USBDM (Combined TBDML/OSBDM code) for JB16

Introduction

The attached files provide a port of the TBDML/OSBDM code to a MC9S08JJB16 processor.

Features

This version of the code & extended hardware provides the following features:

- A single BDM interface for programming and debugging of the following targets:
  - HCS12 (including those without SYNC feature (e.g. 9S12DP256B)
  - HCS08,
  - RS08 and
  - Coldfire V1 microcontrollers
- Compatible with CW for HC12 V4.6, V4.7
- Compatible with CW for Microcontrollers V6.1, V6.2 (required for Coldfire V1 chips)
- A USB2 full speed interface.
- Target reset detection & control (required for HC12 processors). This allows HC12 processors to be reset into BDM mode as required for debugging.
- Target Vdd supply & control. This allows the target power to be cycled when required for a reliable reset into BDM mode. It is also (obviously) convenient since it allows debugging small target boards without a separate target power supply. An external target supply may also be used instead. The Target supply is protected by a polyfuse and software detection of overload.
- Target Vdd monitoring. This allows the hardware to detect changes in the Target supply to provide reliable resetting into BDM modes on HCS08 and RS08 microcontrollers. It also is used to implement the target supply protection.
- A slightly higher BDM interface speed than the existing JB16 OSBDMs. Up to 20MHz.
- Several software options over the existing OSBDM code
  - Automatic re-connection. The existing OSBDM code appears to have problems debugging a target when the target clock speed is changed by the code execution. The software has an option to continuously update the interface speed to prevent loss of communication in this case.
  - Control of BDM clock selection in HCS/RS08/Coldfire microcontrollers. Most of these targets provide an alternative BDM clock selection. This can have advantages when connecting to high speed targets.
  - Selection of 3.3V and 5V target supply.
  - Trial-and-error determination of communication speed for earlier HC12 targets.
- Various bug fixes on the OSBDM code.
- Hardware is physically small (50 mm x 25 mm)

Capabilities

The software is compatible with several of the TBDML/OSBDM hardware variants that are available. Capabilities will vary with the actual hardware.

The table below summarises the capabilities of the BDM for various hardware options:

Target	HC12/HCS12	HCS08	RS08	Coldfire V1
TBML	X	X	-	X
USBDM	X	X	X	X
HCS08-OSBDM+E		X	X	X
OSBDM	X	X		X
WTBDM08	X	X		X

Differences & disadvantages

- The software drivers (**tbdml.dll** & **opensourcebdm.dll**) need to be replaced so that advantage can be taken of the new features.

The bulk of the code is taken from the TBDML and OSBDM projects and I very much appreciate the effort involved in producing the original code. I hope this extension will be useful.

The hardware interface is based on TBDML and Freescale USBSPYDER08.

Any queries please post on the Freescale OSBDM discussion board.

The following is provided

“Installation” folder

1. **tbdml.dll**
   This is a replacement DLL for “CW for HC12 V4.6 or V4.7”. This DLL allows the HCS12 version of Codewarrior to talk to the modified TBDML which actually identifies itself as an OSBDM.
   The original **tbdml.dll** file is located in the windows directory (I think). Copy the one provided to **\Program Files\Freescale\Codewarrior for HC12 V4.7\Prog\gdi** where it will be found first. This allows easy un-installation – just delete or rename the file in the gdi directory.

2. **opensourcebdm.dll**
   This is a replacement DLL for “CW for Microcontrollers V6.1 or V6.2”. This DLL allows the HCS08/RS08 version of Codewarrior to talk to the modified hardware and access the extended features. Note: the hardware identifies itself as an OSBDM interface but is not completely compatible.
   The original **opensourcebdm.dll** file is located in the windows directory (I think). Copy the one provided to **\Program Files\Freescale\CodeWarrior for Microcontrollers V6.2\prog\gdi** where it will be found first. This allows easy un-installation – just delete or rename the file in the gdi directory.

3. **TestOSBDM.exe**
   This is a simple command line program that allows the USBDM hardware to be exercised. It is mostly provided for debugging the hardware. It is self-contained and does not require the above DLLs.

4. **Setboot.exe**
   This command line executable can be used to place the programmed BDM modules into ICP mode for re-programming. This is only usable after initial programming of module.

5. **DebugDLLs folder**
   This contains debug versions of the above DLLs. These create copious log files in the C root directory!

6. **USBDM_Flash_Images**
   This folder contains pre-compiled images for the JB16 flash. These are provided for several common hardware platforms that are available.

5. **libusb-win32-filter-bin-0.1.12.1.exe**
   The USB interface library used. See http://libusb-win32.sourceforge.net/

<u>“Source” folder</u>

This contains the sources for the software (above DLLs and JM60 code).

1. **OSBDM_DLL_Snapshot_xxxxxx.zip**
   Source for the **tbdml.dll** and **opensourcebdm.dll** files. This is an exported file from Eclipse CDT for windows. To rebuild these DLLs you will also need the USB interface DLL (LIBUSB).
2. **USBDM_JB16_Snapshot_xxxxxx.zip**
   Source for the JB16 code. This is a simple Zip file of the Codewarrior project.
3. **HCS08 JB16 BDM Timing.xls**
   This spreadsheet was used to calculate the timing for the bdm_rx() and bdm_tx routines.
4. **RS08FlashProgramming.zip**
   This is the source code for the RS08 programming routine that is downloaded to the RS08 by the BDM when programming the flash memory.

Installation

Refer to the original OSBDM documentation for detailed information on how the JB16 is programmed using the Freescale ICP software.

Initial programming of a blank device or a device containing an incompatible program.

1. **Get the USBDM code for the JB16 flash memory**.

Choose a pre-compiled image from the *USBDM_Flash_Images* directory or re-build the code from scratch using the source files provided in the *source* directory.

2. **Install the modified *tbdml.dll* and *opensourcebdm.dll* files**

Copy the ***tbdml.dll*** and ***opensourcebdm.dll*** files to the appropriate HC12 Codewarrior directories. See the discussion earlier.

3. **Make a backup of the TBDML code.**

Make sure you have a copy of your current TBDML/OSBDM code so you can restore it if my code doesn't work with your hardware.

4. **Initial download of code to JB16 flash**

This initial download of code to the JB16 installs a ICP boot loader. It is intended that the boot loader code would remain permanently in the JB16 to help prevent (the unlikely) event of the JB16 becoming unusable due to Flash programming failing or being interrupted.
If the code is updated again the Boot loader is not changed. **The instructions below ONLY APPLY to the INITIAL DOWNLOAD.** Updating the code later is a slightly different process.

Use whatever method your original TBDML hardware/software used to get into ICP mode. For most hardware this was holding port pin A.0 low while plugging in the BDM.

The Freescale ICP programming software is used to download the code image to the JB16 Flash. The software requires two files (both in ***USBDM_Flash_Images*** directory):

a. A file to configure the ICP programming software for **MASS erase configuration** for this initial download (***Initial Programming_MassErase.imp***).
b. The Flash file image for the JB16 Flash. This depends upon the hardware being used. Choose the appropriate file to download from the ***USBDM_Flash_Images*** directory.

Updating the OSBDM/TBDML code after the ICP code has been loaded the first time.

Do **one** of the following:

Hardware triggered ICP - Connect pin A.0 to ground and plug in the cable. The hardware will boot in ICP mode.

OR

Software triggered ICP -Plug in the BDM hardware and run ICPBOOT.exe. The hardware will be reprogrammed to boot into ICP. Remove and replace the cable to get into ICP mode. This is persistent change. It is necessary to reprogram the cable to restore operation.

Use the Freescale ICP programming software to download the new OSBDM/TBDML code.

It will be necessary to use the **BLOCK erase configuration** file for this (***Re-programming_BlockErase.imp***) so that the ICP code is unchanged.

Completely removing the OSBDM/TBDML code after the ICP code has been loaded.

Do **one** of the following:

Hardware triggered ICP - Connect pin A.0 to ground and plug in the cable. The hardware will boot in ICP mode.

OR

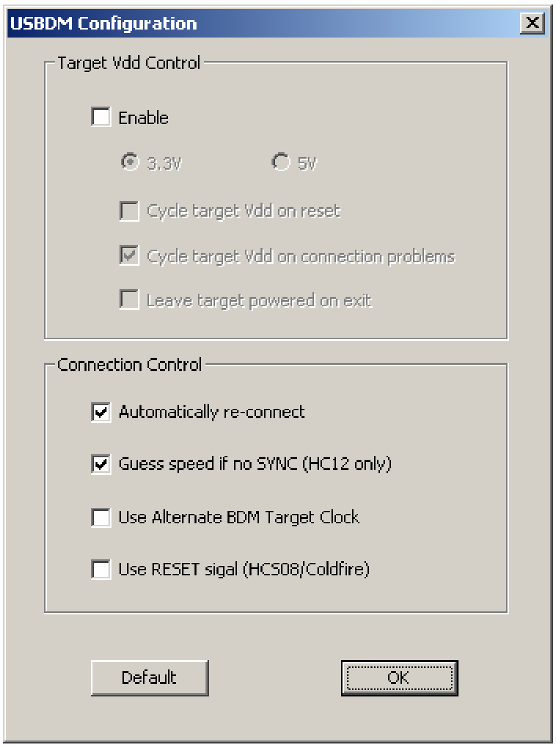
Software triggered ICP -Plug in the BDM hardware and run Setboot.exe. The hardware will be reprogrammed to boot into ICP. Remove and replace the cable to get into ICP mode. This is **persistent** change. It is necessary to reprogram the cable to restore BDM operation.

Use the Freescale ICP programming software to erase the entire Flash memory.
It will be necessary to use the **MASS erase configuration** for this (***Initial Programming_MassErase.imp***) so that the ICP code is removed as well as it is only usable with the USBDM code.
You may now restore your original Flash code and delete the two dll's installed at step 2 earlier.

Using the debugger

When creating a project choose the appropriate debugger option as follows:

CW for HC12	TBDML (Not all targets support this option)
CW for Microcontrollers V6.2	HCS08 Target - HCS08 Open Source BDM RS08 Target - RS08 Open Source BDM Coldfire V1 Target - CFv1 Opensource BDM



USBDM Configuration

Target Vdd Control

☐ Enable

● 3.3V ○ 5V

☐ Cycle target Vdd on reset

☑ Cycle target Vdd on connection problems

☐ Leave target powered on exit

Connection Control

☑ Automatically re-connect

☑ Guess speed if no SYNC (HC12 only)

☐ Use Alternate BDM Target Clock

☐ Use RESET sigal (HCS08/Coldfire)

When starting the debugger you will be presented with the dialogue at left. After closing this dialogue debugging will proceed as usual. Some options in this dialogue may be disabled depending upon the hardware capabilities of the BDM interface.

Target Vdd Control

Enable – Enables supply of Target Vdd from the USBDM module

3.3V or 5V- Select target supply voltage

Cycle Target Power on reset – This option will cause the BDM to cycle the target power as part of the reset sequence.

Cycle Target Power on connection problems – This option will cause the BDM to cycle the target power when it is having trouble connecting to the target.

Leave Target Powered on exit - The target Vdd supply will be left on when exiting the debugger.

Connection Control

Automatically re-connect - The existing OSBDM code appears to have problems debugging a target when the target clock speed is changed. The option causes the BDM to continuously update the interface speed to prevent loss of communication in this case.

Guess speed if no SYNC(HC12 only) - Early HCS12s do not support the SYNC feature which allows the BDM interface speed to be determined. Selecting this option will cause the BDM to attempt to find the communication speed by trial and error. This can take quite a while.

Use Alternative BDM target Clock - Control of BDM clock selection in HCS/RS08 microcontrollers. Most of these targets provide an alternative BDM clock selection. This can have advantages when connecting to high speed targets (mostly of use in the JB16 version).

Use RESET signal (HCS08/Coldfire) – Many of the HCS08, RS08 and Coldfire microcontrollers do not have a dedicated reset signal. This is not a problem as it is possible to reset the target using the BKGD mode commands through the BDM interface. The RESET signal on the BDM interface will often still be connected to the Reset signal of the processor. This option allows the BDM to monitor and control the RESET signal when appropriate. By default, the RESET signal is ignored when using the BDM with targets other than the HC(S)12.
Note: HC(S)12 targets **require** the use of the RESET signal and ignore this option.
Note: The current implementation may still spuriously drive the RESET pin low on the BDM interface when resetting the target. (Work in progress)

Notes:

Bug Fixes:

- An error in the OSBDM code involving BDM08_CMD_WRITECONTROL has been removed. This macro had the side effect of disabling ACKN. There are some other similar bugs.
- The timing for the bdm_rx() routines has been changed. The original code did not take into account the phase difference between a write to, and a read from, a port. This phase difference is due to two possible delays:
  a. Between writing to a port and the value appearing on the port pin, and
  b. An input pin change and that change being readable at the port register.

Other changes

- The bdm_tx()/bdm_rx() routines have been updated for the JB16.
- Generic bdm_tx()/bdm_rx() routines have been written for low frequencies.
- USB code has been replaced.
- Coldfire V1 code added.
- The code has been extensively re-arranged. I’m sorry, I couldn’t stop myself ☺

References

- LIBUSB-WIN32 - http://libusb-win32.sourceforge.net/
- USBSPYDER08 - USB mini board for Freescale's Low-End 8-bit Microcontrollers – http://www.freescale.com
- MAX662 Data sheet - http://www.maxim-ic.com
- I2C voltage level conversion - Phillips application note http://www.standardics.nxp.com/support/documents/i2c/pdf/an97055.pdf
- TBDML - http://forums.freescale.com/freescale/board?board.id=TBDML
- OSBDM - http://forums.freescale.com/freescale/board?board.id=OSBDM08