Developing code for the element14/Freescale XL_STAR board

Document Revision: 1.04 07/2011

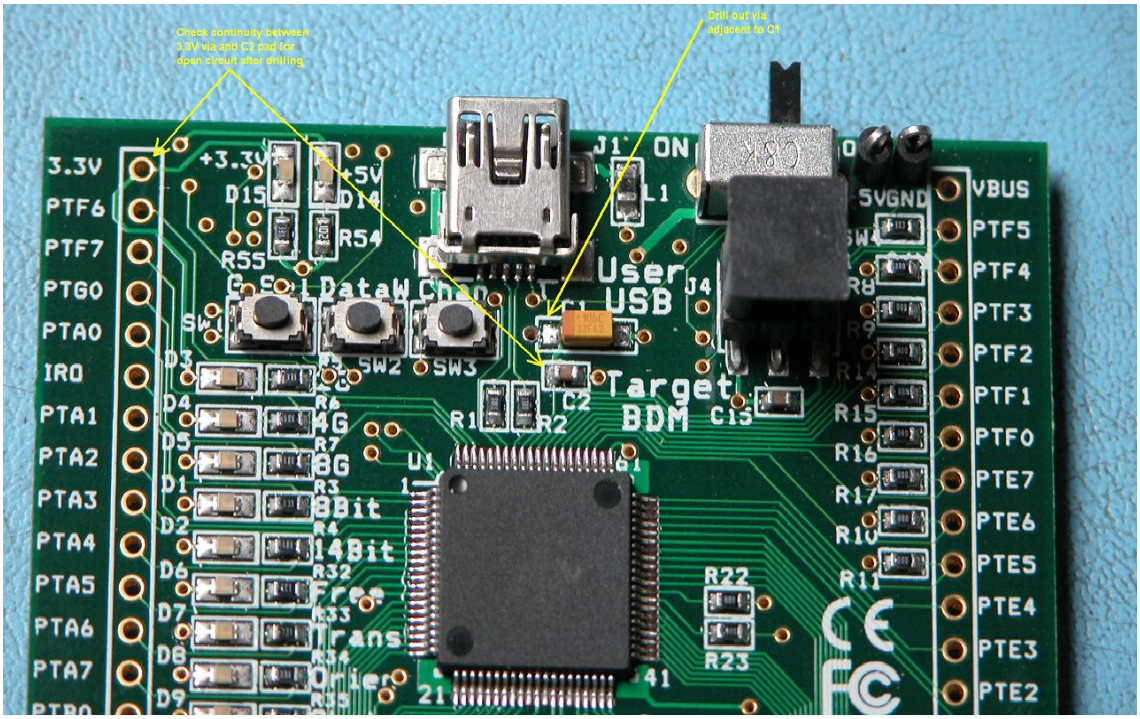
element14

freescale™ semiconductor

Please note – when using the XL_STAR for USB device development, via J1 (the USB connector on the top side of the XL_STAR), the XL_STAR **must** be powered via a USB cable connected to J6 (the USB connector on the underside of the XL_STAR) **before** a USB cable is connected to J1.

If USB cables are not connected in this order the USB regulator on U1 will be over loaded and **permanent damage to U1 may occur.**

For those of you who wish to permanently modify the XL_STAR board so that this damage will not occur the picture below details a relatively simple modification. In summary a via next to capacitor C1 on the top side of the board must be drilled out breaking the connection of the +3.3V supply to C1 and C2. The break in connectivity can be confirmed by placing a multi-meter set for resistance measurement between the +3.3V supply (top left plated through hole) and the left hand pad of C2, which should show high impedance.



Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Table of Contents

1. Introduction

This document describes the process of developing code for Freescale's XL_STAR, a low cost development board featuring one of the most recent members of the HCS08 family of 8-bit microcontrollers, the MC9S08MM128.

The document will look at the following topics:

- An introduction to the XL_STAR hardware
- How to use the sample application pre-programmed onto the board
- How to install Freescale's *CodeWarrior for Microcontrollers V6.3* Integrated Development Environment (IDE) on a Windows PC
- How to compile and build a sample HCS08 project
- Programming the code into the Flash memory on the HCS08
- Debugging
- Writing your own code for the XL_STAR board

If you are keen to experiment before reading this manual, just switch on the board and try the pre-programmed application. It features a number of demos designed to show some of the things you can do with the on-board accelerometer. You can find details about the available demo modes in Section 3.

NB: Do not connect the board to a PC until you have installed CodeWarrior, because the PC won't recognise the board until a USB driver has been installed.

2. XL_STAR board hardware

The XL_STAR is a double-sided board. It may be simplest to imagine that the two sides are separate circuit boards that are stuck together. As a developer you will mainly be concerned with the top side.

2.1. Top Side

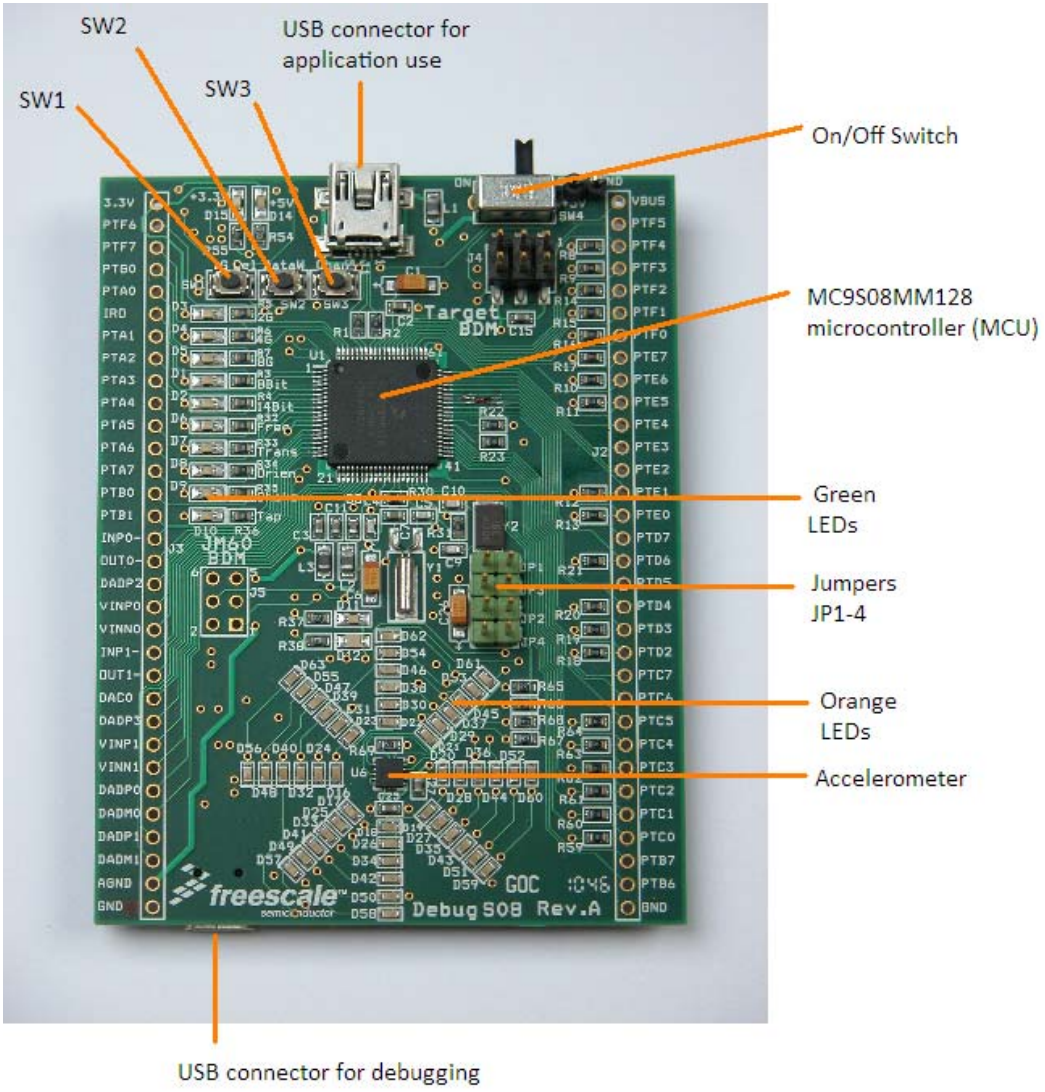


Figure 1: XL_STAR Board - Top Side

On the top side of the board (Figure 1) is the main microcontroller (MCU) - a Freescale MC9S08MM128. This is an 8-bit MCU based on the HCS08-family architecture, with the following specification:

- 128 KB of Flash
- 12 KB of RAM
- Processor speed configurable up to 48 MHz

Also on the top side is a Freescale accelerometer, the MMA8451Q. This can detect both its orientation, through the static pull of gravity, and any movement of the board which causes acceleration or deceleration.

There is a set of 48 orange LEDs arranged in an 8-pointed star with the accelerometer at its centre. The LEDs are used by the demo program to indicate information such as how the board is tilted.

Three small push buttons SW1 - SW3 allow the user to select different accelerometer demos (tilt, tap, shake, freefall detection, etc), and to configure the accelerometer. The accompanying green LEDs are used by the demo program to indicate status information.

There is an On/Off switch connected to a Lithium Ion Coin Cell battery which is mounted on the underside of the board.

Finally there is a spare USB connector next to the On/Off switch. You will not need to use this USB connector unless you are writing your own application which includes a USB interface. It is **not** the USB port which is used to debug the board.

2.2. Bottom Side

In normal use, when the XL_STAR board is not connected to a PC for debugging, most of the circuit on the bottom side of the board is not powered. The only thing in use is the battery.

The bottom side of the board (Figure 2) is required when downloading a new program from the PC to be stored in the flash on the MC9S08MM128, and when debugging the program.

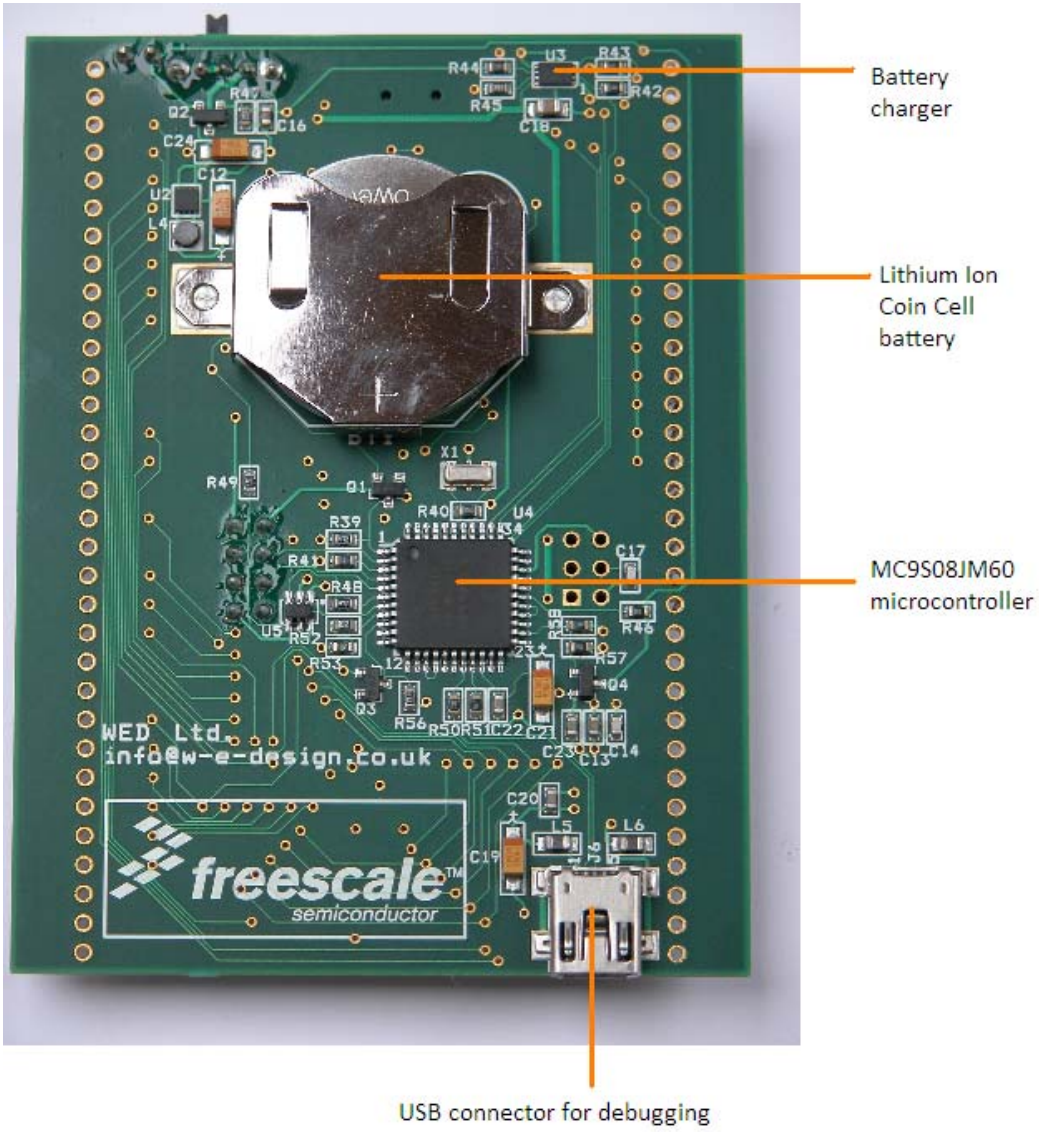


Figure 2: XL_STAR Board - Bottom Side

The bottom side of the board includes the following major components:

- The Li Ion battery
- A secondary 8-bit microcontroller (a Freescale MC9S08JM60) which accepts debugging commands from the PC and forms the Open Source Background Debug Mode (OSBDM) interface. The JM60 then controls the main MCU on the top side of the board using its BDM interface.
- The USB port used to connect the JM60 to a PC
- A Freescale MC34673 battery charger for the Li Ion battery.

When a USB cable is plugged into the underside USB port, the power for both sides of the board is taken from the PC and regulated down to 3.3V using a Freescale MC34727CFC. The battery charger will top up the battery using power drawn from the USB connection.

NB: If you have used other Freescale development systems you may be familiar with a setup in which the target board that you're writing code for is connected to the PC via special debug hardware such as a 'P & E' USB BDM MULTILINK. Such devices typically have a USB connection to connect to the PC, and a 6-pin BDM cable to connect to the target board. In the case of the XL_STAR board this separate debug interface is not required; this functionality is provided by the electronics on the bottom of the board.

2.3. Jumpers

Also on the top side of the board are four jumpers. For normal use they should be configured as follows:

JP1	**Fitted**	Access point for measuring S08MM128 MCU current
JP2	**Not fitted**	When fitted, the JM60 acts as a USB-to-Serial converter (Requires additional software)
JP3	**Not fitted**	When fitted, the JM60 runs a bootloader out of reset, allowing updating of the OSBDM debug firmware. (See Section 0).
JP4	**Fitted**	Access point for measuring MMA8451Q accelerometer current

3. The Sample Application

The XL_STAR board comes with a sample application already programmed into the on-chip Flash of the MM128 MCU. The program is designed to show some of the things that can be done with the on-board MMA8451Q accelerometer.

When first powered on, the board will go into an 'orientation' demo mode. Try picking the board up and tipping it in different directions. You should see that the orange LEDs illuminate to indicate the direction of the tip and its magnitude.

3.1. Demo Selection

You can select between five different demos by repeatedly pressing button SW3, labelled **Chan** as shown in **Figure 3**. The current demo mode will be indicated by the green LEDs D6-D10.

The available demo modes are:

Orient: Try tipping the board and watch the orange LEDs indicate the direction and magnitude of tip

Shake: Try flicking the board in a direction, either up or down, left or right, or forward or back. You need to use a sharp flick, like striking a ball in a game of table tennis. The orange LEDs will point in the direction of the initial flick, re-setting after a couple of seconds.

If the initial flick is not fast enough, the accelerometer may instead detect the rebound event as your hand stops the board's movement. In this case the reverse direction will be illuminated.

Tap: Hold the board loosely and tap it once with your finger; the LEDs will show a 'pulse' pattern. Tap twice in quick succession (double tap) to get a different animated pattern.

Freefall: Drop the board from the height of a few centimetres or more. The freefall event will be detected and displayed on the LEDs for a couple of seconds.

Transient: The orange LEDs will all illuminate if the board is moving (strictly, if it is accelerating or decelerating) and turns off after a short while if the board is stationary.

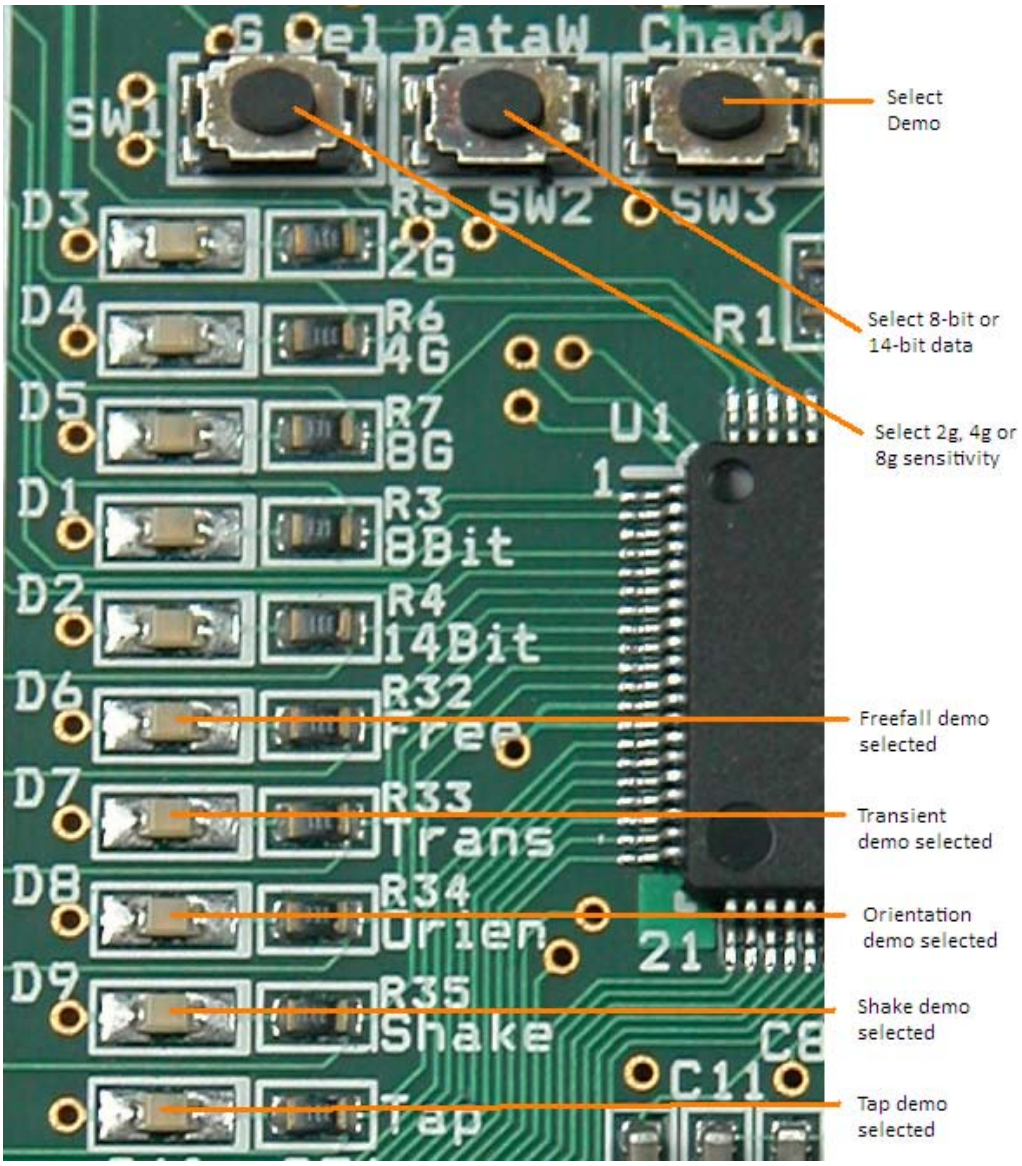


Figure 3: Demo mode selection

3.2. Changing sensitivity

The accelerometer sensitivity can be changed by pressing button SW1, labelled **G sel**. This selects which multiple of normal gravity corresponds to a full-scale accelerometer reading. The options are 2g, 4g and 8g as indicated by the green LEDs D3-D5.

Note that changing the sensitivity only affects the 'Orient' demo, because the other accelerometer functions used here don't depend on the sensitivity setting.

3.3. Selecting 8- or 14-bit data

The accelerometer can return 8-bit or 14-bit data to the MM128 MCU. Applications requiring more accuracy can use 14-bit data; otherwise it is faster to read 8-bit data. You can select between the two options by pressing the SW2 button, labelled **DataW,** see **Figure 3**. The current selection is shown on the green LEDs D1-D2.

Changing between 8-bit and 14-bit data only affects the 'Orient' demo but, although the data width setting is honoured, you probably won't notice the difference. See Section 7.3 for more information.

3.4. Low Power Mode

When the accelerometer doesn't detect any interesting events i.e. acceleration or deceleration for around 20 seconds, software will put the board into a power-saving mode. So that you can see this happening, an 'X' pattern is displayed briefly before almost all the LEDs turn off.

To wake the board up, just pick it up. A '+' pattern is displayed briefly to indicate that the board is awake again, and the current demo mode is then resumed. See Section 7.5 for more information.

4. Installing CodeWarrior

NB: *Do not connect the board to a PC until you have installed CodeWarrior, because the PC won't recognise the board until a USB driver has been installed.*

The XL_STAR development kit includes a copy of *CodeWarrior for Microcontrollers V6.3 Special Edition*. CodeWarrior is an integrated development environment for editing, compiling and debugging code. The Special Edition is free, but the C compiler is limited to a maximum of 32KB of object code. In later releases of CodeWarrior for Microcontrollers V6.3 Special Edition this was increased to 64KB of object code, please check the Freescale website for the latest release - http://www.freescale.com/webapp/sps/site/overview.jsp?code=CW_SPECIALEDITIONS&tid=CWH

4.1. System Requirements

- 1.0 GHz Pentium compatible processor or better
- Microsoft Windows XP or later
- 512 MB RAM (1 GB recommended)
- 2 GB hard disk space on Windows system disk
- CD-ROM drive for installation
- USB port for communications with target hardware

4.2. Easy Install

The software for the XL_STAR development kit is supplied on CD. When you insert the CD, the auto-run feature should launch the *XL_STAR Starter Kit Installation Wizard*. Alternatively you can double-click on the file `install_all.hta` to run the wizard.

Initial installation requires the completion of six install steps (Figure 4), the first five of which launch a separate installer. Follow the on-screen instructions to complete each step.

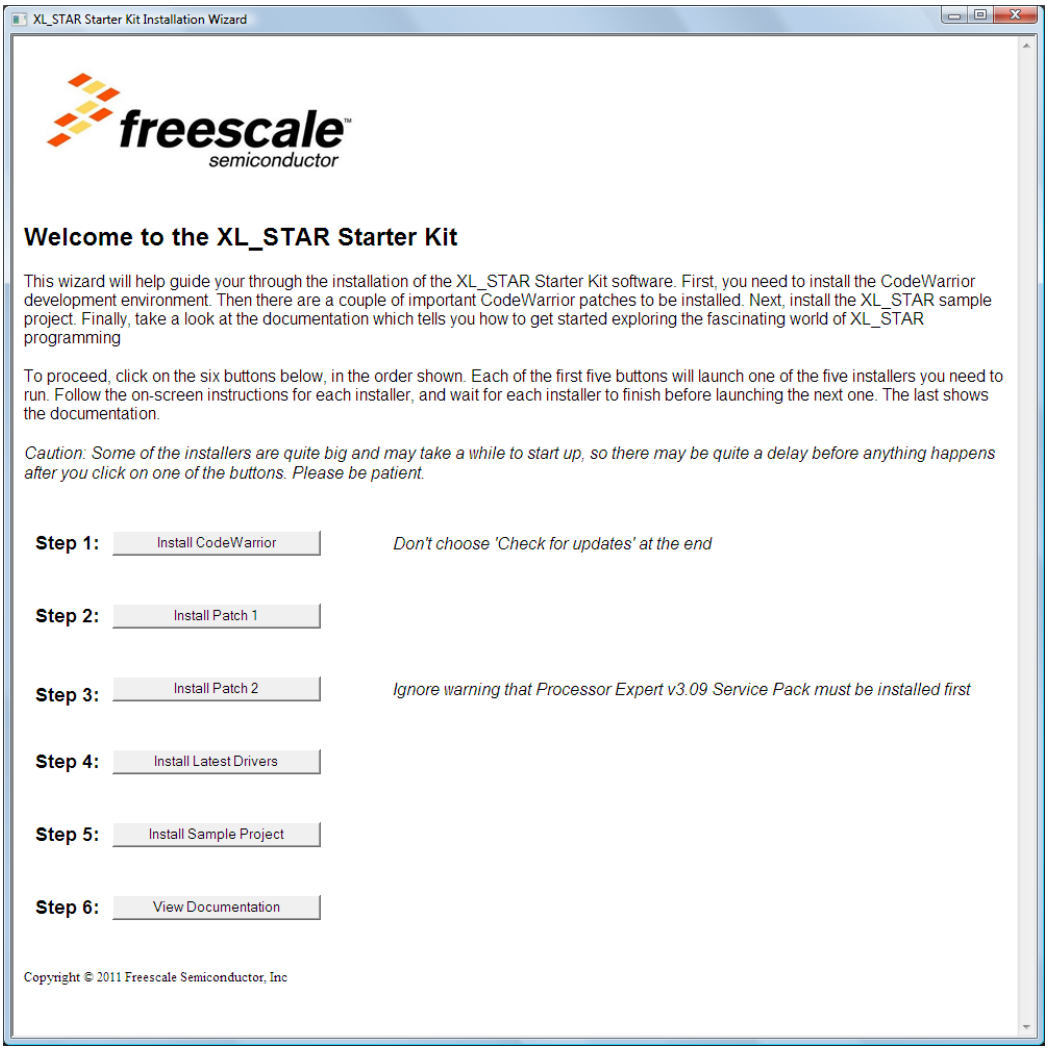


Figure 4: XL_STAR Starter Kit installation wizard

Note that there may be a long delay between clicking on one of the buttons and the individual installer starting up. Please be patient.

Make sure you wait for each install step to complete before going on to the next step.

At the end of Step 1, which installs the base CodeWarrior software, you will see a dialog asking whether you want to check for on-line updates (Figure 5). **Do not check for updates at this time**.

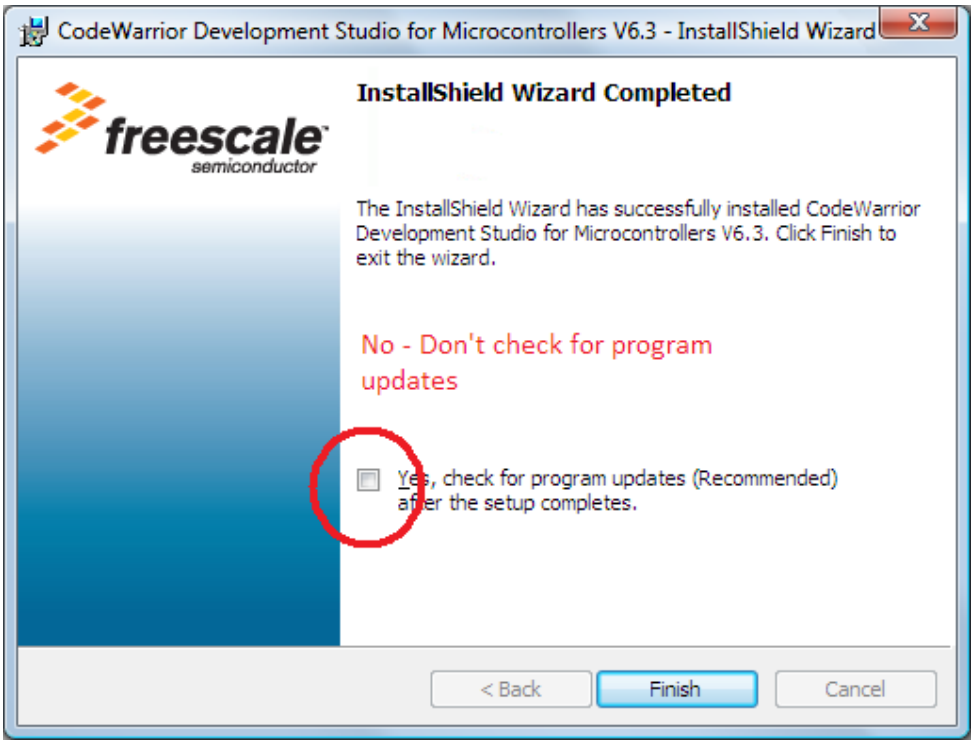


Figure 5: Do not check for updates when the first installer completes

During Step 3 you will see a warning message that *Processor Expert v3.09 Service Pack* is not installed (Figure 6). Ignore this warning and continue with the installation anyway.

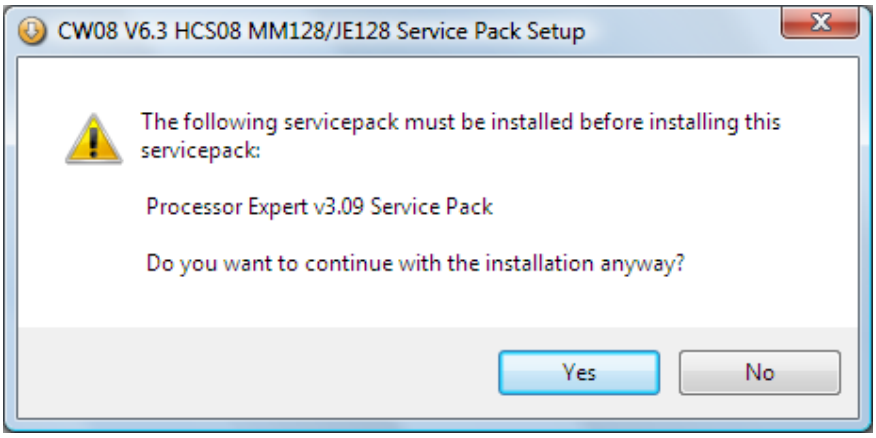


Figure 6: Ignore the warning about Processor Expert

Step 5 allows you to install a demonstration CodeWarrior project for use with the XL_STAR board. Install this in a folder that is easily writable - for example, use the `Documents` folder not `C:\Program Files` which is protected under Windows Vista and later OS versions.

Using the demonstration project is described in detail in Section 5.

4.3. Installing the USB Drivers

The XL_STAR board requires a number of USB drivers. Although the drivers were partially installed in Step 4 above you may need to carry out some additional configuration, depending on the version of Windows you are using. When doing this you should note that there are three drivers in total, and the process will need to be repeated for each. The three drivers are as follows:

OSBDM Debug Port

OSBDM Serial Port

OSBDM Bootloader Port

To install the first two drivers, make sure that jumper JP3 on the XL_STAR board is **not** fitted. Then connect the board to the PC using a USB cable. You should see a dialog similar to the following (Figure 7):

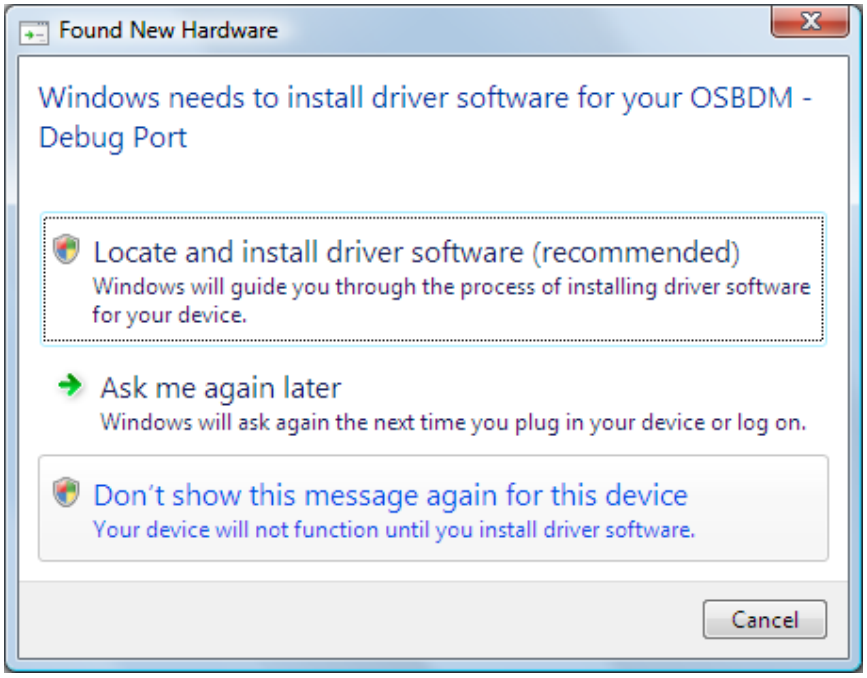


Figure 7: Installing the OSBDM Debug Port driver

Windows will look for a suitable driver in a number of locations, but fail to find one. You will then be given the option of specifying the driver location manually (Figure 8). Specify the folder in which you installed the P & E Drivers in Step 4 above, e.g. `C:\Program Files\PEMicro`

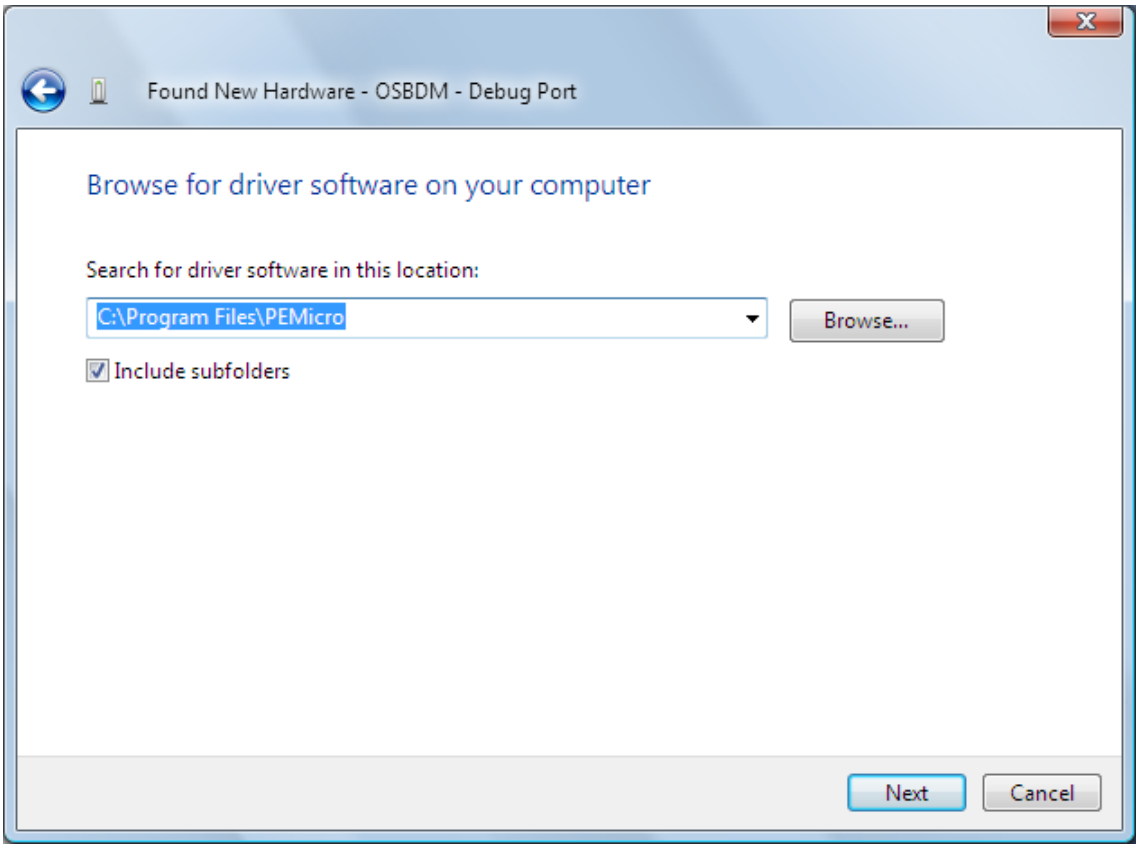


Figure 8: Specifying the location of the OSBDM Debug Port driver

After successful installation of the OSBDM Debug Port driver, you will be prompted to repeat the steps for the OSBDM Virtual Serial Port.

Finally, disconnect the cable, install jumper JP3 and then reconnect the cable. You will be prompted once more to complete the steps for the OSBDM Bootloader Port.

After installing the last driver, remove jumper JP3 and reconnect the board once more, ready for normal operation.

4.4. Checking that the XL_STAR board is recognised by Windows

You can check that the board is recognised by Windows using the Device Manager. With the board connected to the PC and Jumper JP3 not installed, the board will show up twice (Figure 9):

- *Open Source BDM Debug Port* in the section *LibUSB-Win32 Devices*
- *PEMicro USB Serial Port (i1)* in the section *Jungo*

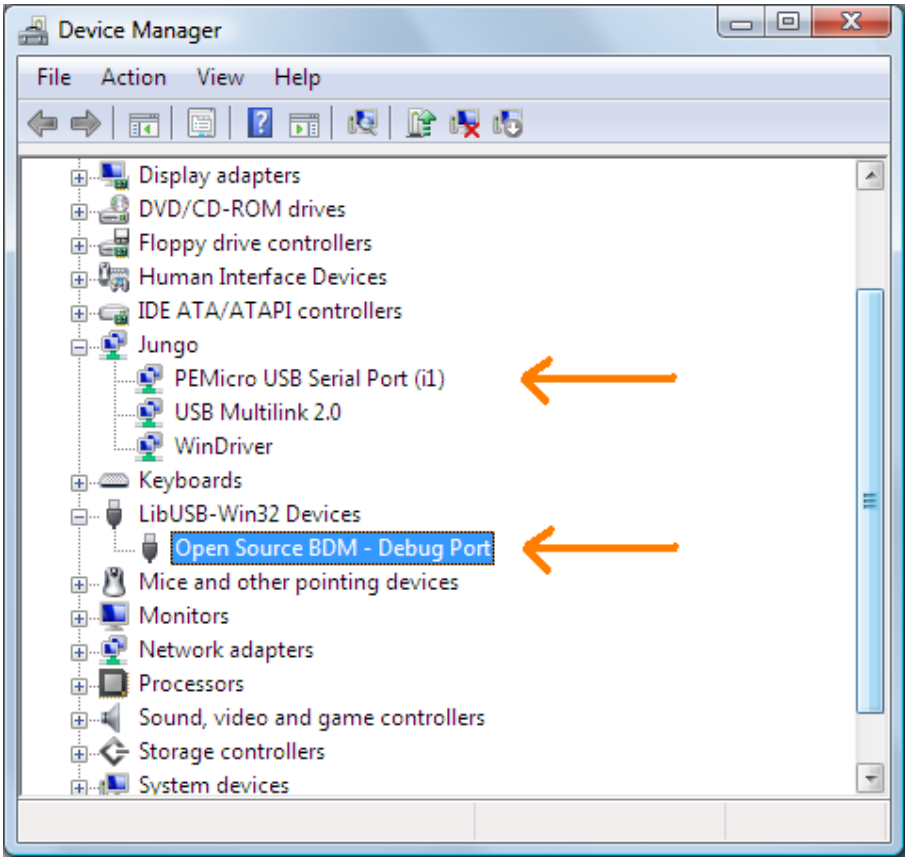


Figure 9: Windows Device Manager showing USB connections to XL_STAR board

4.5. Installing CodeWarrior 6.3 SE manually

Instead of using the installation wizard described in Section 4.2 above, you can also proceed manually, using the individual installers supplied on the CD in the following order:

(1) Install CodeWarrior for Microcontrollers V6.3

```
CW_MCU_V6_3_SE.exe
```

(2) Install the CodeWarrior for Microcontrollers V6.3.1 update

```
CWMCUV631.exe
```

(3) Install the MM128-specific patches

```
CW08_V6_3_HCS08_MM128_JE128_SP.exe
```

(4) Install the latest USB drivers need when debugging code on the board

```
PEDrivers_install.exe
```

(5) Install the sample XL_STAR CodeWarrior project

```
setup_sample_project.exe
```

5. Developing code for the XL_STAR Board

5.1. Introduction

In this section we will look at how to use the CodeWarrior IDE to develop programs to run on the XL_STAR board. The material is a brief introduction aiming to cover the following:

- Editing files
- Compiling and linking
- Programming the application into Flash memory on the MCU
- Debugging

For further information the reader is referred to the CodeWarrior documentation.

5.2. Sample CodeWarrior Project

A sample CodeWarrior project is supplied with the XL_STAR development kit and you can use it to try things out. The project builds the same demo code which is supplied pre-installed on the board and described in Section 3.

Installation of the sample project was described in Section 4.2 above.

To open the project, launch CodeWarrior and, if the startup dialog in Figure 10 is shown, select the option *Start Using CodeWarrior***.** Then use the `Open...` option from the `File` menu to navigate to the installed project and select the project file `XL_STAR_Demo.mcp`

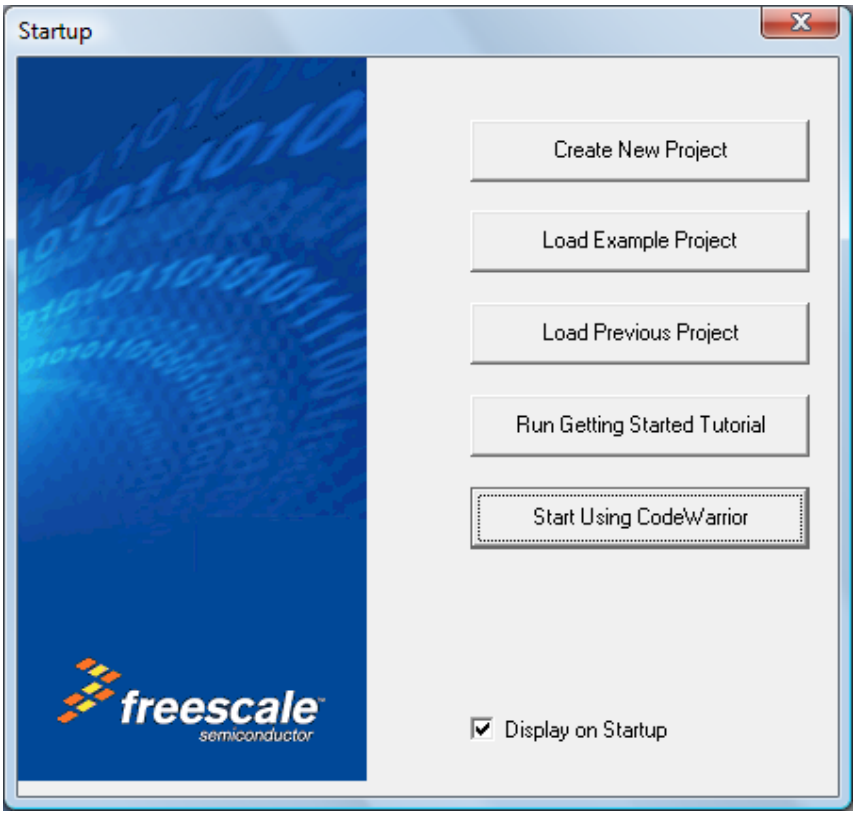


Figure 10: CodeWarrior Startup dialog

5.3. Turning off Multi-Document Interface (MDI) mode (Optional)

CodeWarrior includes support for MDI mode, in which individual CodeWarrior windows are all children of a single container window (Figure 11). This is the default mode when you first install CodeWarrior.

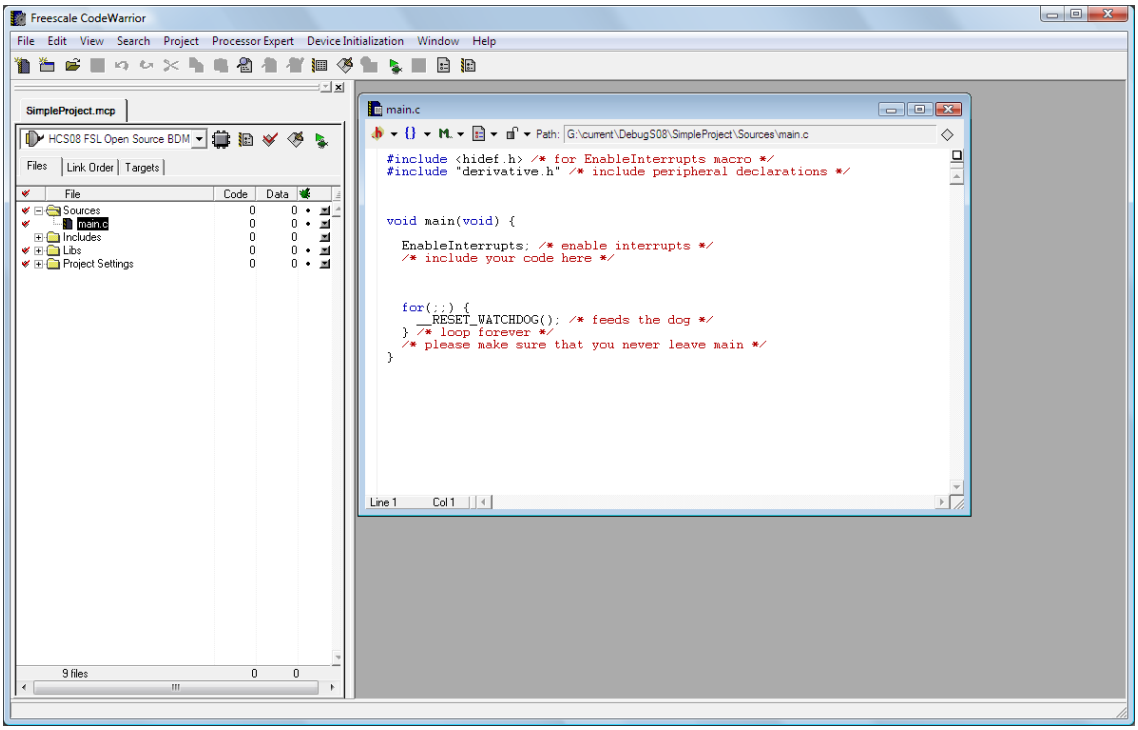


Figure 11: CodeWarrior in MDI mode

Some programmers really like to work this way, but other people prefer the ability to have independent windows that can be placed anywhere on the screen and are not constrained by the container.

If you want to turn MDI mode off you can do so through the `Preferences...` option in the CodeWarrior's `Edit` menu (Figure 12). You need to restart CodeWarrior for this change to take effect.

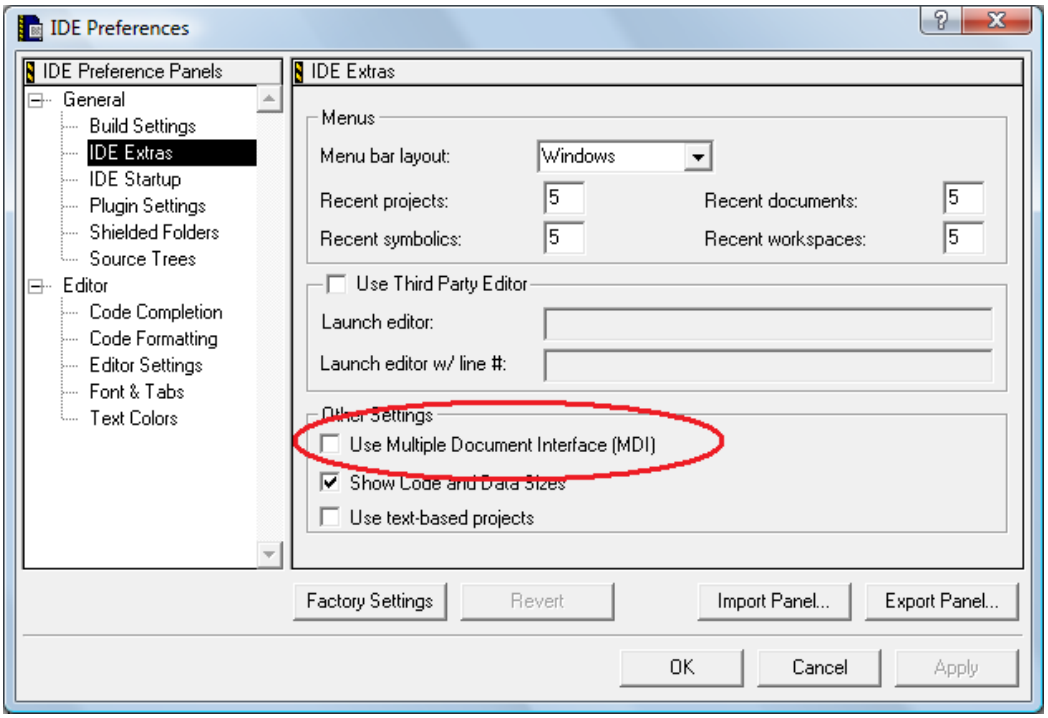


Figure 12: Turning off MDI mode

5.4. The CodeWarrior Project window

All development work in CodeWarrior requires the creation of a Project which specifies which source files and libraries are required to build a specific executable.

Figure 13 shows a screen shot of the demo project open in the CodeWarrior IDE, with the some of the most useful options labelled.

Note that you can access many more project-specific options through the Edit Project Settings button. For example you can configure compiler and linker options here.

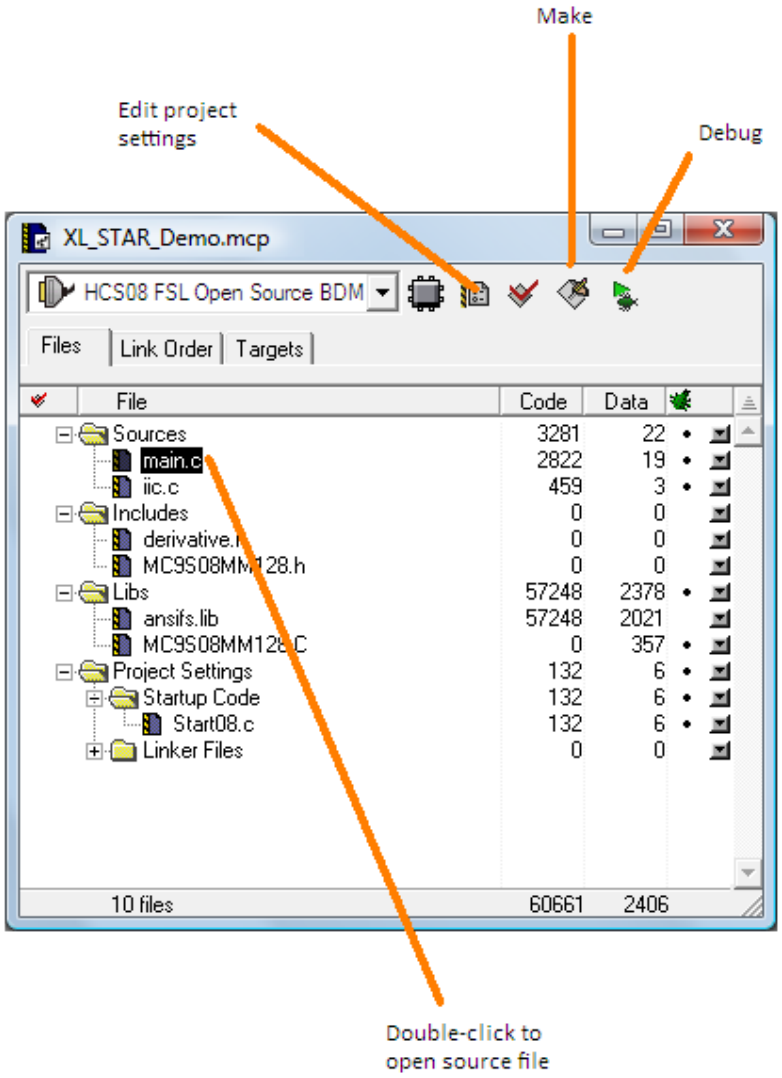


Figure 13: CodeWarrior Project window

When using CodeWarrior you will also see references to *Processor Expert* and *Device Initialization*. These are expert wizards aimed at helping you to write code for embedded targets. Although sometimes useful for new projects, you don't need to make use of them when working with the demo project supplied.

5.5. Editing and compiling a File

You can edit an individual source file by double-clicking on its name in the project window, which brings up an editor window like this (Figure 14):

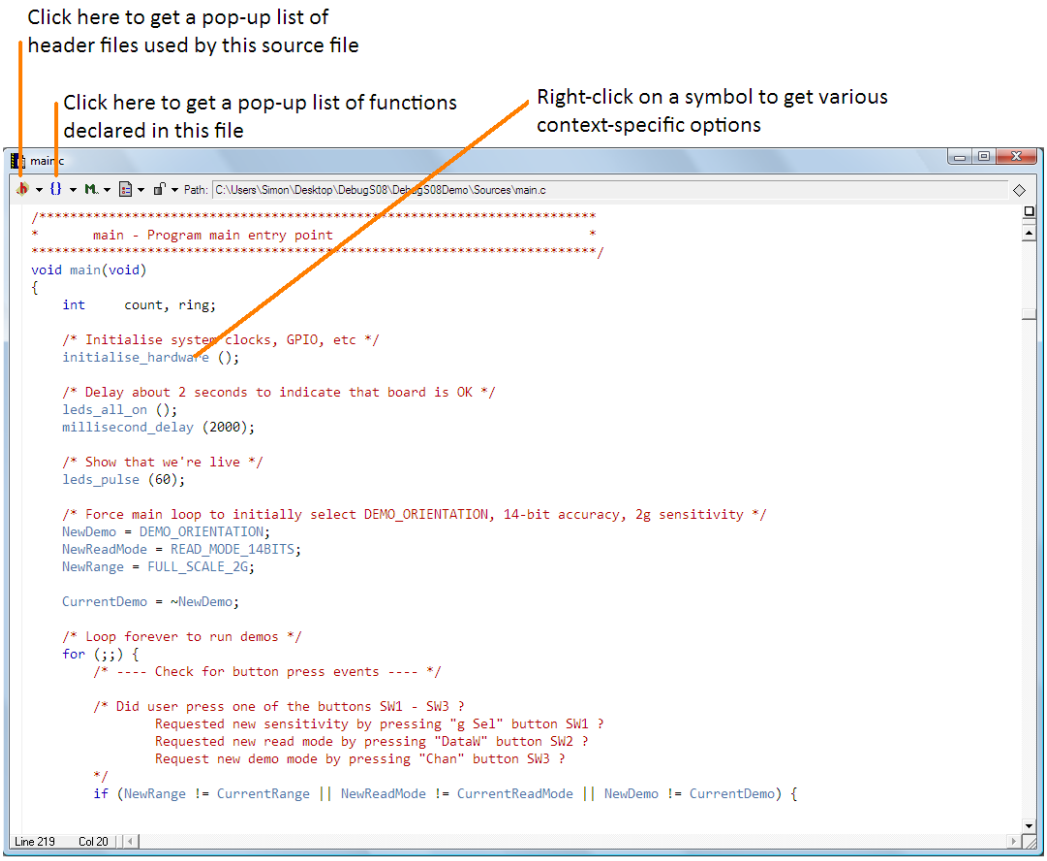


Figure 14: CodeWarrior Editor window

If you right-click within the editor window you will get a context-sensitive popup of useful options, including the option to compile the file.

If there are any error or warning messages when compiling a source file the error window is displayed (Figure 15):

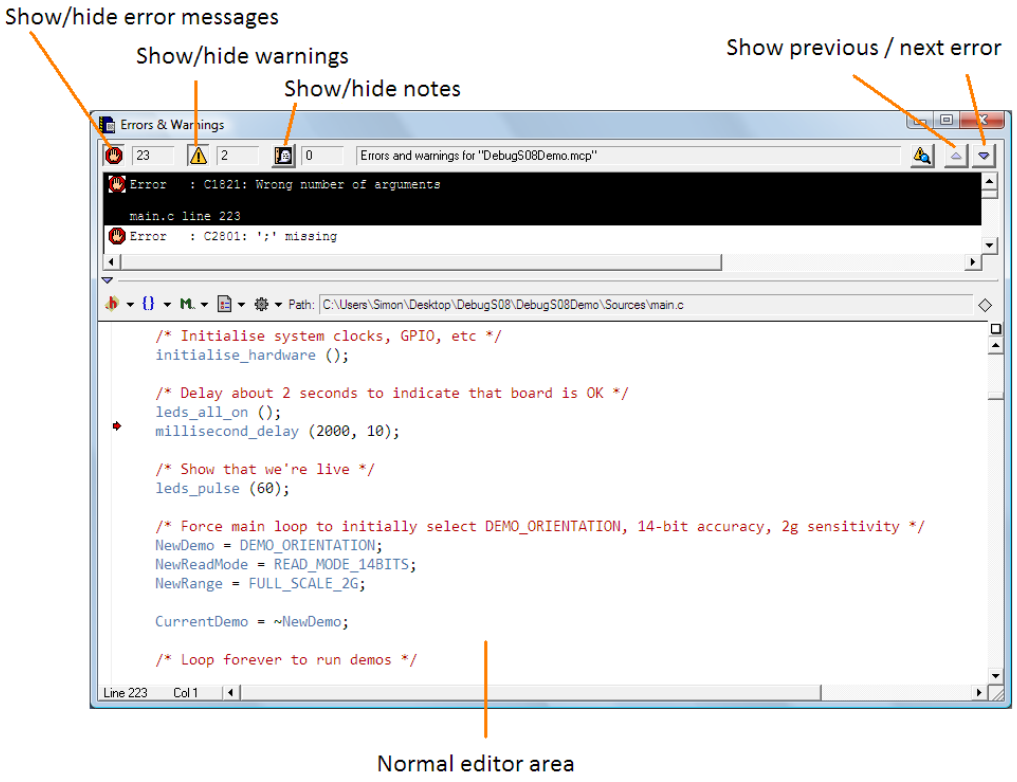


Figure 15: CodeWarrior Errors & Warnings window

Note that you can correct errors by typing directly within the error window, which behaves like a normal edit window.

5.6. Building code

The whole build process of compiling the individual source files and then linking them together with any libraries can be performed in a single step using the 'Make' button in the project window, or by selecting *Make* in the *Project* menu.

Compilation proceeds as described above and, if there are no compilation errors in the individual source files, the linker is invoked (Figure 16):

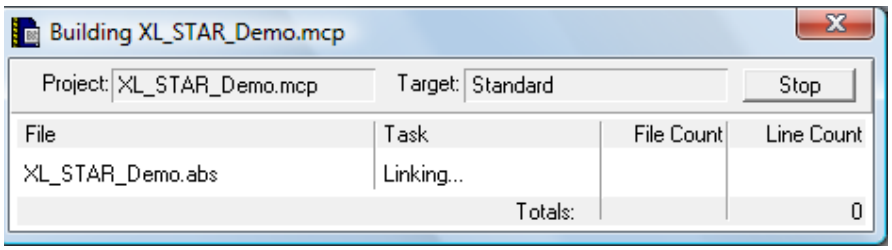


Figure 16: CodeWarrior Build Progress window

5.7. Debugging code

To download code to the XL_STAR board, first ensure that the board is connected to the PC using a USB cable - see Figure 17.

Note that the USB cable needs to be plugged into the USB port on the **underside** of the XL_STAR board as shown below. Don't be confused by the other USB port which is for use when developing applications for the S08MM128 which need their own USB interface.

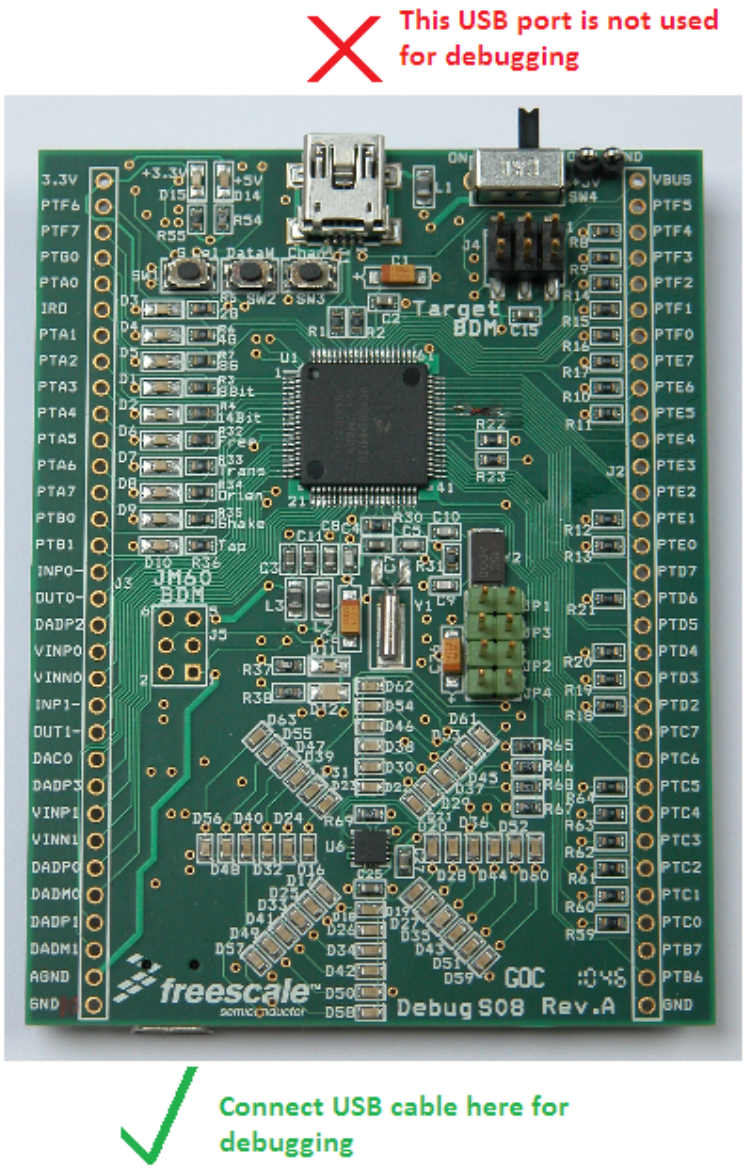


Figure 17: Connecting a USB cable for debugging

To download code to the board click the 'Debug' button in the project window, or select *Debug* from the *Project* menu. This will cause the Hiwave debugger to launch in a separate window.

The Hiwave debugger will automatically program the application into Flash memory on the XL_STAR board, as shown in Figure 18.

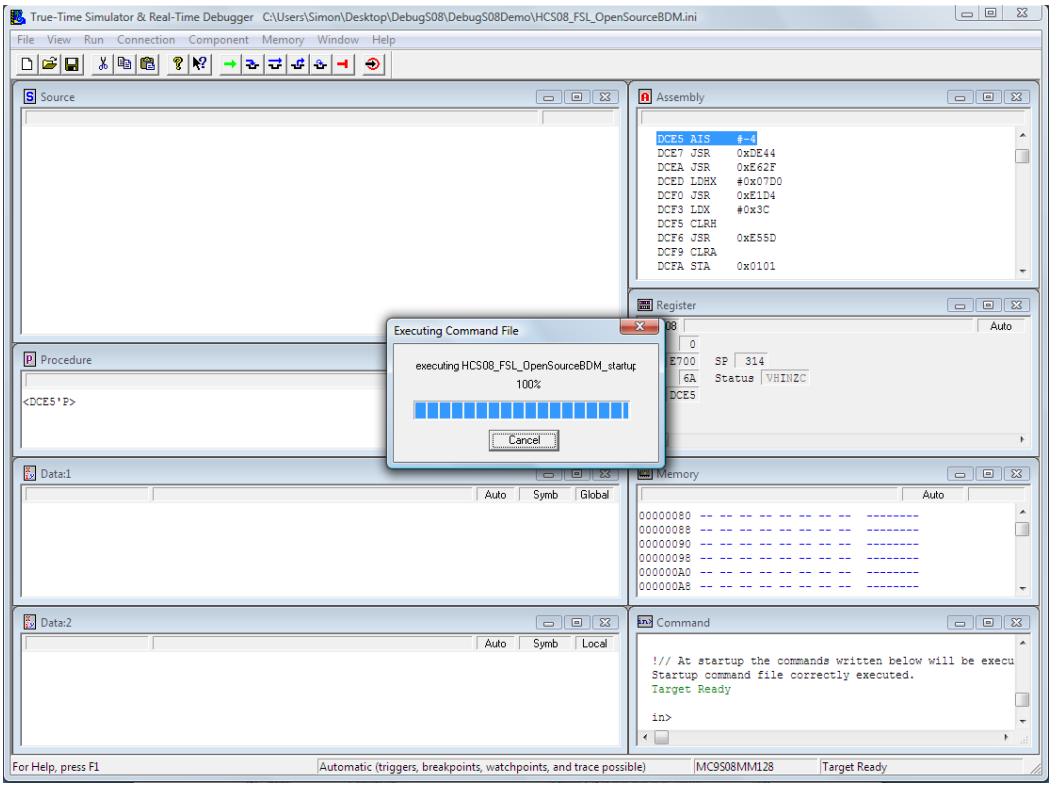


Figure 18: Hiwave Debugger with code download in progress

Following successful programming, the application will be executed automatically up until the point where the 'main' procedure is about to start (Figure 19 below). The point at which code execution halts can be changed through the *Configuration...* menu option in the debugger's *File* menu if required.

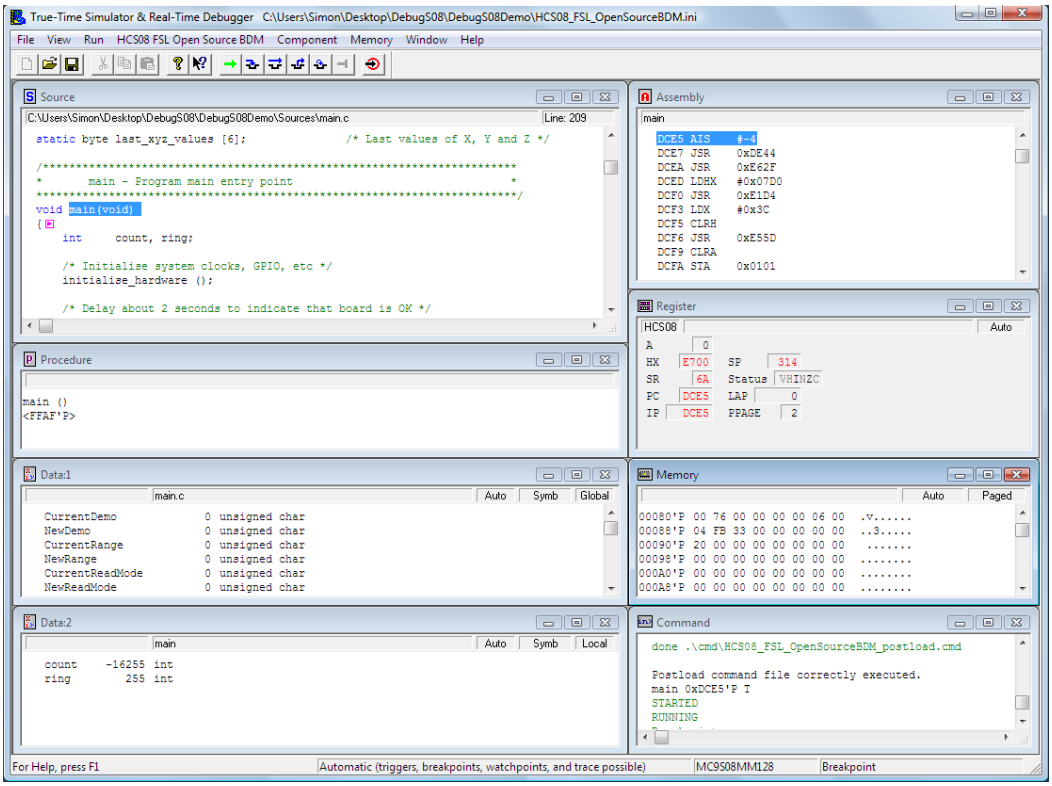


Figure 19: Hiwave Debugger

The close up in Figure 20 shows the most useful buttons in the debugger's tool bar:

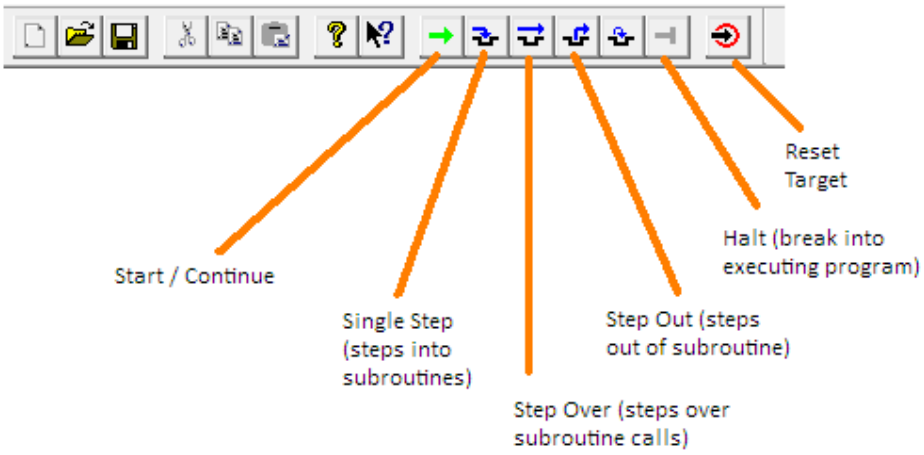


Figure 20: Hiwave debugger Toolbar

To set a breakpoint, use the debugger Source window to right-click on the line where you want the breakpoint to be set. A context-sensitive popup menu (Figure 21) includes *Set Breakpoint*

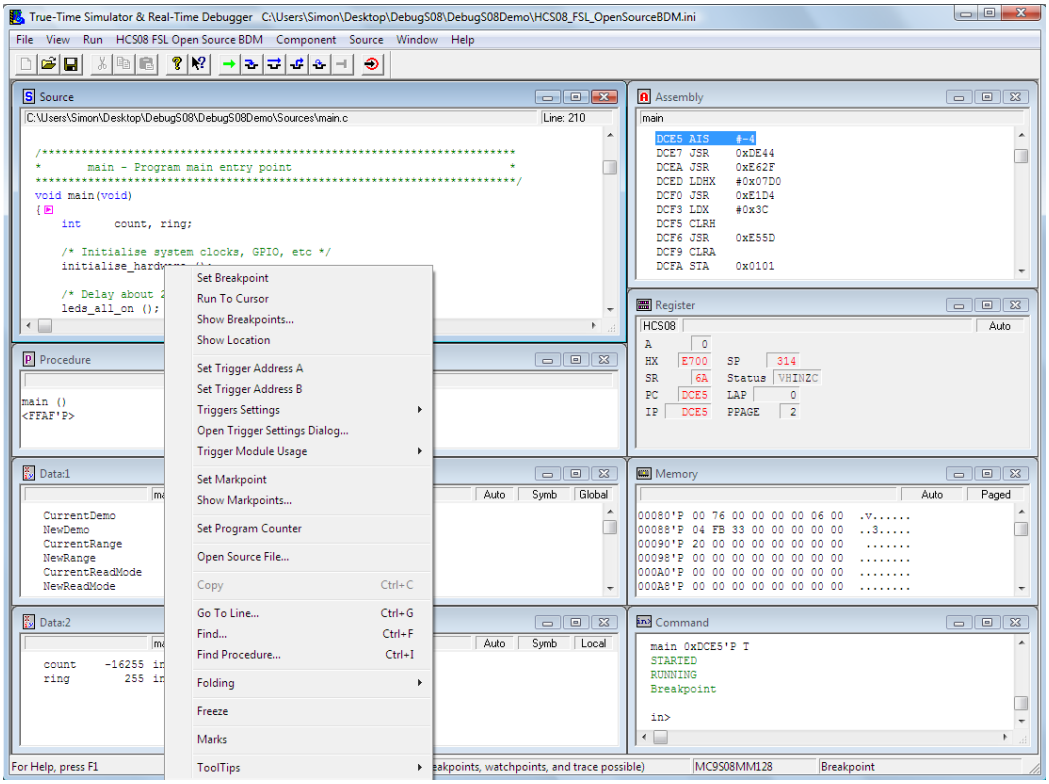
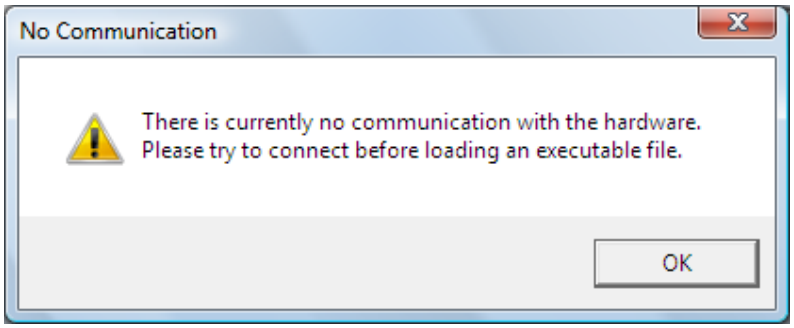


Figure 21: Context-sensitive menu available through right-click

5.8. Problems connecting to the board

In rare circumstances the debugger may have problems connecting to the board, e.g.



One possible cause is that Windows has failed to recognise that the board is connected to the PC.

The first thing to try is to disconnect the USB cable and then reconnect it. This should cause Windows to see the board. NB: Make sure that you are connecting to the USB port on the underside of the board, ***not*** the one near the On/Off switch on the top of the board.

If this doesn't work, try using another USB port on your PC. Also make sure that the board connects directly to the PC, not through an external USB hub.

You can verify that the XL_STAR board has been recognized by checking the Windows Device Manager whilst the board is connected. You should an entry for '*Open Source BDM-Debug Port (LibUSB-Win32 Devices)*' or similar see Figure 22:

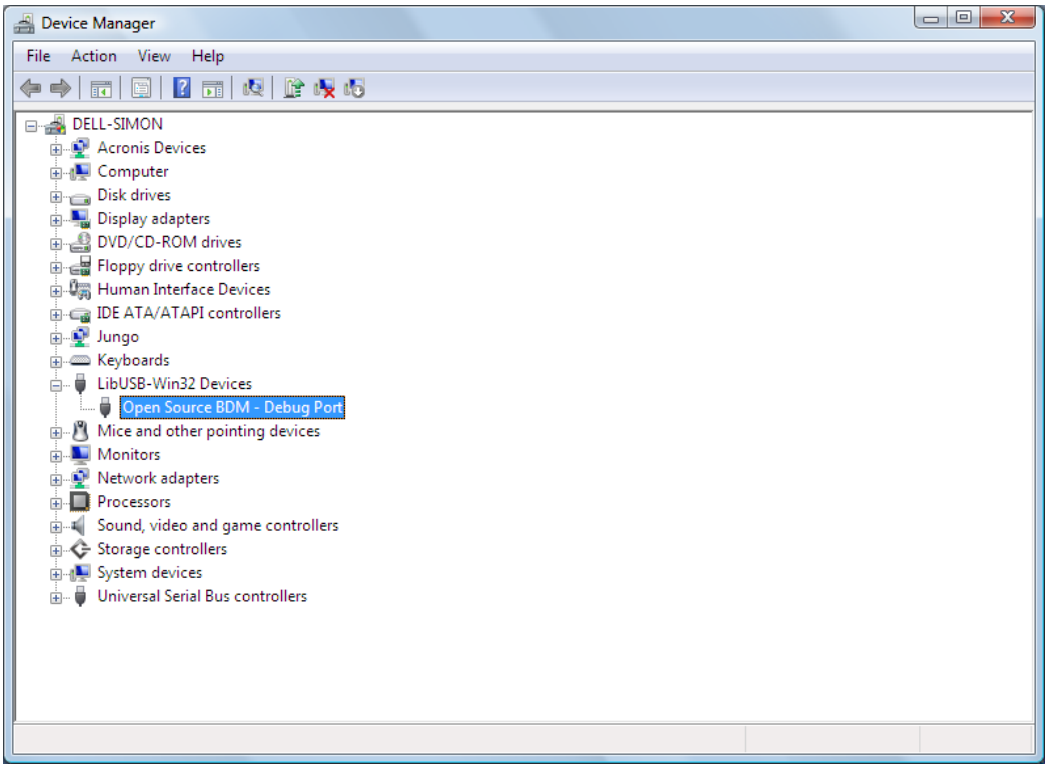


Figure 22: 'Open Source BDM-Debug Port' in Windows Device Manager

If Windows can detect that the XL_STAR board is attached, but warns that it cannot locate the appropriate drivers, see Section 4.3.

5.9. Problems debugging at high clock speeds

You may find that debugging works perfectly with the example project, but that you encounter problems when writing your own code which runs the MM128 MCU at a higher bus clock speed.

The bus clock provides the clock source for the MM128's Background Debug Controller (BDC), so if the bus clock is too fast the debug information sent on the BGND pin will be too fast for the OSBDM module on the bottom side of the board to handle.

For applications which require a high bus clock speed (e.g. 24MHz for the USB port) you will need to use an external debug device, such as a P & E USB Multilink Interface, connected to the BDM header labelled 'Target BDM'.

6. Updating the OSBDM-JM60 Firmware on the board

WARNING

You will not normally need to update the OSBDM firmware on the XL_STAR board, and doing so wrongly can render the board unusable.

To fix this situation you would need to be able to reprogram the JM60 using a separate programming device like the P&E Micro HCS08 USB Multilink cable.

The XL_STAR board includes two microcontrollers. On the top side of the board is the main MM128 MCU. This is the processor you will normally target when writing new programs. You update the code for this MCU every time you change your program, and programming mistakes shouldn't cause lasting problems.

The other processor, on the underside, is the JM60 MCU. This runs firmware which allows the CodeWarrior debugger to control the MM128 via its Background Debug Mode (BDM) module.

The firmware is known as Open Source BDM, or OSBDM. It is responsible for USB communications with the debugger on the PC, controlling the MM128 via the BGND debug pin, and configuring the on-board battery charger. It also includes a Bootloader to allow you to update the running code with a newer OSBDM version.

You do not need to update the OSBDM firmware in normal operation, unless Freescale issues an update to the code.

If you do need to update the OSBDM firmware, **make sure that you use a version that's specifically intended for the XL_STAR board**. The following steps should help you.

1. Disconnect the board from the PC.

2. Install a jumper in position JP3 to select Bootloader mode.

3. With the On/Off switch set to the Off position, connect the board to the PC using a USB cable plugged into the board's normal debugging port on the underside of the board.

4. If you wish, you can check that the board has been recognised by the PC by using the Windows Device Manager:

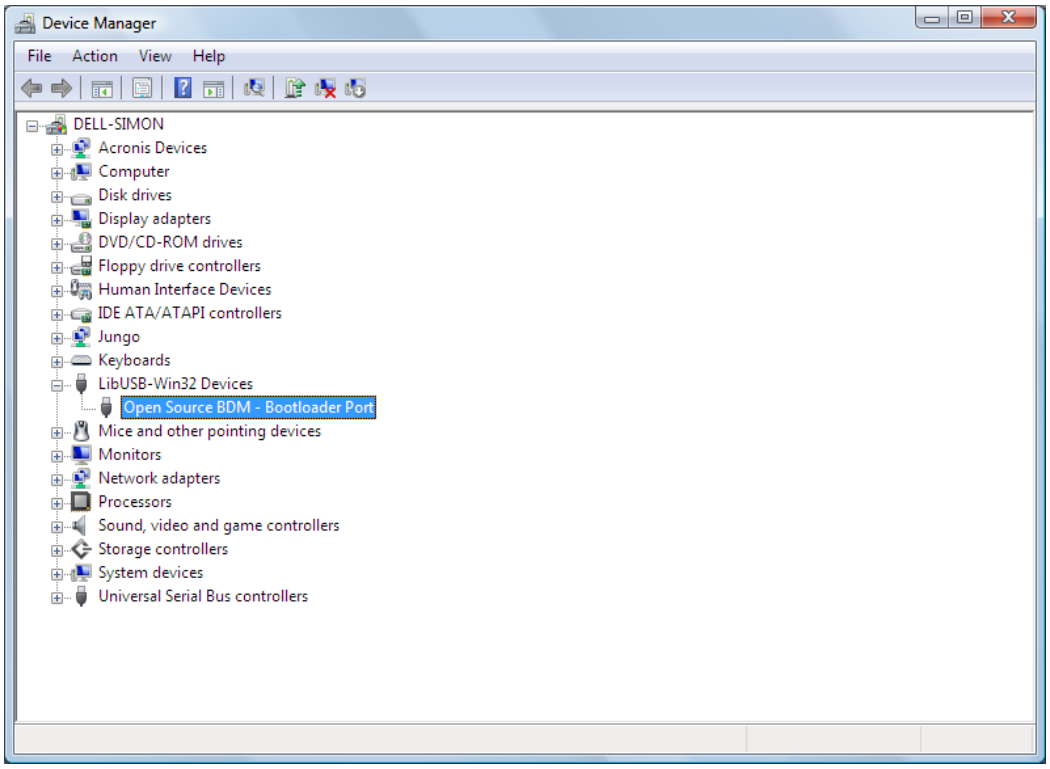


Figure 23: The board shows up as *Open Source BDM - Bootloader Port* in Windows Device Manager

5. Use the *PE Firmware Updater* program version 1.01, available from Freescale's website (see Figure 24):

- Select *OSBDM/OSJTAG* as the hardware type
- Select *Embedded OSBDM/OSJTAG Device - Bootloader Mode* as the device
- Select *S08* as the architecture to support
- Select the *Choose Firmware Update File or S-Record* option. NB: Don't choose the *Automatic* option.
- Select the file containing the new OSBDM firmware
- Press the *Update Firmware* button. You should get a progress dialog similar to Figure 25

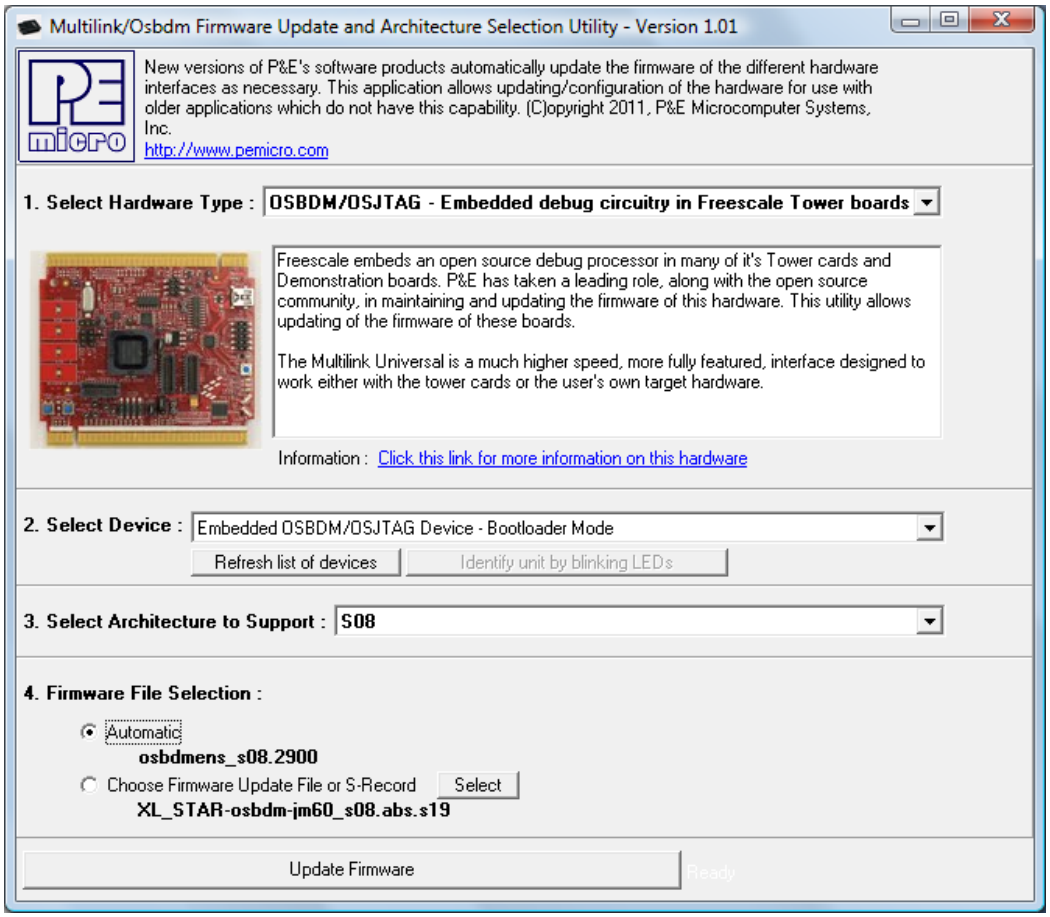


Figure 24: PE Firmware Updater, used to update the bootloader

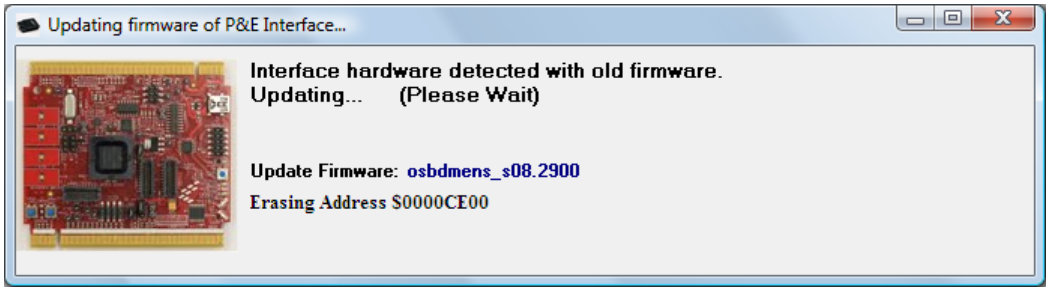


Figure 25: OSBDM update in progress

7. When the update completes you will see a warning similar to Figure 26. Unplug the USB cable, remove the JP3 jumper, and finally reconnect the USB cable to return to normal operation.

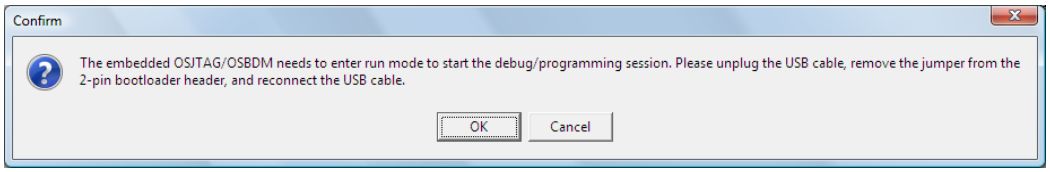


Figure 26: Last stage of updating OSBDM firmware

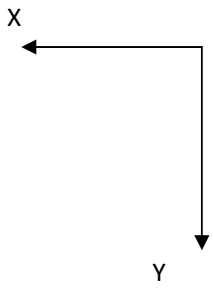
7. Understanding the source code of the Demo program

You may want to explore the source code of the demo program and develop it further. The following remarks may be helpful when looking at the source code.

7.1. Orientation

The MMA8451Q is a three axis accelerometer. It can measure the acceleration in three independent directions X, Y and Z.

Holding the board so that the On/Off switch is at the top and the word 'Freescale' is at the bottom, the accelerometer is oriented as follows:



Z axis out of page towards reader

This is referred to in the MMA8451Q documentation as the Portrait Down (PD) orientation mode. In this orientation, the reading for the Y axis would be 1g, and X and Z would be 0g.

7.2. Lighting the orange LEDs

The orange LEDs are arranged in an 8-pointed star as shown in Figure 27. Each of the 8 'spokes' of the star is connected to a GPIO pin. For example, LEDs D62, D54, D46, D38, D30 and D22 are all connected to Port D Pin 4 (PTD4)

Similarly, each of the 6 concentric 'circles' is connected to another GPIO pin. For example the outer circle D56-D63 is connected to Port C Pin 5 (PTC5).

Thus to illuminate D62 you would drive PTD4 low and PTC5 high.

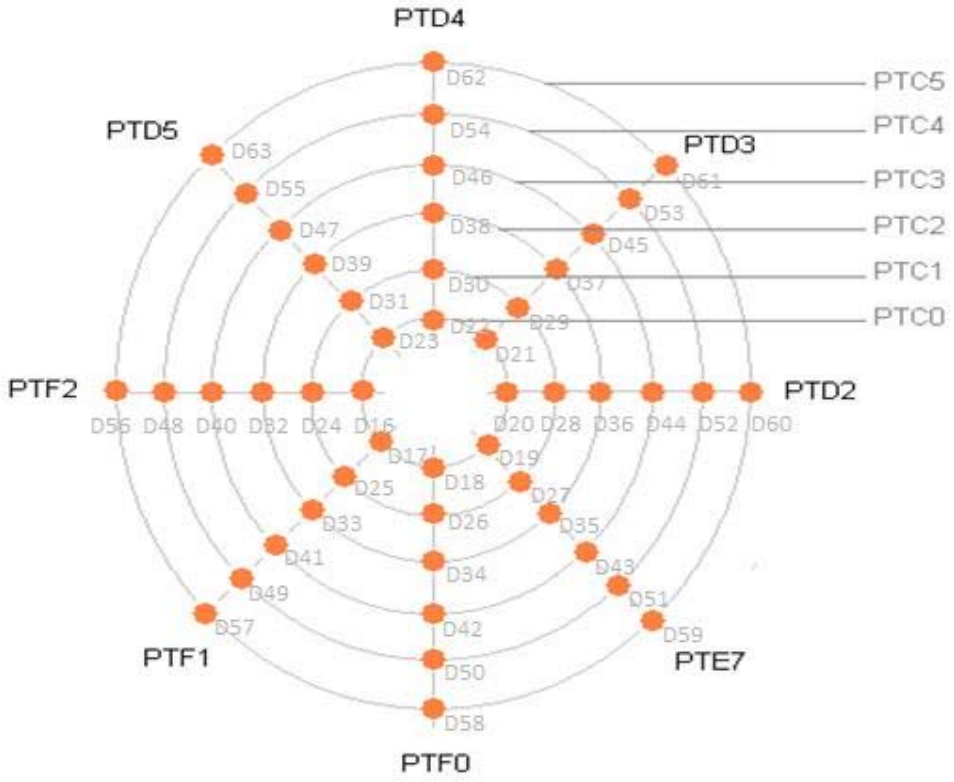


Figure 27: Relationship between GPIO pins and LEDs

For one of the demo modes - the Orientation demo - the LEDs are driven in a more sophisticated way. This takes advantage of the fact that each of the GPIO pins connected to a 'spoke' can be reconfigured so that it is controlled by one of the MCU's Timer/PWM (TPM) modules.

GPIO pin	TPM module/channel
PTD5	TPM1CH3
PTD4	TPM1CH2
PTD3	TPM1CH1
PTD2	TPM1CH0
PTE7	TPM2CH3
PTF0	TPM2CH2
PTF1	TPM2CH1
PTF2	TPM2CH0

By driving a pin with a PWM signal which has a duty cycle of less than 100%, the LED can be illuminated less intensely. The orientation demo uses this effect to illustrate the direction in which gravity is pulling relative to the board's orientation.

7.3. Communicating with the Accelerometer

The MCU is connected to the accelerometer using an industry-standard connection known as Inter-Integrated Circuit, or IIC.

The accelerometer's IIC address is 0x1D, determined by a pull-up resistor on the board.

Typically IIC is used to read or write a single 8-bit register location within the accelerometer - for example software reads the PULSE_SRC register to get information about the latest tap/pulse event that the accelerometer has detected.

In the case of the accelerometer's data registers, which are used to obtain the most recent readings on the X, Y and Z axes, the full readings are 14 bits each, split into an MSB/LSB pair of registers. This means that the software must read a total of six registers to obtain a full XYZ readout. This is done with a single IIC command capable of transferring multiple sequential bytes in one go.

Alternatively the accelerometer can be reconfigured to only return the MSB part of each X, Y and Z value - i.e. the most significant 8 bits. This means that the IIC command must only read three bytes, making it faster for applications which don't require full 14-bit accuracy.

To demonstrate how to write code using either 8-bit or 14-bit sensitivity, the demo program uses the SW2 switch (labelled DataW) to allow the user to reconfigure the accelerometer for either mode.

7.4. Interrupt Handling

In order to conserve power, both the accelerometer and the MCU are configured to switch to low-power sleep modes when no events of interest are occurring. The MCU is woken from sleep mode when an interrupt occurs, either from the accelerometer or caused by one of the push buttons SW1 - SW3.

In the case of the accelerometer, the interrupt is signalled when the accelerometer drives its `INT1` pin low. This is connected to the MCU's Keyboard Interrupt KBI2P4, which is configured to detect a falling edge. When `INT1` goes low, the falling edge triggers the interrupt to the MCU.

Note that although `INT1` remains low until software has cleared the source of the interrupt by reading data from the accelerometer, this will not cause a further interrupt to the MCU because no further edges occur. This means that the interrupt service routine can be kept as short as possible, being used only to wake the MCU from sleep.

7.5. Low Power mode

The demo software is designed so that the accelerometer will detect when there has been no event of interest for 20 seconds or more. This will depend on the demo mode selected; for example in freefall detection mode, only dropping the board will cause an event.

After 20 seconds of inactivity the accelerometer will switch to a low power sleep mode. This causes an interrupt, allowing the MCU to be put into a low power mode too.

In the case of the demo software, the board can be woken again by picking it up. To achieve this, the accelerometer is reconfigured at the moment when the MCU is about to sleep. In the new configuration the accelerometer runs in Sleep mode and is set to detect any transients which would indicate that the board has been moved.

Once the accelerometer detects movement it will wake up, and also signal to wake the MCU. The MCU then reconfigures the accelerometer, restoring its normal wake-mode configuration.

7.6. Floating Point code

The sample program is unusual because it makes use of floating point data types like `double`, and maths support routines like `atan2`. Most embedded software only uses integer data types.

The CodeWarrior compiler includes complete support for floating point data types, implemented in software (The HCS08 family of microcontrollers don't support floating point instructions in hardware).

In order to link to floating point support routines, the CodeWarrior project includes a library `ansifs.lib`. For integer-only routines this can be replaced by `ansiis.lib` which may result in a slightly smaller program size.

7.7. Freescale Application Notes for the MMA8451Q Accelerometer

You may find the following Freescale application notes useful when writing code for the accelerometer, all of which are available from the Freescale website – www.freescale.com:

- AN4068, Embedded Orientation Detection Using the MMA8451, 2, 3Q
- AN4069, Offset Calibration of the MMA8451, 2, 3Q
- AN4070, Motion and Freefall Detection Using the MMA8451, 2, 3Q
- AN4071, High Pass Data and Functions Using the MMA8451, 2,3Q
- AN4072, MMA8451, 2, 3Q Single/Double and Directional Tap Detection

- AN4073, Using the 32 Sample First In First Out (FIFO) in the MMA8451Q
- AN4074, Auto-Wake/Sleep Using the MMA8451, 2, 3Q
- AN4075, How Many Bits are Enough? The Trade-off Between High Resolution and Low Power Using Oversampling Modes
- AN4076, Data Manipulation and Basic Settings of the MMA8451, 2, 3Q
- AN4077, MMA8451, 2, 3Q Design Checklist and Board Mounting Guidelines

8. Writing your own code for the XL_STAR Board

In this final section we will walk through the steps for creating your own CodeWarrior project for the XL_STAR board. In brief the steps are as follows:

- Create a new project
- Write some code to do something interesting with the XL_STAR board
- Compile and link
- Download code to the board
- Debug and test

8.1. Creating a new project

1) When CodeWarrior is launched you should see the following dialog (Figure 28) which includes the option to create a new project. Alternatively you can use the `New Project...` option in CodeWarrior's `File` menu.

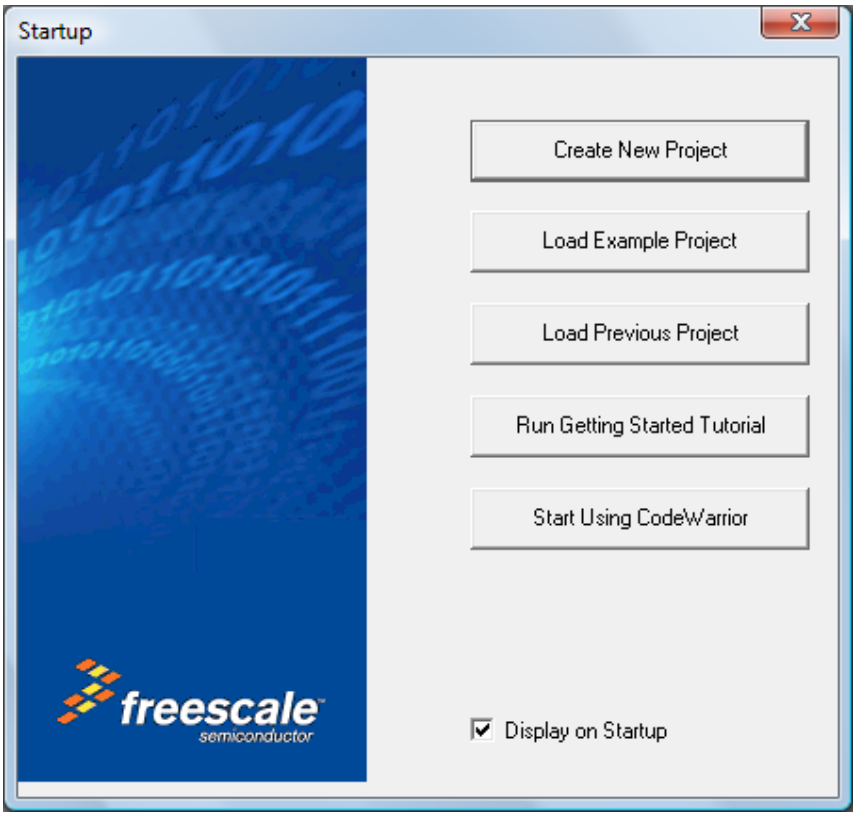


Figure 28: CodeWarrior startup dialog

2) This will launch a New Project wizard (). Select **MC9S08MM128** as the derivative and **HCS08 FSL Open Source BDM** as the connection method:

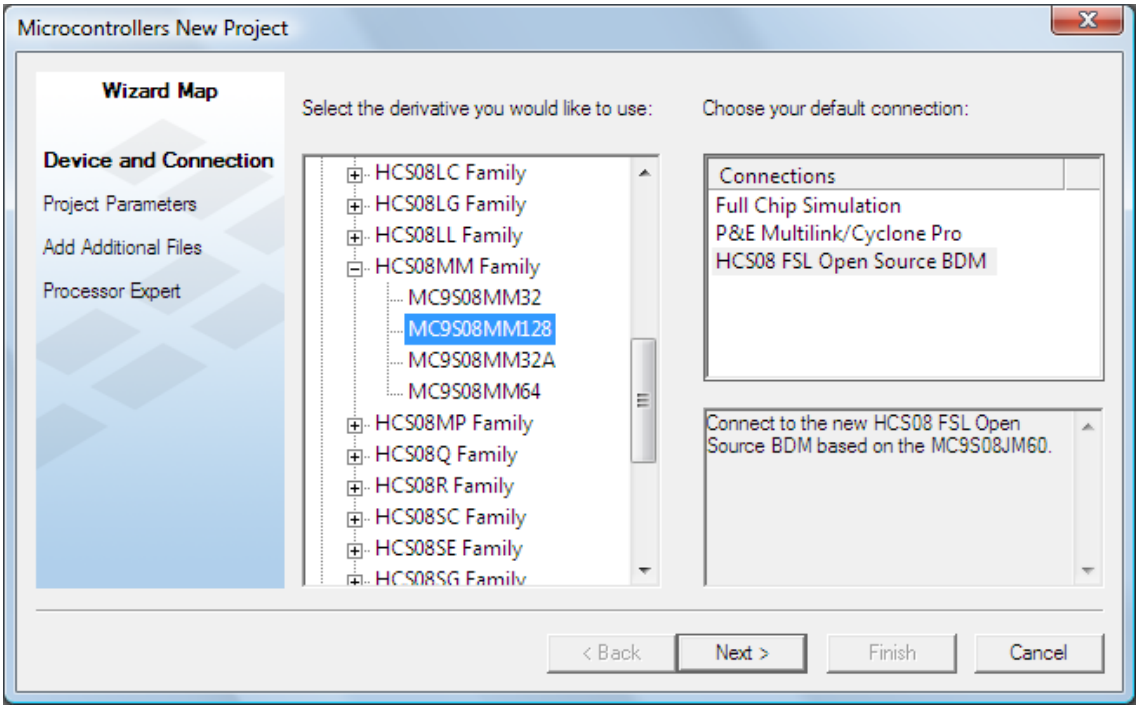


Figure 29: CodeWarrior New Project wizard

3) In the next step (Figure 30), specify the name to give to the CodeWarrior project and its location, using the `Set . . .` button.

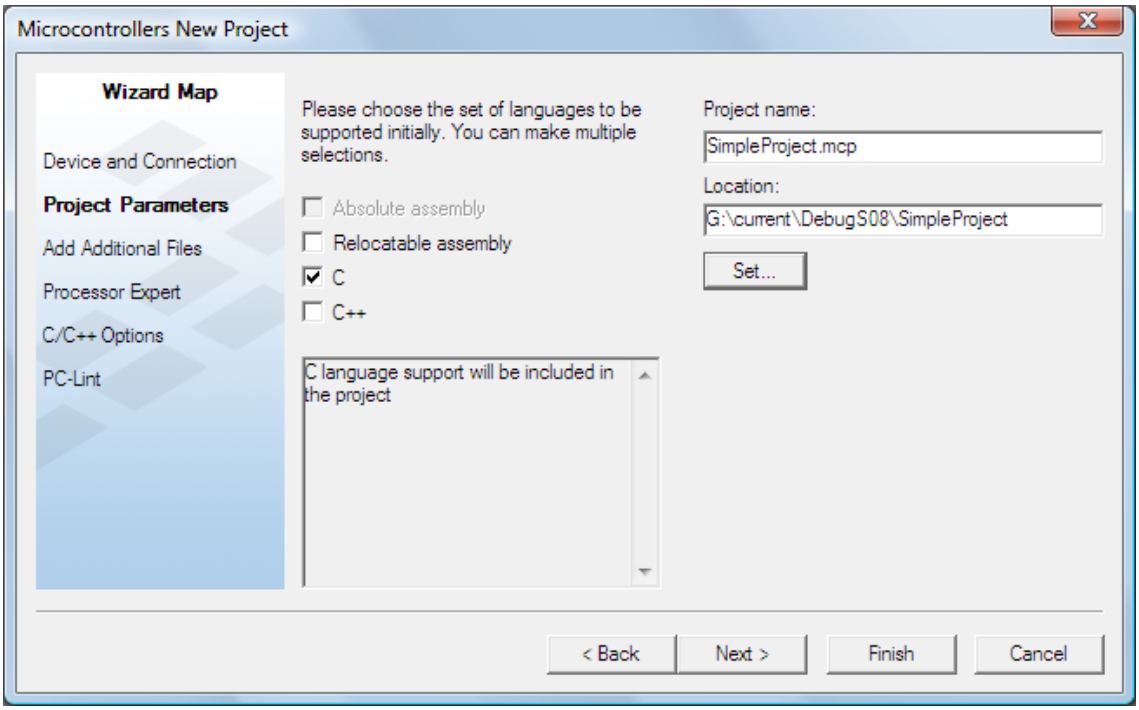


Figure 30: Specifying the project name and location

Make sure that the directory where the new project is placed is writeable - for example, place it in `Documents` rather than `C:\Program Files` (Figure 31). Note that CodeWarrior will automatically add the file extension `.mcp` to the name of the project file.

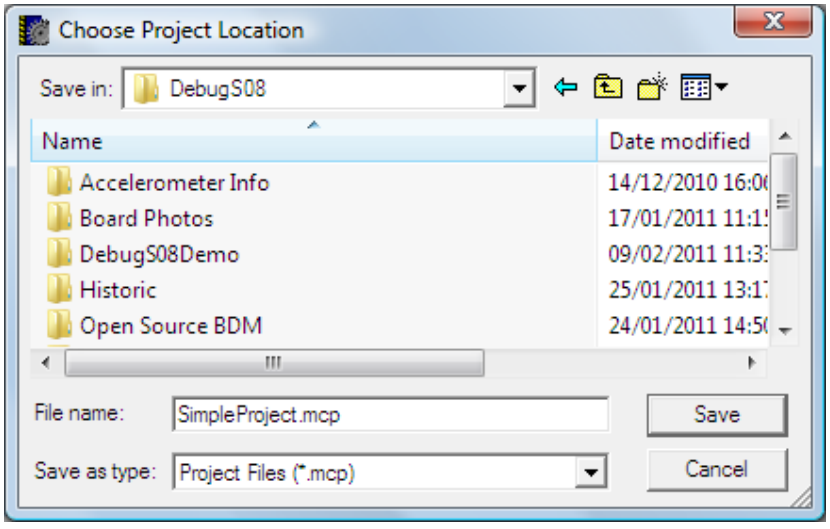


Figure 31: Use a writeable directory for the new project

4) In the next step (Figure 32) you have the option to add existing files to the new project. We will not add any for this demonstration.

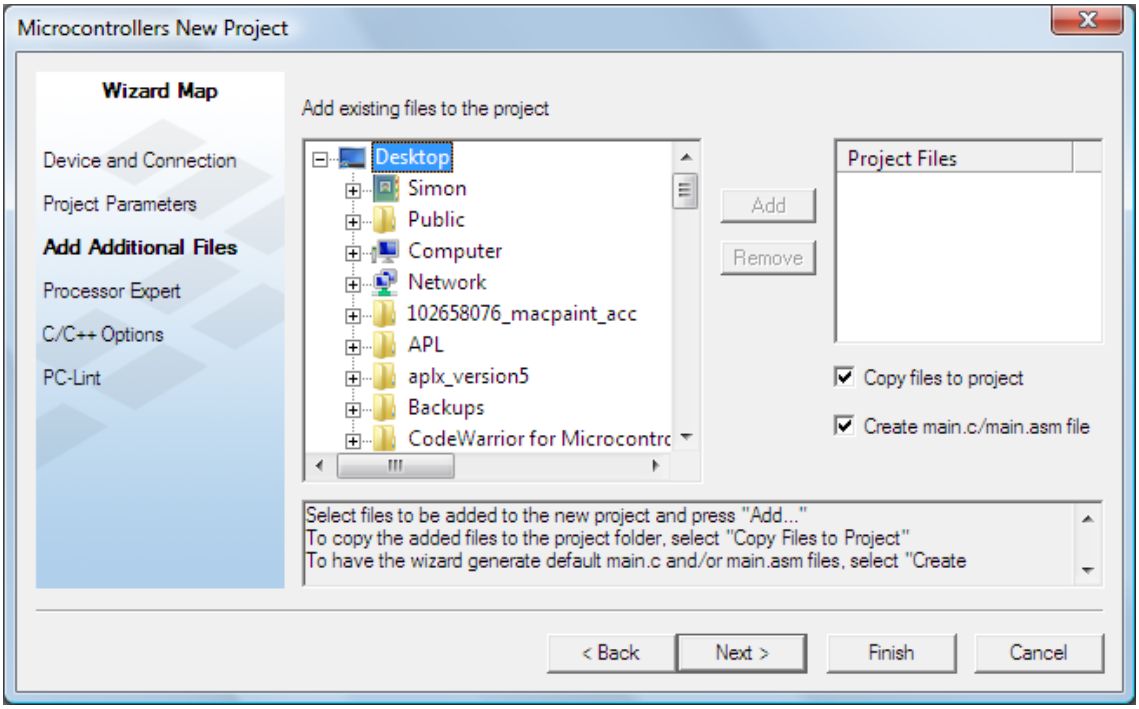


Figure 32: Don't add any existing files

5) The demonstration does **not** make use of the Processor Expert or Device Initialization wizards (Figure 33):

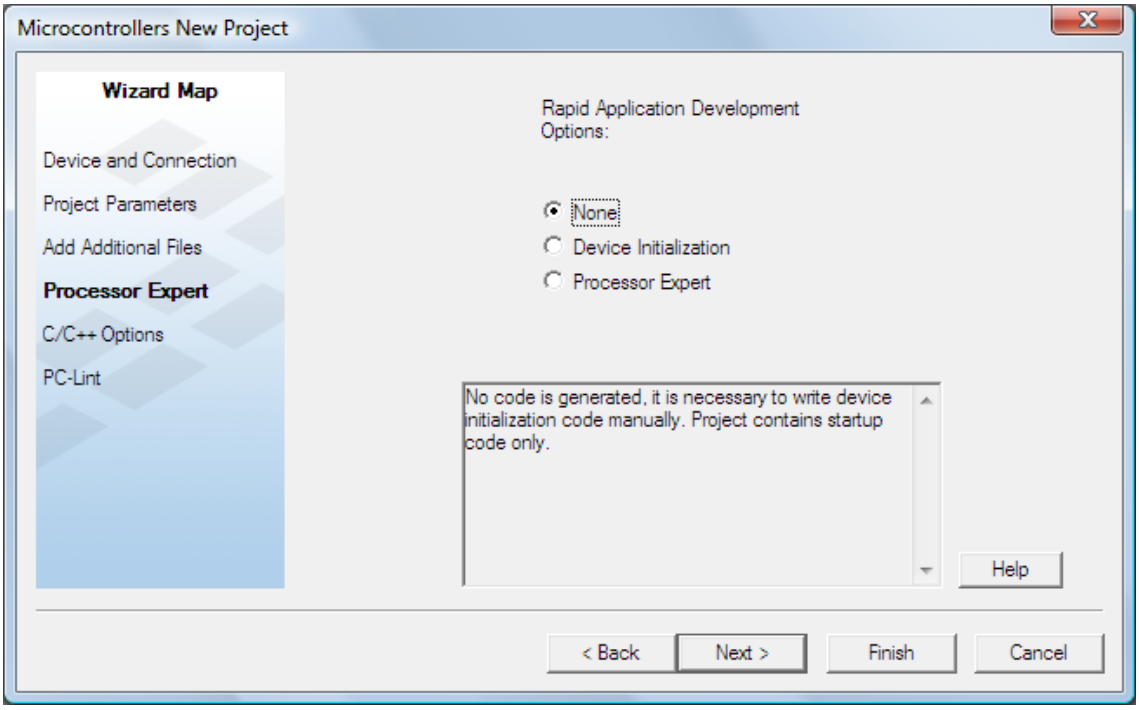


Figure 33: Don't use a Rapid Application Development wizard

6) Accept the default C/C++ options (Figure 34):

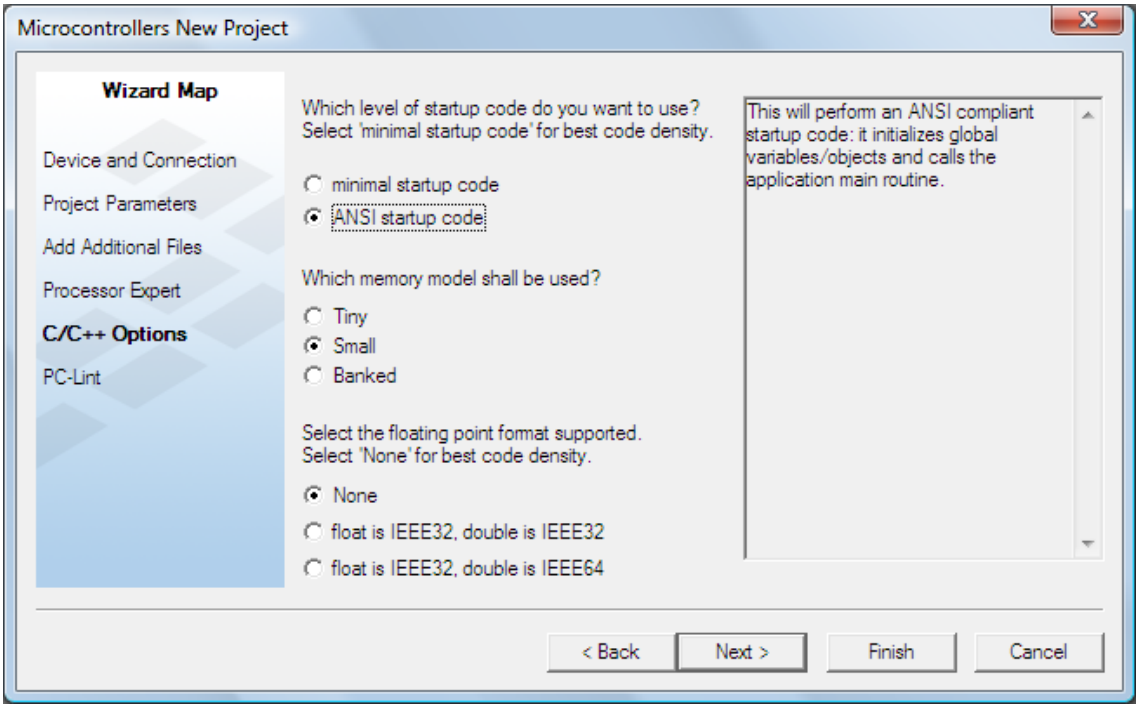


Figure 34: Use the default C/C++ options

7) Don’t enable the use of the PC-lint™ source code checker (Figure 35):

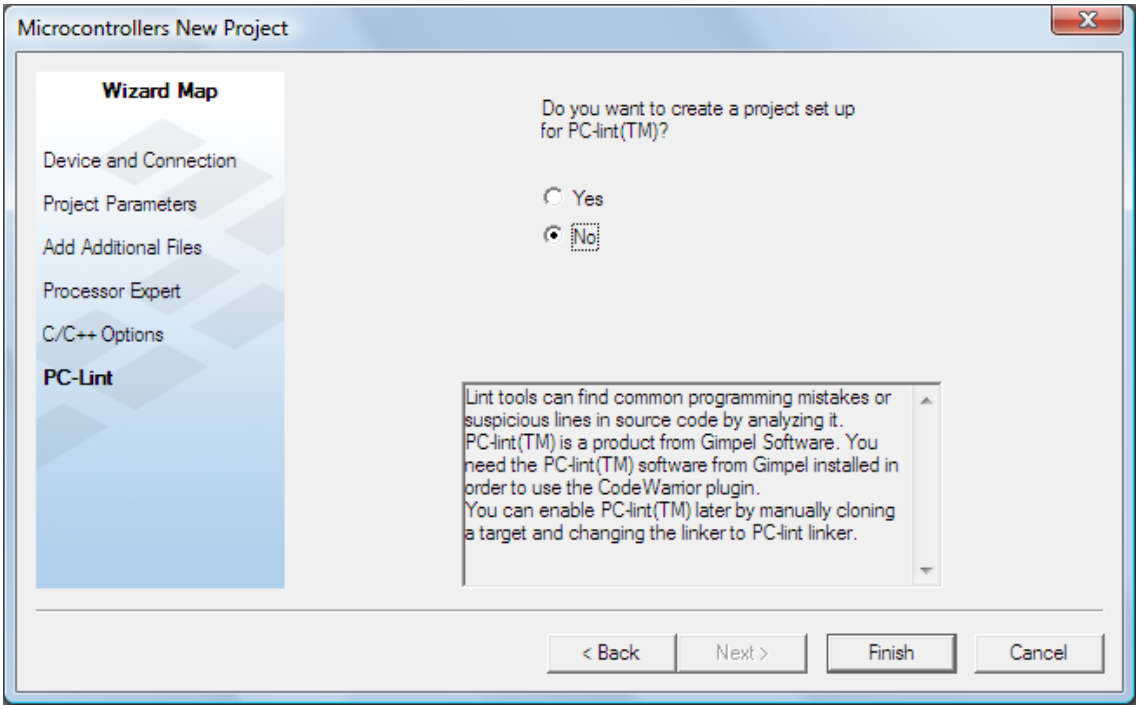


Figure 35: Don't use PC-lint™ for this project

8.2. Finding your way around the CodeWarrior Project Window

The steps outlined in Section 8.1 will create a CodeWarrior project suitable for use with the XL_STAR board, and open the main project window (Figure 36)

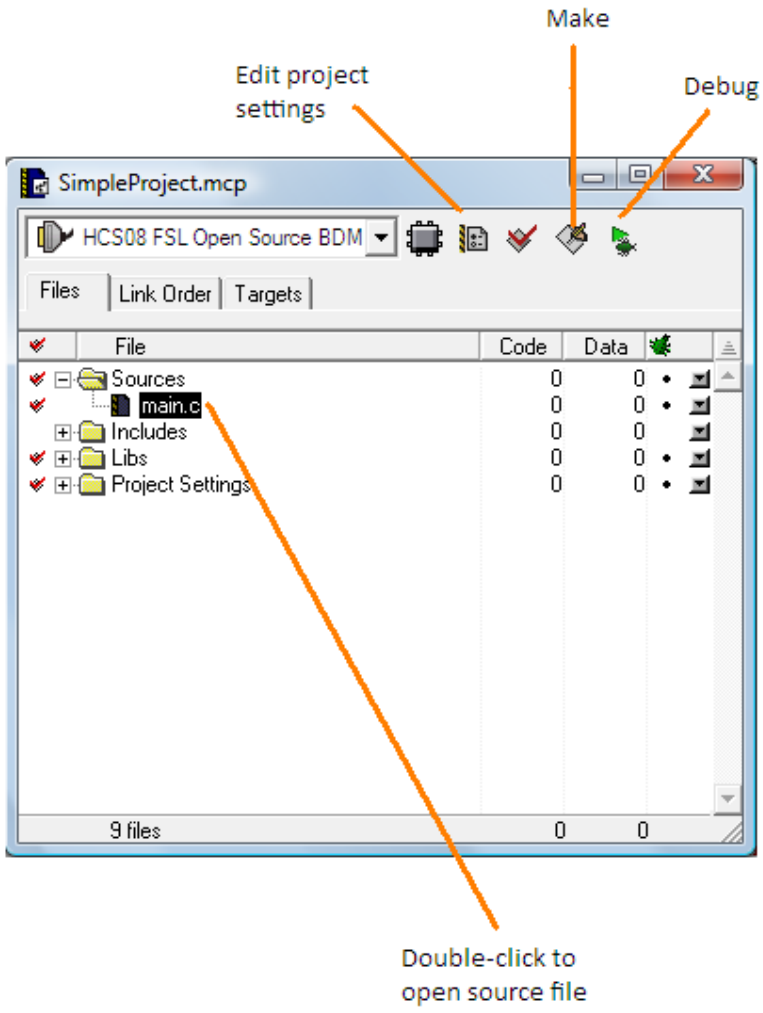


Figure 36: CodeWarrior Project Window

8.3. Writing some simple source code

At the moment, our sample application code doesn't actually do anything interesting - although you could build it and download it to the XL_STAR board if you wanted to.

Shown below is the listing of a simple replacement for the code in the file `main.c` which does the following:

- Turns off the COP Watchdog timer.

  The Computer-Operating-Properly (COP) watchdog timer is useful for applications which need to monitor whether software is functioning correctly. When the COP watchdog is enabled the application must service it periodically using a specific

sequence. If the servicing does not occur within a set time period, the COP will timeout and cause the processor to reboot.

To keep our example as simple as possible the COP watchdog is disabled.

- Uses the default internal clock.

  The XL_STAR board includes an external 16MHz crystal which can be used to give an accurate processor clock. To keep this demo as simple as possible, the external crystal is not used, and the processor remains in its boot-up clocking mode.

  For an example of reconfiguring the processor to use the external crystal, see the accelerometer demo code.

  Note that the OSBDM module can only be used for debugging if the MM128 processor's bus clock is slow enough (See Section 5.9)

- Initialises the GPIO pins connected to the orange LEDs so that they are all outputs (See Section 7.2)

- Enters an endless loop to flash the LEDs, using a pattern in which the spokes light in turn, giving the effect of a rotating finger of light.

```
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */


void main(void)
{
    unsigned char     spoke;
    unsigned short    i;

    /* Configure write-once System Options Register (SOPT)
           STOPE=1   : Stop Mode enabled
           COPE=0    : Disable watchdog
           BKGDPE=1  : Background Debug Mode Pin Enable
    */
    SOPT1 = 0x23;

    /* Configure all GPIO pins connected to orange LEDs
       as outputs
    */
    PTCDD |= 0x3f;
    PTDDD |= 0x3c;
    PTEDD |= 0x80;
    PTFDD |= 0xc7;

    /* Port C pins must be high to turn on LEDs */
    PTCD |= 0x3f;
```

```
    /* Loop forever */
    for (;;) {


        for (spoke = 0; spoke < 8; spoke++) {

            /* Delay */
            for (i = 0; i < 20000; i++)
                ;

            /* All LEDs off */
            PTDD |= 0x3c;
            PTED |= 0x80;
            PTFD |= 0xc7;


            /* Turn on a spoke of LEDs */
            switch (spoke) {
            case 0:
                PTDD &= ~ (1 << 4);     /* Spoke D62 on */
                break;
            case 1:
                PTDD &= ~ (1 << 3);     /* Spoke D61 on */
                break;
            case 2:
                PTDD &= ~ (1 << 2);     /* Spoke D60 on */
                break;
            case 3:
                PTED &= ~ (1 << 7);     /* Spoke D59 on */
                break;
            case 4:
                PTFD &= ~ (1 << 0);     /* Spoke D58 on */
                break;
            case 5:
                PTFD &= ~ (1 << 1);     /* Spoke D57 on */
                break;
            case 6:
                PTFD &= ~ (1 << 2);     /* Spoke D56 on */
                break;
            case 7:
                PTDD &= ~ (1 << 5);     /* Spoke D63 on */
                break;

            }
        }
    }
}
```

8.4. Building the new project

To build the project, use the **Make** button in the CodeWarrior project window (Figure 36).

8.5. Downloading the project to the XL_STAR board

To test the project, first make sure that a USB cable is connected between your PC and the XL_STAR board, as described in Section 5.7. Note that the USB cable connects to the port on the underside of the board, **not** the one near the On/Off switch.

Use the **Debug** button in the CodeWarrior project window (Figure 36) to launch the Hiwave debugger in a new window. This will program the new application into Flash memory on the MM128 MCU, erasing any existing application (Figure 37)

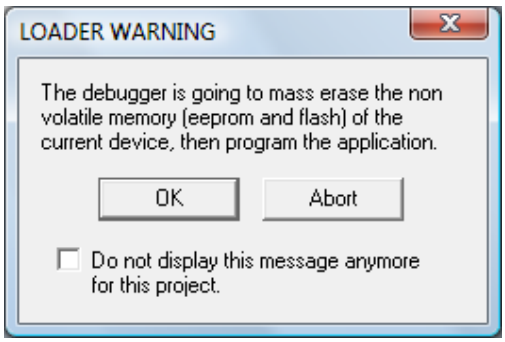


Figure 37: Loading code to the MM128 MCU

8.6. Running the code

Hit the green **Start** arrow in the Hiwave debugger window to start the program executing (Figure 38). You should see a rotating pattern on the board's orange LEDs.

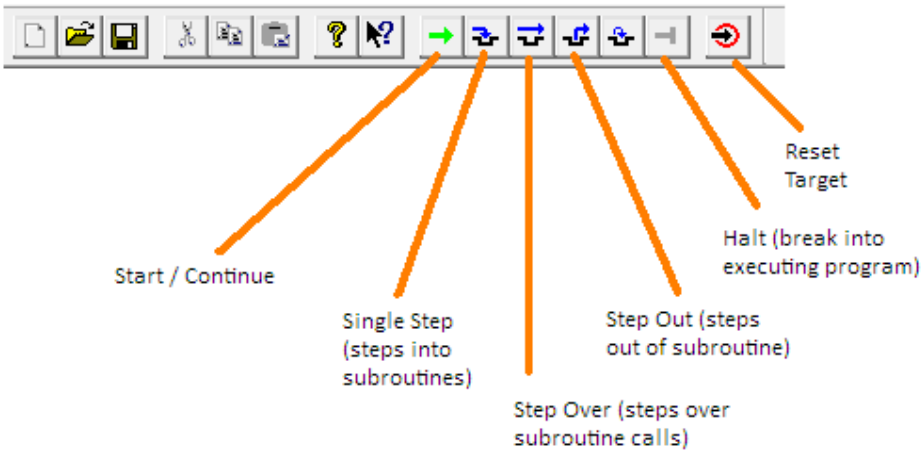


Figure 38: Hiwave debugger Toolbar

For more information about debugging, see Section 5.7.