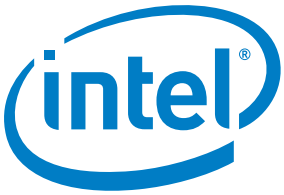


Intel® System Studio for Microcontrollers

Getting Started Guide

February 2016

Document Number: 333597-002EN

Contents

February 2016
Document Number: 333597-002EN

Tables

Revision History

Date	Revision	Description
February 2016	002	Updated release
December 2015	001	Initial release

1.0 Introduction

Intel® System Studio 2016 for Microcontrollers is an integrated tool set for developing, debugging and optimizing systems and firmware for the Intel® Quark™ microcontroller D2000.

This document provides a general overview of the Intel® System Studio 2016 for Microcontrollers and information how to use it for developing and debugging applications for the Intel® Architecture-based microcontrollers on Linux* and Windows* platforms from the command line and from the Eclipse* IDE. It gives a list of compiler options, Floating Point Emulation library functions, and points to additional product information and technical support.

The package integrates the Intel® Quark™ Microcontroller Software Interface Board Support Package (Intel® QMSI BSP) and all tools to cross compile, flash, and debug for Intel® Quark™ microcontroller D2000 with command line or Eclipse* IDE.

Intended Audience

This document is intended for the experienced system and application microcontroller developers who develop Intel® Architecture-based microcontroller systems and devices.

1.1 Terminology

Table 1. Terminology

Term	Description
BSP	Board Support Package
FTDI	Future Technologies
IRC	Intel Registration Centre
JTAG	Joint Test Action Group
QMSI	Quark Microcontroller Software Interface
UART	Universal Asynchronous Receiver/Transmitter

2.0 Useful Information before Starting

For more information on Intel® Quark ™ microcontroller D2000, refer to:
http://www.intel.com/content/www/us/en/embedded/products/quark/mcu/d2000/overview.html

For community support and FAQ's please go to:
https://communities.intel.com/community/tech/microcontrollers

Note: It is advised to read through the "Known Limitations" section of the ISSM release notes before starting.

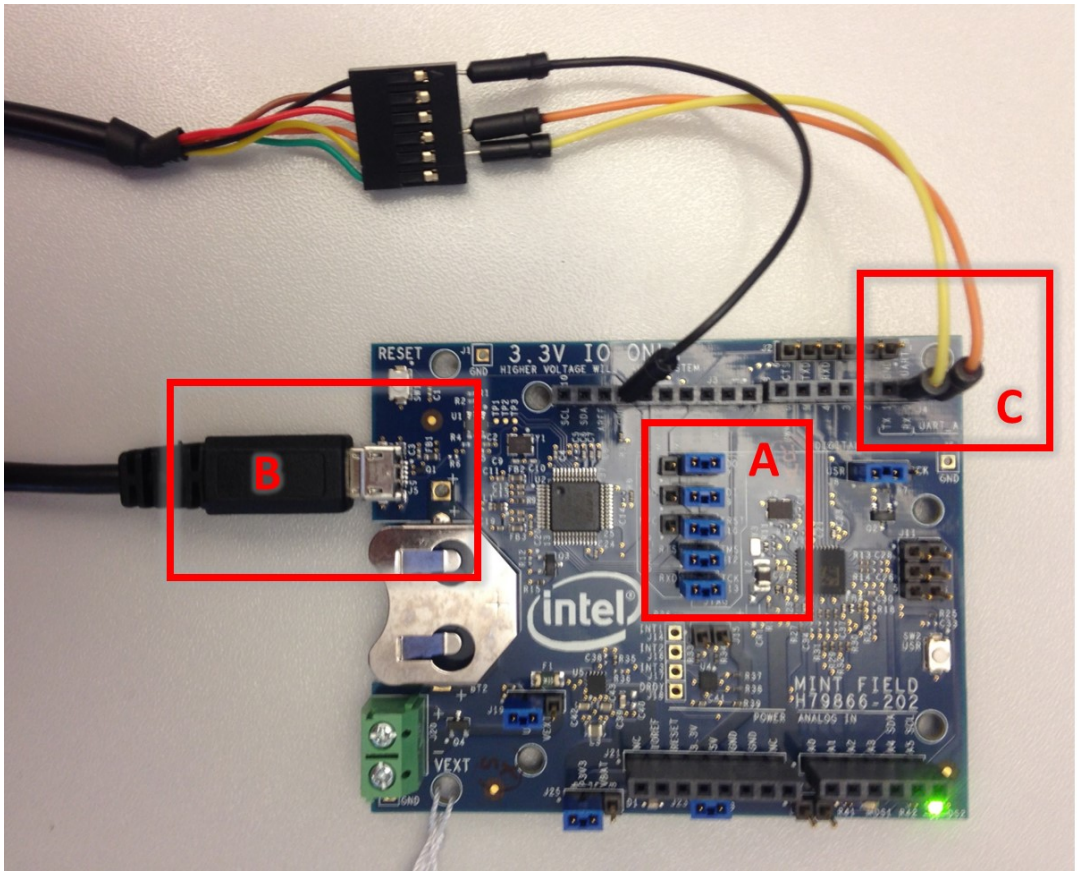
3.0 Set Up Hardware Configuration

The following information refers to a Fab B board. Check sticker on underside for identification. All other versions of board default jumper settings are adequate to use.

1. Make sure all **JTAG-Jumpers** are set correctly.
2. Connect the **USB** cable to a host machine and also to USB power supply and JTAG over USB interface to power up the Intel® Quark™ Microcontroller Board.
3. Attach the **UART-A pins** to a **FTDI cable**, for serial output of the board.

    - Connect GND (black) on the serial cable to the board's GND pin
    - Connect TXD (orange) on the serial cable to the board's RX pin
    - Connect RXD (yellow) on the serial cable to the board's TX pin.

Figure 1. Hardware Configuration – Fab B board



Note: If RX and TX of UARTB are connected to each other, then GDB will fail to initialise.

Visit http://www.cyberciti.biz/hardware/5-linux-unix-commands-for-connecting-to-the-serial-console/ and http://www.ftdichip.com/Products/Cables/USBTTLSerial.htm for serial connection details.

4.0 Linux Environment Set Up

1. Install required tools:

```
sudo apt-get install telnet
sudo apt-get install default-jre
```

2. Extract the contents of the archive to a directory with write access and run installation "as a current user":

```
tar -xvf l_cembd_mv_XXX
./l_cembd_mv_XXX/install_GUI.sh
```

3. Set the USB rules, to allow flash and debug the target:

```
~/intel/issm_2016.Y.XXX/tools/utils/install_driver.sh
```

Note that the default installation path is:

```
- ISSM_ROOT for sudo/root install:
        /opt/intel/issm_2016.Y.XXX
- ISSM_ROOT as user:
        ~/intel/issm_2016.Y.XXX
```

4. Connect the Board and verify that the Linux detects it

```
lsusb

Bus 001 Device 002: ID 0403:6014 Future Technology Devices International,
Ltd FT232H Single HS USB-UART/FIFO IC
```

5. From the ISSM_ROOT location, start the Eclipse* IDE `iss_mcu_ide_eclipse-launcher`.

The default locations would be:

```
Linux:
- ISSM_ROOT for sudo/root install:
        /opt/intel/issm_2016.Y.XXX/iss_mcu_ide_eclipse-launcher
- ISSM_ROOT as user:
        ~/intel/issm_2016.Y.XXX/iss_mcu_ide_eclipse-launcher
```

6. Go to Section 6.0 for information on using the Eclipse IDE.

5.0 Windows Environment Setup

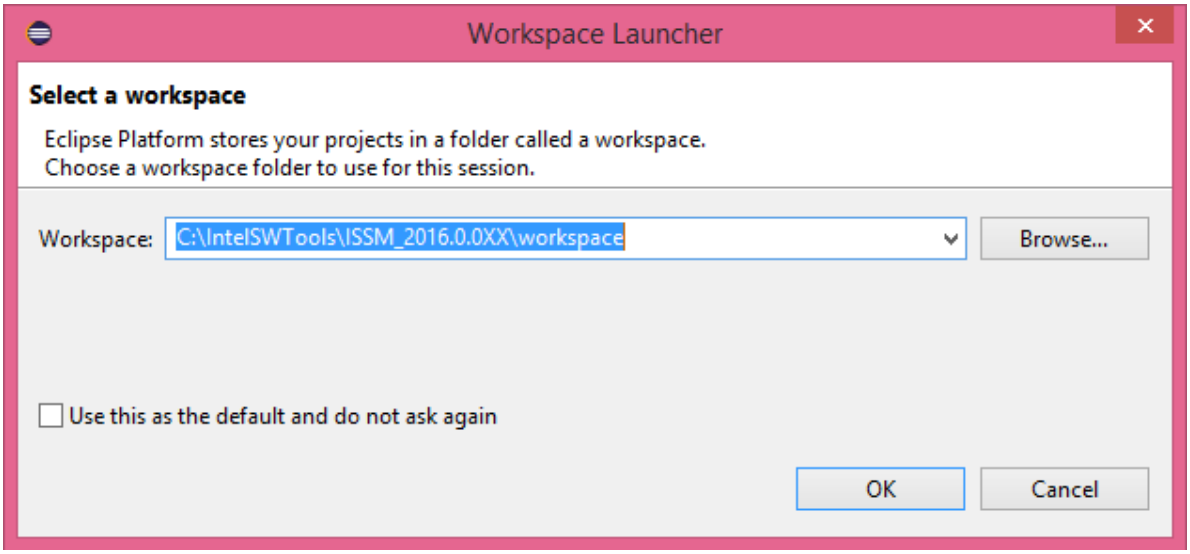
1. Ensure all prerequisites have been met:
    a. The package supports x64 host architecture only.
    b. Java* Runtime Environment version 1.7(64-bit) or higher must be installed in order to use Eclipse* Luna IDE.
    c. D2000 board should be plugged in the host PC in order to properly install the Microsoft* Winusb driver (user can reinstall the driver later from C:\IntelSWTools\ISSM_2016.0.XXX\tools\debugger\driver\install.bat)
    d. Internet connection is required to install the TinyCrypt component.

2. Double click on the installer (w_cembd_mv_XXX.exe) to run it.
3. Tools are typically installed in C:\IntelSWTools
4. ISSM_ROOT location will typically be C:\IntelSWTools\ISSM_2016_0_XXX
5. To launch the IDE:
    a. Either click on the iss_mcu_ide_eclipse-launcher.bat in that folder or
    b. In Windows 7/8 type 'Intel System Studio 2016 for Microcontrollers' in the "Search Programs and Files" pane and click on the following icon

6.0 Using the Eclipse* IDE

1. Select the location to create a workspace and click ok:

Figure 2. Create a Workspace



2. Once Eclipse is launched, the main page gives the user immediate access to quick start options and useful documentation.

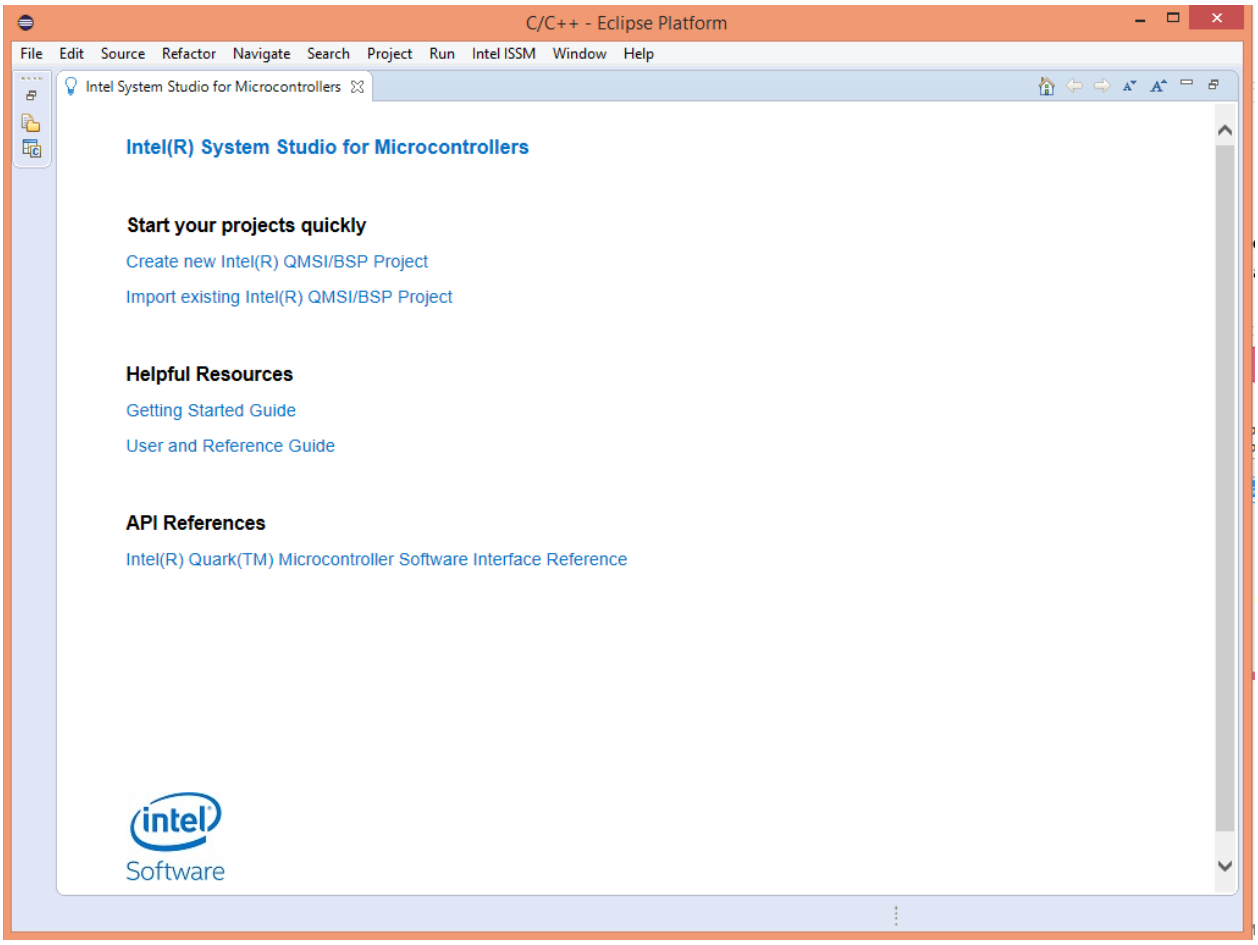


Figure 3. Eclipse* Opening Window

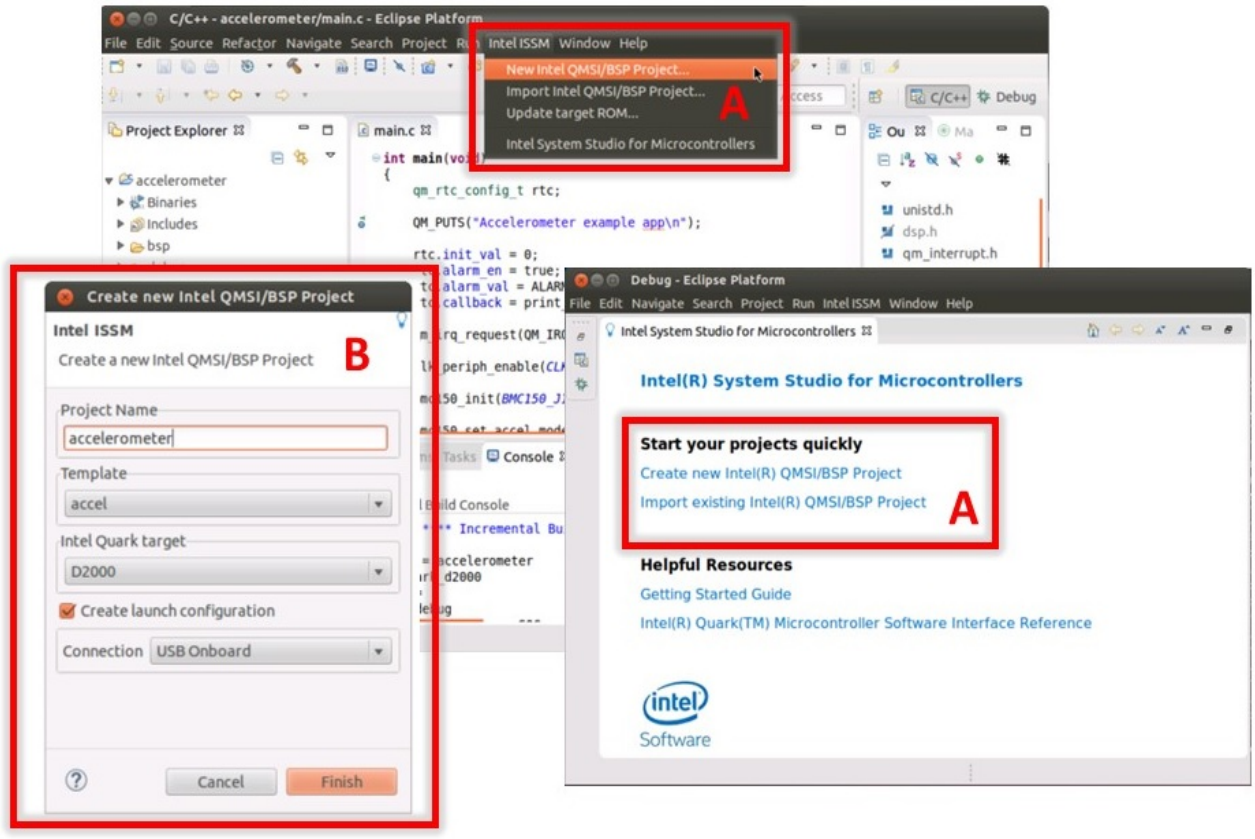
7.0 Creating a New Project in Eclipse

1. From the toolbar menu, select:
Intel ISSM > New Intel QMSI/BSP Project ...

2. In the new wizard, choose:

- **Project Name**: accelerometer
- **Template**: accel
- **Intel Quark Target**: D2000
- **Connection**: USB Onboard

Figure 4. Creating a New Project in Eclipse



8.0 Building a Project in Eclipse

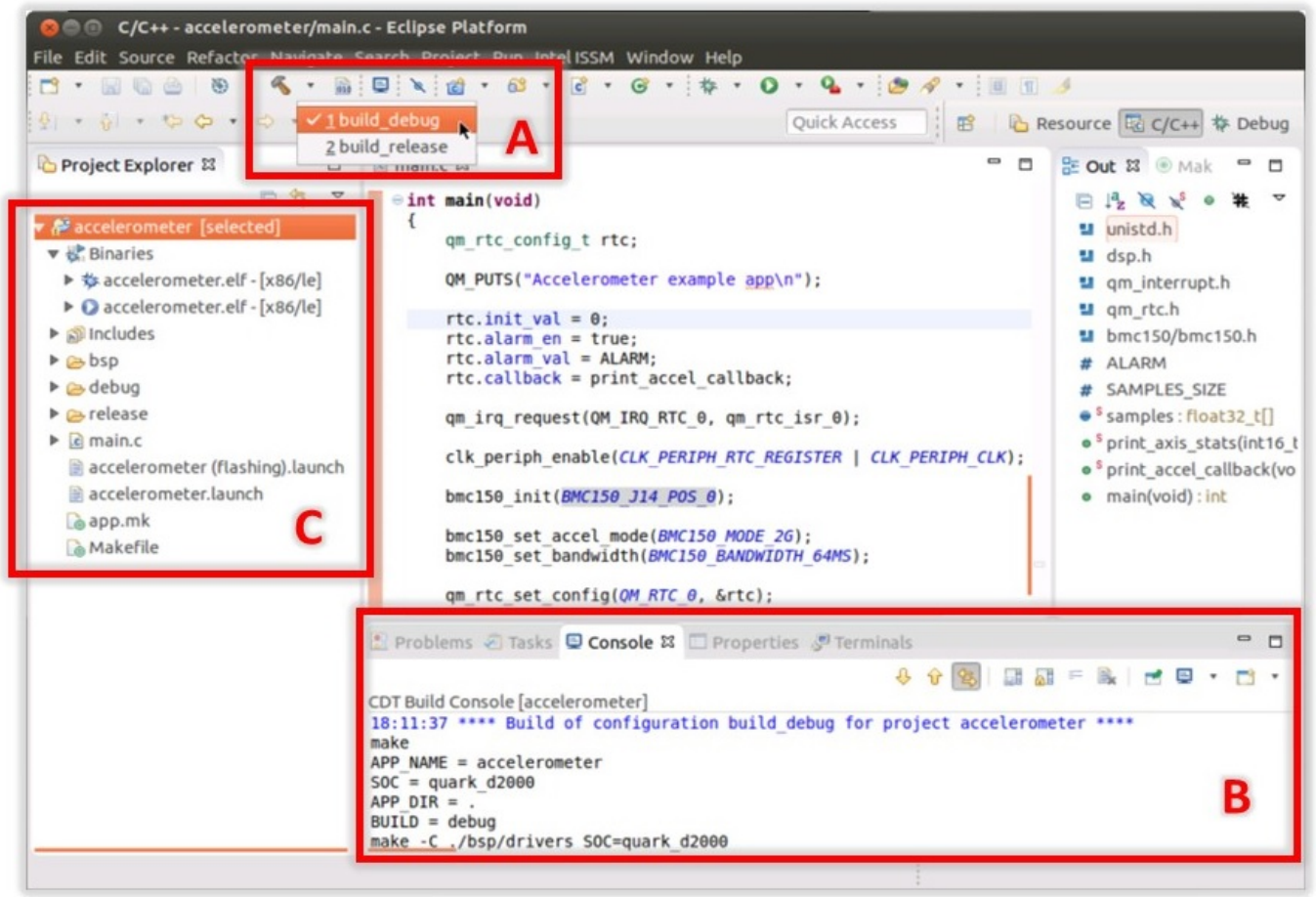
1. Highlight the chosen project and select a **Build** configuration from the drop-down list of the hammer icon (Ctrl+B)

   build_debug: create symbols for enhanced debugging

   build_release: optimize the code for deployment

2. Review the build process if desired
3. Locate the binaries

Figure 5. Building a Project in Eclipse



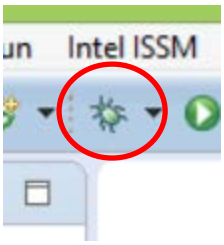
9.0 Deploying a Project

The first time a new board is used, its ROM image will need to be updated otherwise user applications may not run properly. Jump to Section 14 – “Update ROM Image” in this guide before continuing.

10.0 Debugging a Project

Click in the **Debug** drop-down list with the bug icon:

Figure 6. Bug Icon

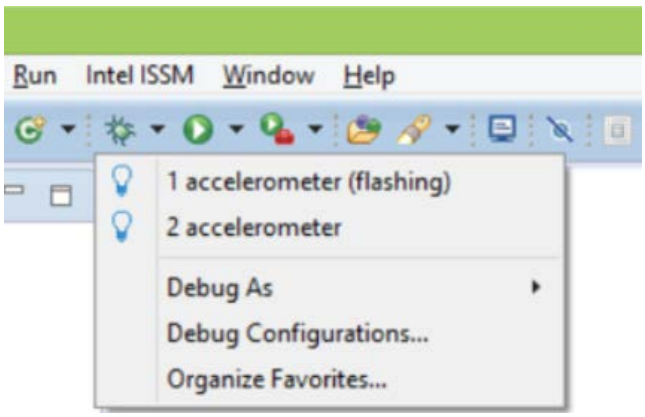


Project_Name (flashing): rebuilds the project if necessary, flashes the binary to the target, and runs to main function of the application to start debugging (slow startup time)

Project_Name: resets target, reuses flashed image target, and runs to main function of the application to restart debugging (fast startup time but the binary of the target and the host PC have to be the same)

Note: Average flashing times are approx. 40secs.

Figure 7. Debug Drop Down List



11.0 Terminating a Debug Session

Only one project can be debugged or run at the same time.

From the 'Debug' perspective, make sure all sessions are terminated before deploying a new project.

```
Toolbar Menu > Window > Open Perspective > Debug
```

1. Select the running projects to be terminated.
2. Right click and select the "Terminate" button

It is important to verify that there are no open debug sessions running **before** Running/Debugging a new project

12.0 Running a Project

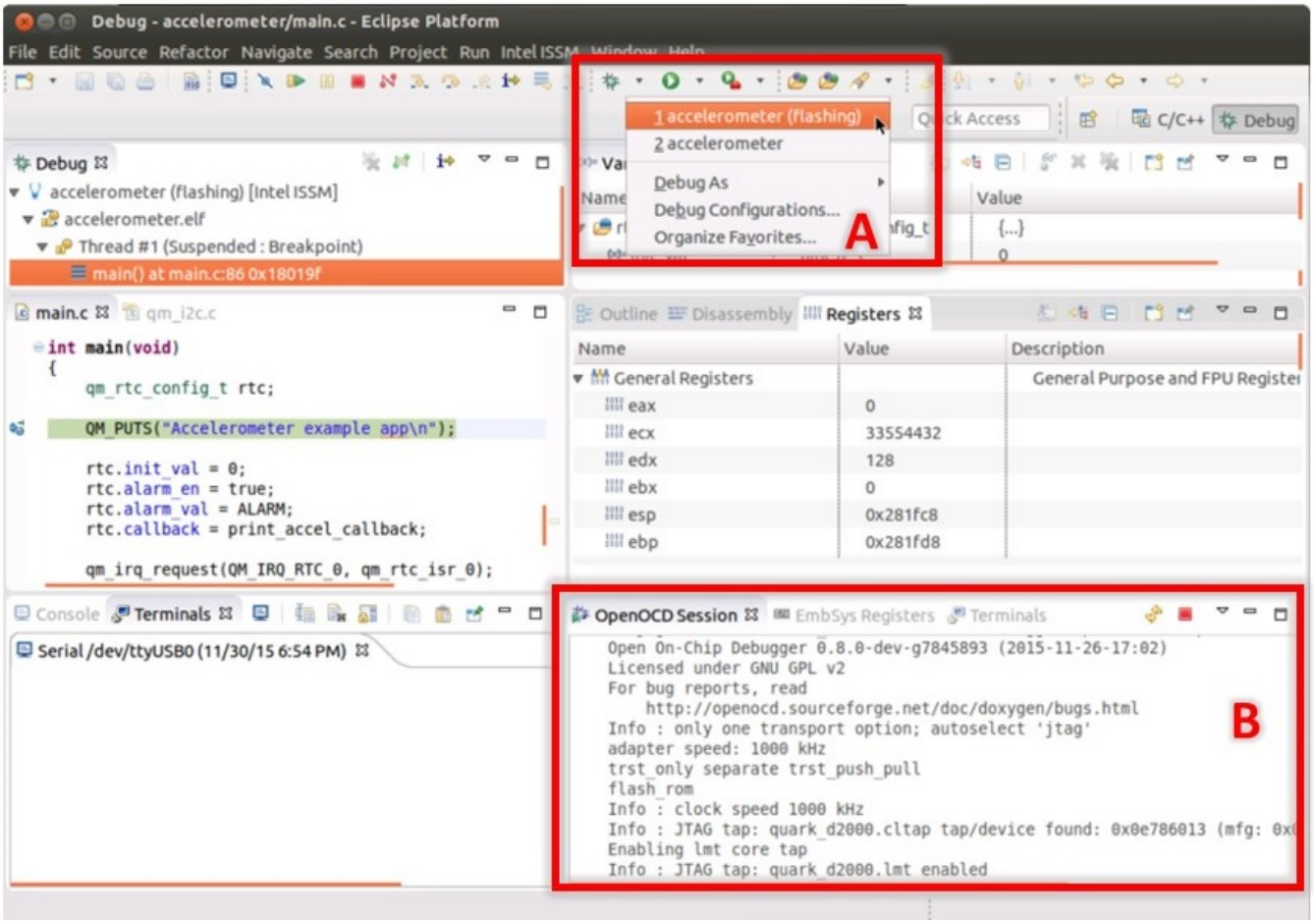
1. Click in the **Run** drop-down list with the play icon:
   Project_Name (flashing): rebuilds the project if necessary, flashes the binary to the target, and keeps the target running (slow startup time).

   Project_Name: resets target, reuses flashed image target, and resumes the target (fast startup time)

2. OpenOCD output is viewable in the session window as below

Note: Average flashing times are approx. 40secs.

Figure 8. Running a Project



Note: Both configurations will start the OpenOCD Session automatically to establish the JTAG connection. Therefore, make sure the board is connected beforehand.

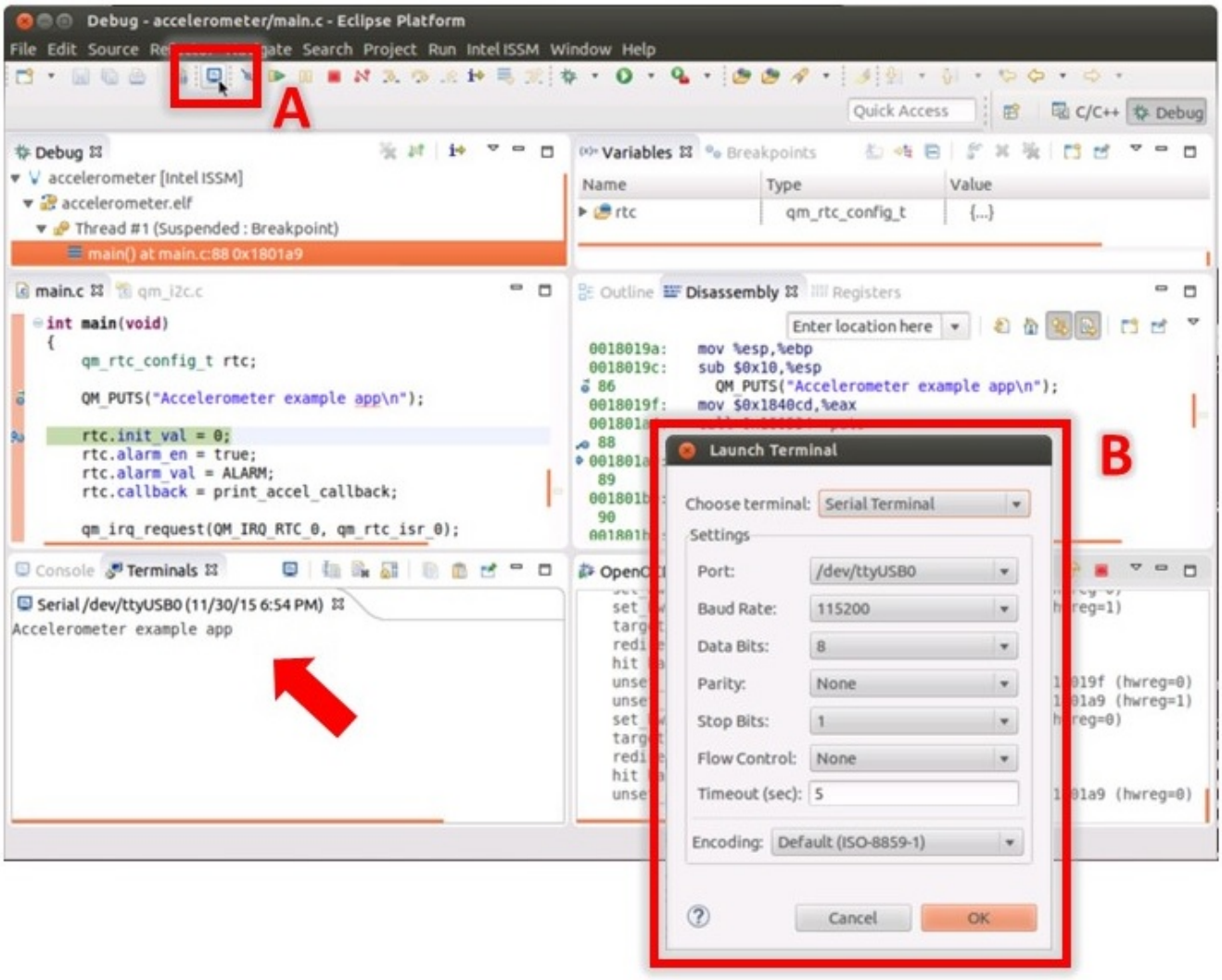
13.0 Getting a Serial Output View

Connect the serial cable as described before in the **Set up Hardware configuration**. It is possible to get the serial output within Eclipse as follows:

1. Open the launch **Terminal** with the monitor icon (Ctrl+Alt+T)
2. Configure the serial connection, by default:

Baudrate: 115200 **Parity**: None **Data Bits**: 8 **Stop bits**: 1

Figure 9. Opening a Serial View



Tip: The port will vary depending on the serial hardware used. To identify, which is the correct one, use ‘dmesg’ in Linux or ‘Device Manager’ on Windows.

Notes:

- In Linux, permissions must be available to access the USB port. Optionally run `usermod -aG dialout $(whoami)` and restart the Linux user session.
- In Windows, make sure the proper drivers for the FTDI cable are installed.

14.0 Access SOC Registers – EmbSys Registers View

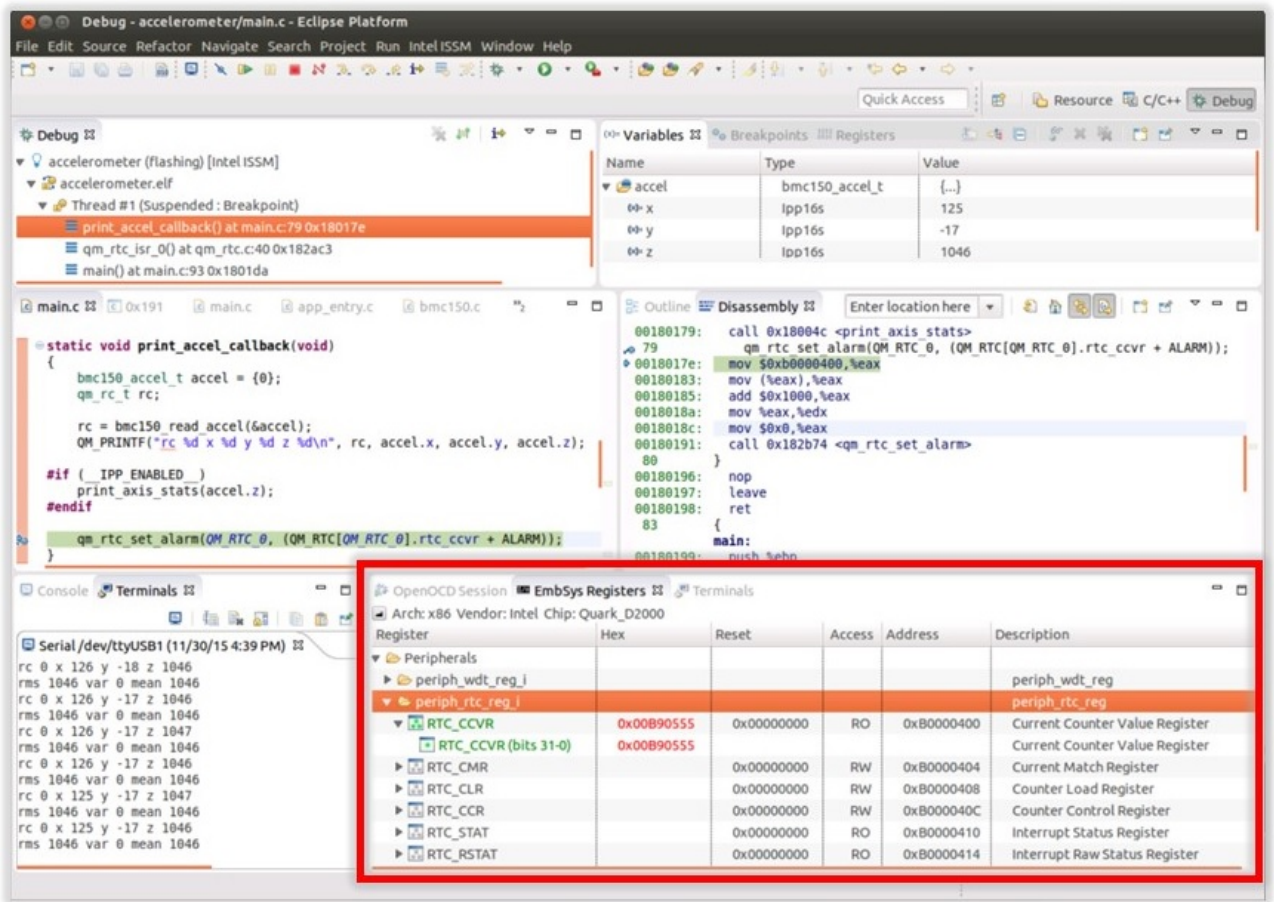
The **EmbSys Registers View** below will show the memory map registers of the target during a debug session:

- Double click on the register of interest.

  It will change to green and its value will be automatically updated when the target performs a step or a run/halt operation

- Double-click on the value to modify the value of the register

Figure 10. Accessing SOC Registers



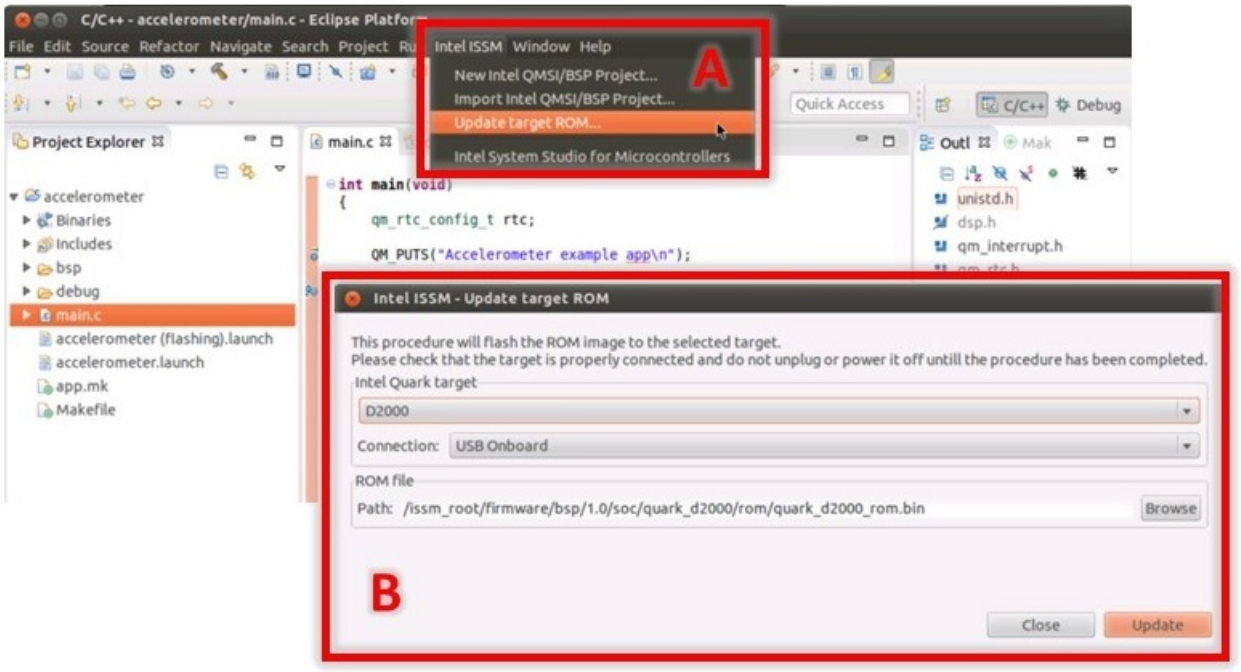
15.0 Updating ROM Image

Note: This procedure is only required if it is the first time the board is being used.

1. From the toolbar menu, select:
   Intel ISSM > Update target ROM...
2. In the new dialog, select the connection that matches the target:

   - **Intel Quark Target**: D2000
   - **Connection**: USB Onboard
   - **ROM file**: ISSM_BSP_ROOT/soc/quark_d2000/rom/quark_d2000_rom.bin

Figure 11. Updating ROM Image



NOTES:

- Make sure the board is connected beforehand and be aware that any debug/running configuration will be stopped.
- Average flashing times are approx. 40secs.

16.0 Updating the ROM from the Command Line

Note: This section is intended for advanced users, therefore only a generic workflow is explained.

Before starting, write access is required to the BSP folder. Alternatively make a copy of it and point to it within the issm_env.sh (Linux*) or issm_env_bat (Windows*) file from the **ISSM_ROOT** location.

ISSM_BSP_ROOT=$ISSM_ROOT/firmware/bsp/1.0

Or

ISSM_BSP_ROOT=c:\IntelSWTools\ISSM_2016.0.XXX\firmware\bsp\1.0

Note: Average flashing times are approx. 40secs.

16.1 Linux

From the ISSM_ROOT location, set up the environment with issm_env.sh and flash the ROM:

```
source <ISSM_ROOT>/issm_env.sh
cd $ISSM_DEBUGGER_ROOT/openocd
bin/openocd -f scripts/board/quark_d2000_onboard.cfg -c “init”
-c “flash_rom
$ISSM_BSP_ROOT/soc/quark_d2000/rom/quark_d2000_rom.bin” -c
“shutdown”
```

16.2 Windows

From the ISSM_ROOT location, set up the environment with issm_env.bat and flash the ROM:

```
c:\IntelSWTools\ISSM_2016.0.XXX\issm_env.bat
cd %ISSM_DEBUGGER_ROOT%\openocd
bin\openocd –f scripts\board\quark_d2000_onboard.cfg –c
“flash_rom
c:\\IntelSWTools\\ISSM_2016.0.XXX\\firmware\\bsp\\1.0\\soc\\qua
rk_d2000\\rom\\quark_d2000_rom.bin 0”
```

17.0 Developing in the Command Line

17.1 Linux

1. Once the steps in Section 4 have been completed the follow commands can be run.
2. From the ISSM_ROOT location, set up the environment with issm_env.sh and locate the Intel® QMSI BSP examples inside the firmware folder:

```
source ISSM_ROOT/issm_env.sh
cd $ISSM_BSP_ROOT
cd examples/magneto
make SOC=quark_d2000
```

   Find the build in:

```
ISSM_BSP_ROOT/examples/magneto/debug/quark_d2000/obj/magneto.elf
```

3. Open one terminal to run JTAG-OpenOCD:

```
source ISSM_ROOT/issm_env.sh
cd $ISSM_DEBUGGER_ROOT/openocd
bin/openocd -f scripts/board/quark_d2000_onboard.cfg
```

4. Open a second terminal to connect GDB to port 3333

```
source ISSM_ROOT/issm_env.sh
cd $ISSM_DEBUGGER_ROOT/gdb
./gdb-ia

target remote :3333
monitor gdb_breakpoint_override hard
set remotetimeout 30
monitor clk32M
file
../../../firmware/bsp/X.X/examples/magneto/debug/quark_d2000/obj/magneto.e
lf
load
monitor reset halt
flushregs
tb main
c
c
```

17.2 Windows

1. From the ISSM_ROOT location(typically C:\IntelSWTools\ISSM_2016_0_XXX), set up the host environment with issm_env.bat and locate the Intel® QMSI BSP examples inside the firmware folder:

```
C:\IntelSWTools\ISSM_2016_0_XXX\issm_env.bat
cd %ISSM_BSP_ROOT%
cd examples\magneto
make SOC=quark_d2000
```

   Find the build in:

```
ISSM_BSP_ROOT\examples\magneto\debug\quark_d2000\obj\magneto.elf
```

2. Open one terminal to run JTAG-OpenOCD:

```
C:\IntelSWTools\ISSM_2016_0_XXX\issm_env.bat
cd %ISSM_DEBUGGER_ROOT%\openocd
bin\openocd -f scripts\board\quark_d2000_onboard.cfg
```

3. Open a second terminal to connect GDB to port 3333

```
C:\IntelSWTools\ISSM_2016_0_XXX\issm_env.bat
cd %ISSM_DEBUGGER_ROOT%\gdb
gdb-ia

target remote :3333
monitor gdb_breakpoint_override hard
set remotetimeout 30
monitor clk32M
file
..\\..\\..firmware\\bsp\\X.X\\examples\\magneto\\debug\\quark_d
2000\\obj\\magneto.elf
load
monitor reset halt
flushregs
tb main
c
c
```

4. To view the output from the example above, attach the UART as per Section 3 - C and launch a serial terminal emulator like PuTTY (Select the correct COM port for the Serial line, Speed=115200 and ensure 'Implicit CR in evert LF' is checked for terminal).

18.0 Use Intel® Integrated Performance Primitives for Microcontrollers (Intel® IPP for Microcontrollers) Library

The library, Intel® IPP for Microcontrollers provides several example files located at <install-dir>/tools/lib/ipp/1.0.0/examples/. The examples use the Board Support Package (BSP) and link to Intel® Quark™ Microcontroller Software Interface.

18.1 Linux

1. In the command line, change the directory to the example file location:

```
source ISSM_ROOT/issm_env.sh
cd $IPP_LIBRARY_PATH/examples/<example-name>
```

2. Run the following command::

```
make BUILD=<release|debug>
```

18.2 Windows

1. In the command line, change the directory to the example file location:

```
C:\IntelSWTools\ISSM_2016.0.XXX\issm_env.bat
cd %IPP_LIBRARY_PATH%\examples\<example-name>
```

2. Run the following command::

```
make BUILD=<release|debug>
```

See the Intel® System Studio 2016 for Microcontrollers User and Reference Guide for more information by clicking on Help in Eclipse*.