS_SENSE	The ELE_PS_SENSE is an optional input signal that allows for better integration of the OSBDM design within Freescale's Tower Cards and Elevator evaluation framework. Please refer to ELE_PS_SENSE Signal for more details.

5.1.5.1 Power and Enumeration Status Indicators

The green Enumeration Status and a yellow Target Power (TPWR) indicator play an important role in OSBDM design. Both LEDs are solely controlled by OSBDM firmware. Enumeration status LED lights up once OSBDM has been properly enumerated on the USB bus by Windows or Linux operating systems. The TPWR LED illuminates if OSBDM detects target power, supplied to the OSBDM design via V_TRG input signal.

5.1.5.2 On-Board Virtual Serial Port

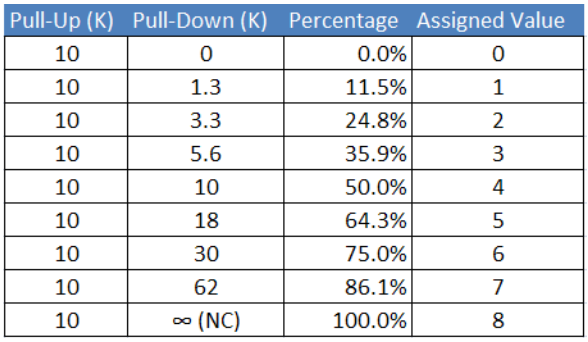
The OSBDM tower board has a built-in virtual serial port which may be connected to the microprocessor's SCI RXD/TXD. Virtual serial port allows to capture and transmit serial traffic through PC USB interface. Once OSBDM is plugged into USB port, virtual serial port enumerates on USB as a separate USB device. It then handles all communication between PC applications and SCI module of on board device. Virtual serial port is a very convenient OSBDM feature, which simplifies capturing serial data and at the same time eliminates the need for onboard RS232 serial port hardware. OSBDM will be receiving traffic from the SCI TXD of the target MCU through the TXD line. OSBDM will be transmitting traffic into the serial module of the target MCU through the RXD line. A set of different virtual serial port utilities are available at no charge from P&E Microcomputer Systems. This toolset collection includes Accelerometer Demo, Serial Grapher, Serial Redirector and Terminal Utilities. For more information and a download link, please go to www.pemicro.com/OSBDM

5.1.5.3 OSBDM/OSJTAG Revision Signals

The OSBDM tower board has a set of signals (2:0) BRD_REV0/1/2 that act as an OSBDM ID to help keep track of the different versions of OSBDM designs, as seen in Figure 5.5. Their values are set based a combination of pull up and pull down resistors. On the other hand, there are also BRD_ID0/1 signals that are connected to ATD inputs and designed to help identify specific target device that resides on a given evaluation board. It is also controlled via a set of pull up and pull down resistors. Prior to designing a given evaluation board, a user should contact Freescale or P&E Microcomputer Systems to request an OSBDM Revision and Board ID. This will allow P&E to add special handling to firmware and run control software layer in case if a workaround is required for a silicon or board defect. Upon request, the user will be assigned a numerical decimal BRD_ID and OSBDM_REV ID. While OSBDM_REV IDs only range between 0 and 7, BRD_IDs can have up to 81 different values based on the following combination scheme of pull up/pull down resistors.

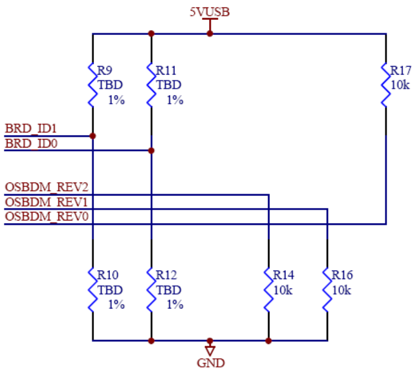
NOTE OSBDM CFV234 design does not support BRD_ID and OSBDM_REV IDs.

Figure 5.4 OSBDM/OSJTAG Board ID



Pull-Up (K)	Pull-Down (K)	Percentage	Assigned Value
10	0	0.0%	0
10	1.3	11.5%	1
10	3.3	24.8%	2
10	5.6	35.9%	3
10	10	50.0%	4
10	18	64.3%	5
10	30	75.0%	6
10	62	86.1%	7
10	∞ (NC)	100.0%	8

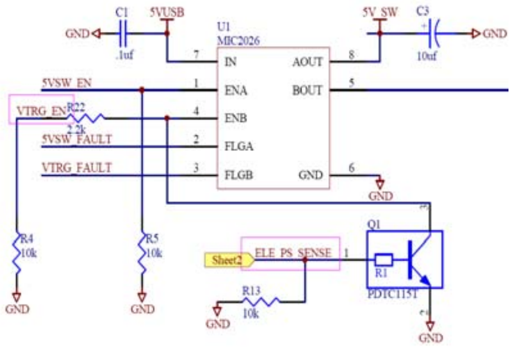
Figure 5.5 OSBDM/OSJTAG Revision Signals



5.1.5.4 ELE_PS_SENSE Signal

The ELE_PS_SENSE is an optional input signal that allows for better integration of the OSBDM design within Freescale's Tower Cards and Elevator evaluation framework, as seen in Figure 5.6. This input signal can also be used if the user decides to use their own power source for the evaluation board. Once the signal senses (a signal high) that the user selected the Elevator's voltage source or the user's own power source to provide power to the rest of the system, the voltage generation circuitry within the OSBDM design will automatically shut off by pulling VTRG_EN low.

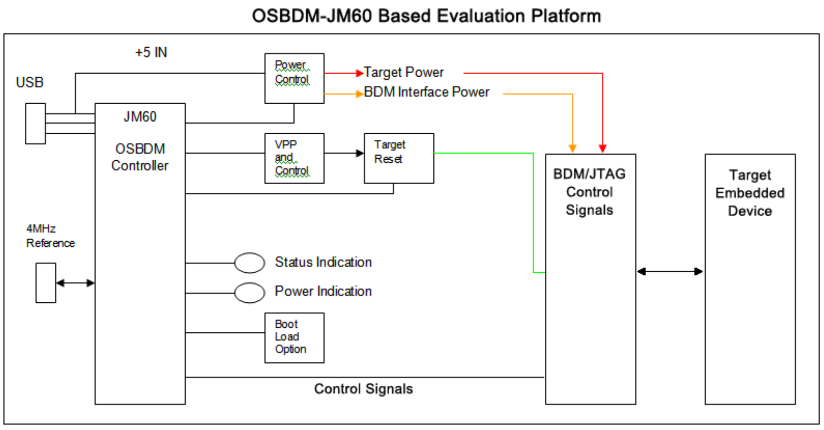
Figure 5.6 ELE_PS_SENSE Signal



5.2 OSBDM Hardware Application

The OSBDM JM60 provides and controls target devices via PC OS DLL and outlined set of API calls. The OSBDM-JM60 reference design provides support for devices belonging to a wide range of Freescale architectures: S08/RS08/CFV1/HC(S)12(X), ColdFire V234, Kinetis ARM, DSC, and PPC Nexus Qorivva. The following sections describe the circuits, the type of devices supported, OSBDM-JM60 schematic diagrams, and a bill of materials list. The figure below shows the block diagram of OSBDM-JM60.

Figure 5.7 OSBDM-JM60 Hardware Block Diagram



5.2.1 OSBDM and JM60 Internal Support Circuits

OSBDM based software applications use OS DLL and underlying JM60 OSBDM hardware design to communicate with onboard embedded microprocessor. This section describes the OSBDM circuits that support different families of Freescale devices.

5.2.1.1 Reference Clock

The JM60 applies a 4.00MHz crystal reference for USB communication timing.

JM60 initialization configures the Multipurpose Clock Generator (MCG) to enable and apply the external crystal reference. MCG configuration sets the JM60 core and USB clocks to 48Mhz and the bus clock to 24Mhz for BDM operation.

5.2.1.2 USB Connection

The JM60 USB 2.0 connection is the primary host communication interface for any application that is designed to communicate to target device via OSBDM. It is also used to support a built in Virtual Serial port functionality, that is a part of the latest OSBDM firmware. The connection also powers the OSBDM design and provides 3.3V and 5V power sources to the rest of the target board. Please note that total current draw of embedded OSBDM design and the rest of evaluation system should not exceed 0.5A.

5.2.1.3 Power Control

The OSBDM provides a dual power switch with current limit to power the OSBDM internally and the target, if enabled. At initial USB connection, the OSBDM is limited in power consumption by USB specifications and the power switches are OFF. After the USB connection with the host PC, additional power is applied and the JM60 will power the OSBDM internal circuits with switch A. During OSBDM application operation, the JM60 will power the target with power switch B.

Power switch A provides +5V_SW signal to power the internal OSBDM circuitry.

Power switch B provides +5V_TRG output voltage to power the rest of the evaluation target board. +5V_TRG can be in turn regulated down to 3.3Volts via an optional on board 3Volt regulator.

NOTE The OSBDM current should not exceed 500ma from the USB connection. This includes OSBDM and target power. Damage to the host PC USB port may occur if the current exceeds 500ma. Use caution while powering target boards from the OSBDM.

5.3 Hardware Schematic Design

Please refer to the architecture-specific JM60 based OSBDM schematics for development:

HCS08/CFV1/HC(S)12(X)

Use the reference design found within the OSBDM package called OSBDM_S08_HCS12_CFV1.pdf

RS08

Use BOM_RS08.xls.

The RS08 OSBDM design has a separate dedicated Bill of Materials, even though it will remain a part of the HCS08/RS08/CFV1/HC(S)12(X) schematic.

ColdFire V234

Use the reference design found within the OSBDM package called OSBDM_CFV234.pdf

Kinetis ARM

Use the reference design found within the OSBDM package called OSBDM_ARM.pdf

Qorivva MPC55xx/56xx & DSC

Use the reference design found within the OSBDM package called OSBDM_QORIVVA_DSC.pdf

5.4 Bill of Materials

Please refer to the architecture-specific JM60-based OSBDM Bills of Material (BOM) for development:

HCS08/CFV1/HC(S)12(X)

Use the BOM found within the OSBDM package called BOM_OSBDM_S08_RS08_HCS12_CFV1.xls

RS08

Use the BOM found within OSBDM package called BOM_RS08.xls

ColdFire V234

Use the BOM found within the OSBDM package called BOM_OSBDM_CFV234.xls

Kinetis ARM

Use the BOM found within the OSBDM package called BOM_OSBDM_ARM.xls

Qorivva MPC55xx/56xx & DSC

Use the BOM found within the OSBDM package called BOM_OSBDM_QORIVVA_DSC.xls

5.5 OSBDM-JM60 Specifications

Table x describes the OSBDM-JM60 specifications.

Table 5.5 OSBDM-JM60 specifications

Specifications	
Input Voltage	+5V from USB connection or JM60 BDM port, 40mA internal power
Output Voltages	+5V to target, 400mA maximum +3.3V to target, 250mA maximum +12V VPP programming voltage, 20mA maximum
BGND Operation (HCS08, RS08, HC(S)12(X), ColdFire V1	500KHz to 25MHz target clock for BDM, 1.8 to 5V signaling
ColdFire V2/3/4	500KHz minimum target clock for BDM, 2.5 to 5V signaling
JTAG (ARM Kinetis, Qorivva)	1MHz minimum target clock, 2.5 to 5V signaling

6

OSBDM Commercial Alternatives

OSBDM is a low-cost debugging and programming product for Freescale devices which is intended for use in embedded evaluation boards, like Freescale’s Tower Cards, for rapid prototyping and firmware development. When users move on to building their own boards and going into production, they can choose to use P&E Microcomputer System’s cost effective tools without the feature and support limitations of OSBDM tools:

- Users of P&E’s hardware will easily find that improved performance is becoming an integral part of their development process.
- OSBDM can calculate a ±2.0% TRIM value for HCS08, RS08, and ColdFire V1 devices, while P&E’s hardware has advanced features with super-accurate TRIM calculations of ±0.2%.
- The Cyclone Pro and Cyclone Max allow utilization of serial, USB, and Ethernet interfaces for flexibility in how it communicates with the PC.
- Support of wider range of communication frequencies (32kHz-50MHz for HCS08/RS08/CFV1 devices)

P&E Microcomputer Systems, in addition to its software development tools, has fast, low-cost, supported hardware interfaces which provide a seamless transition from embedded OSBDM products. These work with a variety of third-party software tools. Below is an overview of the features and intended use of P&E’s Multilink Universal and Cyclone Pro and Max to assist users in choosing the appropriate development tool. Please visit www.pemicro.com for more information.

6.1.USB Multilink Universal

The USB Multilink Universal is a powerful and cost effective development tool that holds a performance edge over existing OSBDM hardware interfaces. Users doing rapid development will find this interface easy to use and fully capable of fast-paced debugging and programming.

Figure 6.1 USB Multilink Universal

The USB Multilink Universal includes features such as:

- Direct user control of target's execution
- Programming and debugging capabilities
- Read/write registers and memory values
- Compact and lightweight
- Communication via USB 2.0
- Supported by P&E software and Freescale's CodeWarrior
- All-in-one support for Freescale Qorivva MPC55xx/56xx, HCS08, HC(S)12(X), RS08, ColdFire V1/+V1, ColdFire V2-4, and Kinetis ARM.

6.2.Cyclone Pro and Cyclone Max

Cyclone Pro and Cyclone MAX are advanced, yet cost effective hardware interfaces that support FLASH programming of devices in standalone mode. A Cyclone Pro and Cyclone MAX user can preprogram Cyclone with multiple images targeting different devices, belonging to different Freescale architectures. One can then select any image to program into a given device, using a built in LCD screen, without direct PC connection. The Cyclone Pro and Cyclone Max are more complete solutions designed for both development and production.

Figure 6.2 Cyclone MAX Automated Programmer and Debug Interface

Some of their features include:

- PC-Controlled and User-Controlled Stand-Alone Operation
- Interactive Programming via Host PC
- In-Circuit Debugging, Programming, and Testing
- Communication via USB, Serial, and Ethernet Ports
- Multiple on-board image storage and optional expandable compact flash memory
- LCD screen menu interface
- Supported by P&E software and Freescale's CodeWarrior
- Cyclone Max supports Freescale ColdFireV2/3/4, Qorivva MPC55xx/56xx, and Kinetis ARM microcontroller families
- Cyclone Pro supports Freescale HCS08, HC(S)12(X), RS08, and ColdFire V1/+V1
- The Cyclone Pro on-board relays can provide and/or control switching of the target's power via the ribbon cables.