# Intel® Quark™ Microcontroller Developer Kit D2000 Magnetometer and PWM: Lab Guide

Contents

[Intel® Quark™ Microcontroller Developer Kit D2000 Magnetometer and PWM: Lab Guide 1](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812519)

[Overview 2](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812520)

[Prerequisites 2](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812521)

[Lab Workflow 2](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812522)

[Lab Instructions 2](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812523)

[Step 1: Connect the Intel® Quark™ Microcontroller Developer Kit D2000 board 2](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812524)

[Step 2: Check the USB drivers for COM port and JTAG adapter 3](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812525)

[Step 3: Run PuTTY 4](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812526)

[Step 4: Run Intel® System Studio™ for Microcontrollers 5](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812527)

[Step 5: Update target ROM 6](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812528)

[Step 6: Create a new project 8](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812529)

[Step 7: Build the project 9](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812530)

[Step 8: Flash and run the project 9](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812531)

[Step 9: Test the project 10](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812532)

[Lab Source Code 11](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812533)

[How it works 11](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812534)

[Source Code 13](file:///C:\Users\localadmin\Downloads\D2000%20-%20Lab%20Guide%20-%20Accelerometer%20and%20PWM%20(3).DOCX#_Toc450812535)

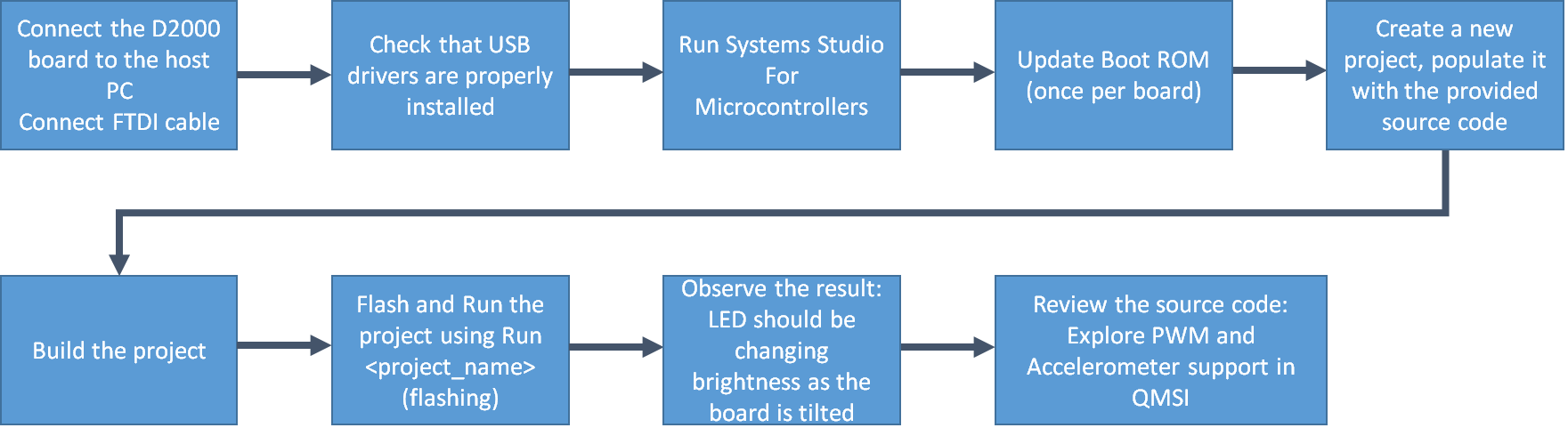
## Overview

The purpose of this lab is to show how to program Intel® Quark™ Microcontroller D2000 PWM module and Bosch BMC150 magnetometer integrated in Intel® Quark™ Microcontroller Developer Kit D2000 board using Quark Microcontroller Software Interface (QMSI) and Intel® System Studio 2016 for Microcontrollers. The sample project used in this lab reads the magnetic field, calculates the degree and prints it to serial port, and also changes on-board user LED brightness using PWM depending on the degree of direction.

## Prerequisites

* Host machine:
  + PC running Microsoft Windows 7 or later with two available USB ports.
* Software:
  + Intel® System Studio for Microcontrollers 2016.
  + PuTTY terminal emulation software
* Hardware:
  + Intel® Quark™ Microcontroller Developer Kit D2000 board.
  + USB Type A to Micro USB Type B cable. This cable is supplied with the kit.
  + FTDI TTL-232R-3V3 USB to Serial cable
  + 3 male to male jumper wires

## Lab Workflow



## Lab Instructions

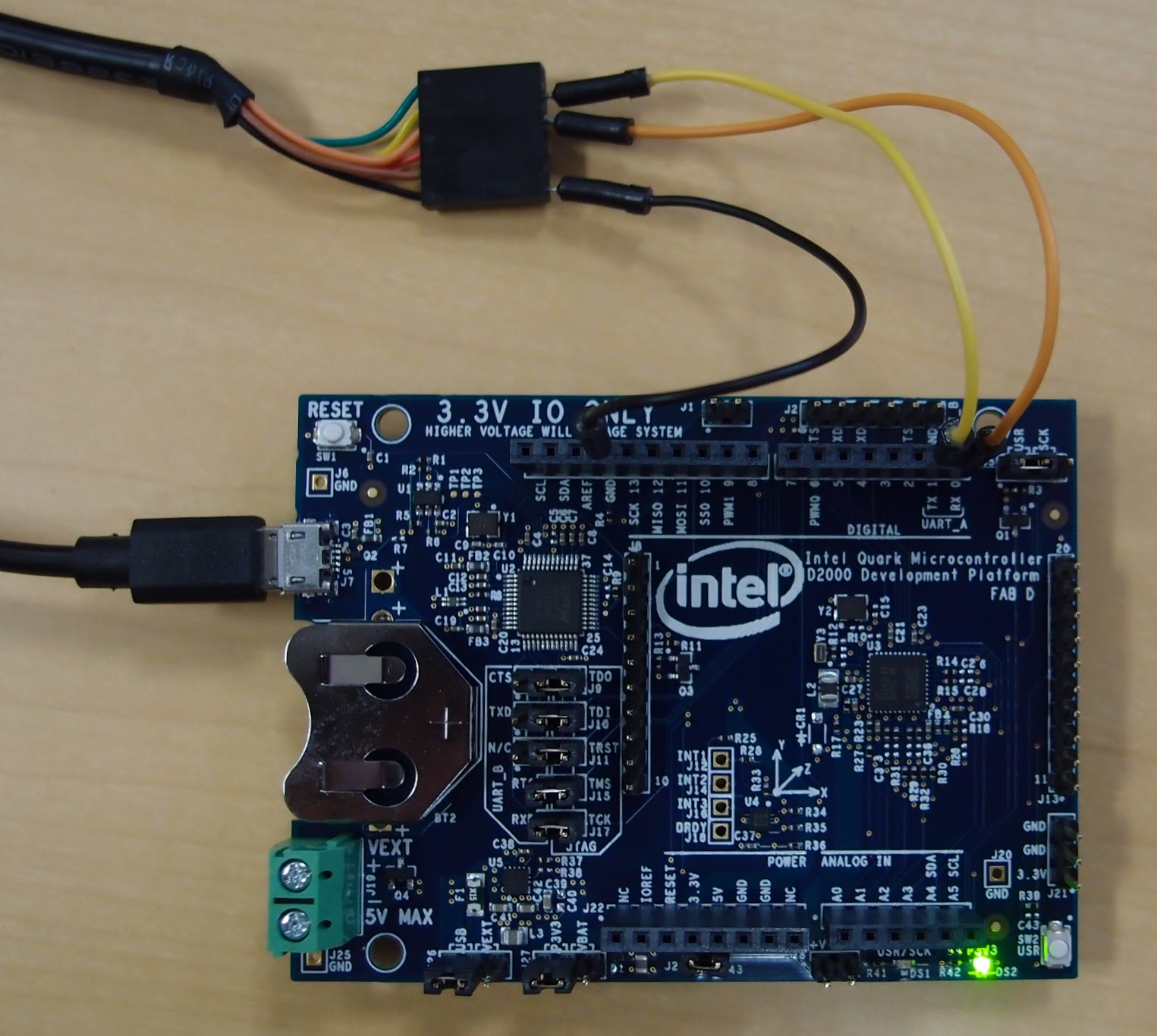
### Step 1: Connect the Intel® Quark™ Microcontroller Developer Kit D2000 board

Using jumper wires connect the FTDI TTL-232R-3V3 cable to the microcontroller as shown on the picture below.

Make sure that pin 1 of the cable (black wire, also identified by a small arrow on the connector) is connected to the GND pin of the board. Pin 4 of the cable (orange wire) is connected to the RX pin (0) of the board. And pin 5 of the cable (yellow wire) is connected to the TX pin (1) of the board.

Connect the USB end of the FTDI cable to your PC.

Connect the board to your PC using the USB Type A to Micro USB Type B cable provided in the kit. The green P3V3 LED on the board should light up.



### Step 2: Check the USB drivers for COM port and JTAG adapter

Open Windows ***Device Manager***. Make sure that you have a ***USB Serial Port*** device under ***Ports (COM & LPT)*** and ***OpenOCD JTAG*** device under ***Universal Serial Bus devices*** as shown on the screenshot below. Remember the COM port number (COM6 on the screenshot – might be different on your computer)



If the ***USB Serial Port*** device does not appear or the ***Ports (COM & LPT)*** category is missing, the FTDI driver needs to be installed. It can be downloaded from the [FTDI web site](http://www.ftdichip.com/Drivers/VCP.htm).

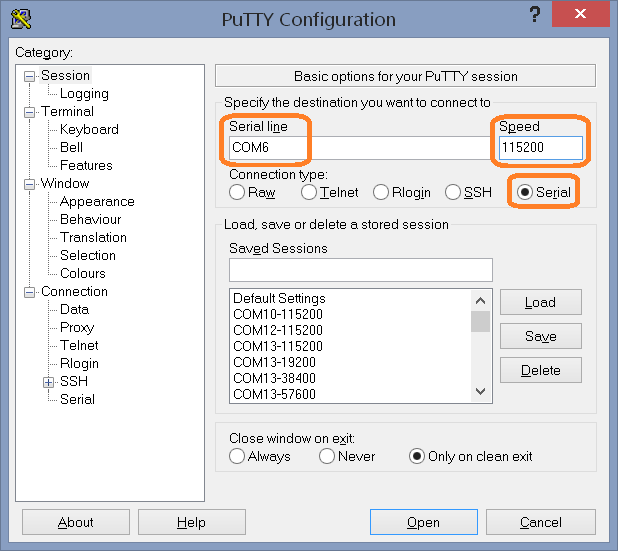
If the ***OpenOCD JTAG*** device does not appear or the ***Universal Serial Bus devices*** category is missing, the USB driver needs to be installed. To install the driver navigate to [*C:\IntelSWTools\ISSM\_2016.0.027\tools\debugger\driver*](file:///C:\IntelSWTools\ISSM_2016.0.027\tools\debugger\driver) directory using ***File Explorer***, double click on the *install.bat*, and answer ***‘y’*** in the ***OpenOCD JTAG Driver*** window that will appear.

### Step 3: Run PuTTY

Click on the Windows start button, type **putty**, and press the ***Enter*** key to the application.

The ***PuTTY Configuration*** dialog will appear. Using this dialog configure PuTTY as follows (also refer to the screenshot below):

* Select ***Serial*** radio button
* Type **115200** in the ***Speed*** field. 115200 bits per second (bps) is the default speed of the UART
* Type the COM port number (e.g. **COM6**) in the ***Serial Line*** field.

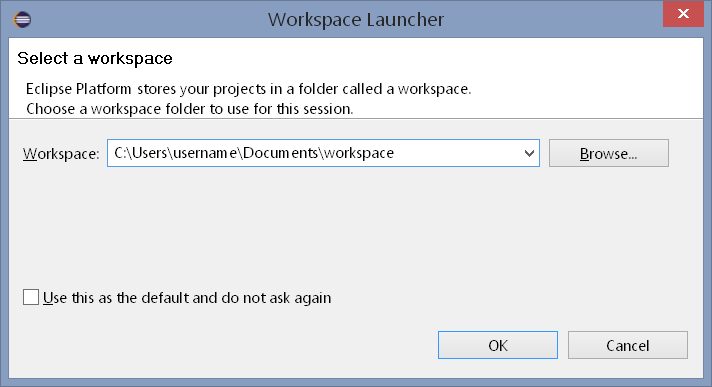


Click ***Open*** button to open the serial terminal

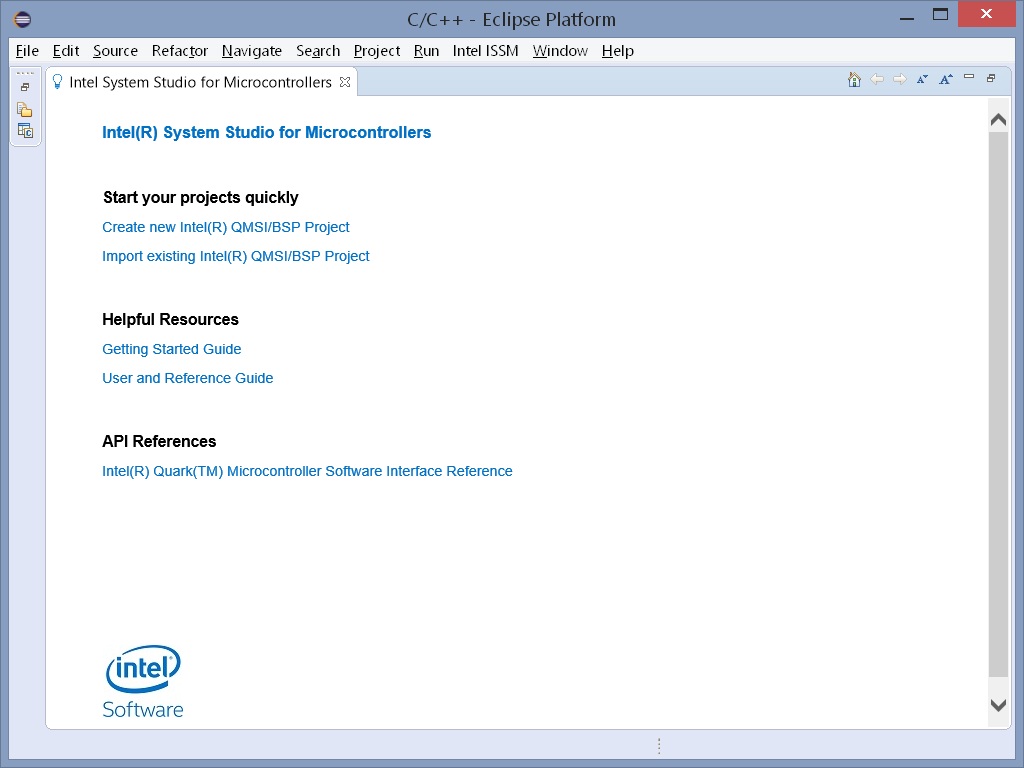
### Step 4: Run Intel® System Studio™ for Microcontrollers

Using ***File Explorer*** navigate to [*C:\IntelSWTools\ISSM\_2016.0.027*](file:///C:\IntelSWTools\ISSM_2016.0.027) directory, and double click on the *iss\_mcu\_ide\_eclipse-launcher.bat*.

The Intel® System Studio™ for Microcontrollers will prompt for the workspace location:

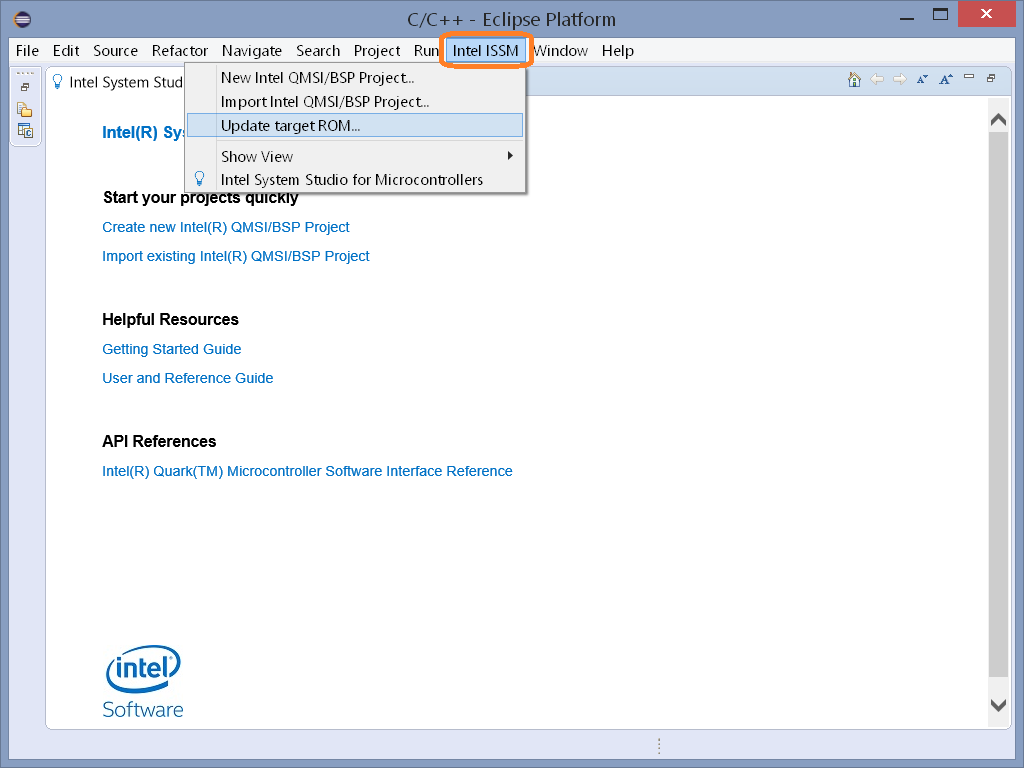


It is recommended to click **Browse…** button here and change the workspace location to your user directory, for example **C:\Users\username\Documents\workspace**. Click ***OK*** button to confirm workspace selection. The Intel® System Studio for Microcontrollers windows will appear as shown on the screenshot below.

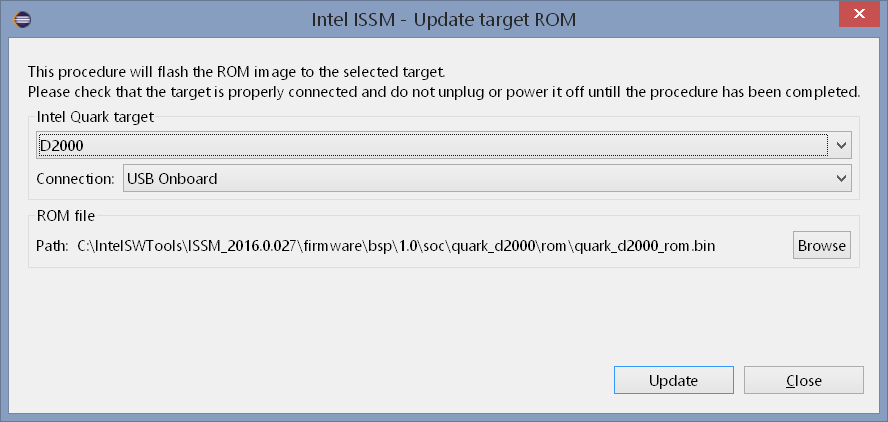


### Step 5: Update target ROM

*Note: This step needs to be performed only once per board.*

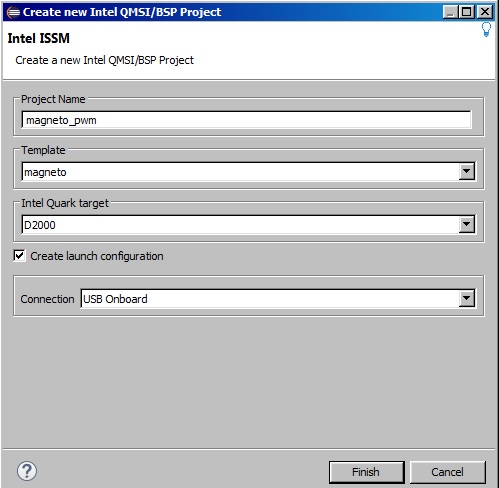
From ***Intel ISSM*** menu select ***Update target ROM…*** entry as shown on the screenshot below.

The ***Update target ROM*** dialog will appear. Make sure that the selected *Intel Quark target* is ***D2000***, and the selected *Connection* is ***USB Onboard***. Click ***Update*** button to program the ROM to the microcontroller.



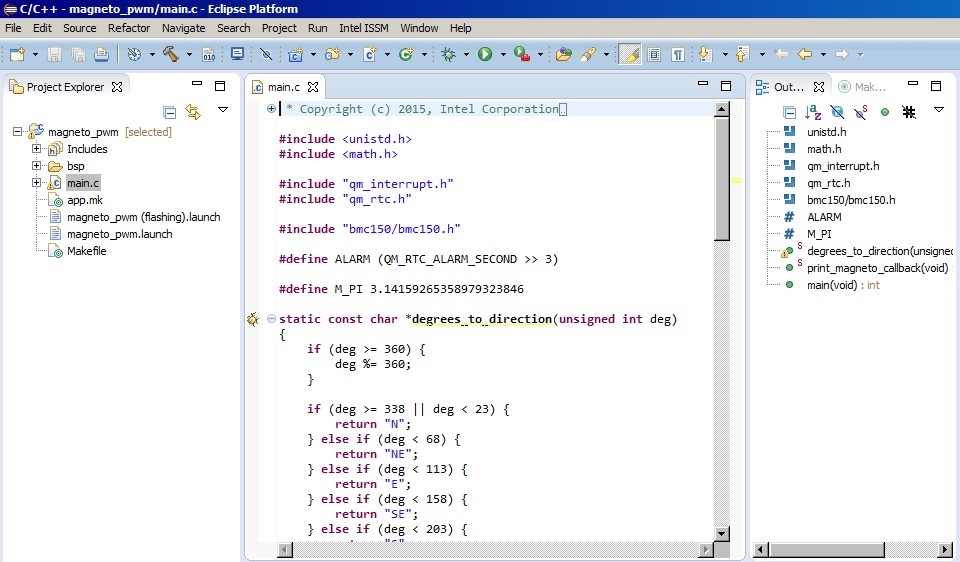
### Step 6: Create a new project

From the ***Intel ISSM*** menu select ***New Intel QMSI/BSP Project…*** The ***Create new Intel QMSI/BSP Project*** will appear. Type ***magneto\_pwm*** in the Project Name field. Make sure that other settings match to the settings on the screenshot below.



Click the ***Finish*** button to create the new project.

In the ***Project Explorer*** window on the left side of Intel® System Studio for Microcontrollers click on the *magneto\_pwm* project. Find *main.c* file and double click on it to open it the editor. Replace the *main.c* content with the code given below in the “Source Code” section (lab2\_accel\_pwm.c.txt on the USB drive). Save the file by pressing Ctrl-S key combination.



### Step 7: Build the project

Click on the ***Build*** button (“hammer”  icon) on the toolbar to build the project. Alternatively it is possible to select the build configuration using the drop down menu next to that button. Intel® Systems Studio for Microcontrollers provides two configurations: ***debug*** and ***release***. The ***debug*** configuration includes symbols in the generated binary files to facilitate debugging. It is the default configuration. The ***release*** configuration optimizes code for deployment.

### Step 8: Flash and run the project

Click on the drop down menu next to the ***Run*** button (“play” icon). Select ***magneto\_pwm (flashing)*** from the menu. The Intel® Systems Studio for Microcontrollers will recompile the code, and flash it to the microcontroller.

*Note: Intel® System Studio for Microcontrollers offers two ways to run the code: The first –* ***magneto\_pwm*** *option assumes that the code had been already programmed to the microcontroller. The second –****magneto\_pwm (flashing)*** *option will recompile and re-flash the code. Use this option if the source code had been changed since the last time microcontroller was programmed. This is the option used in this lab step.*

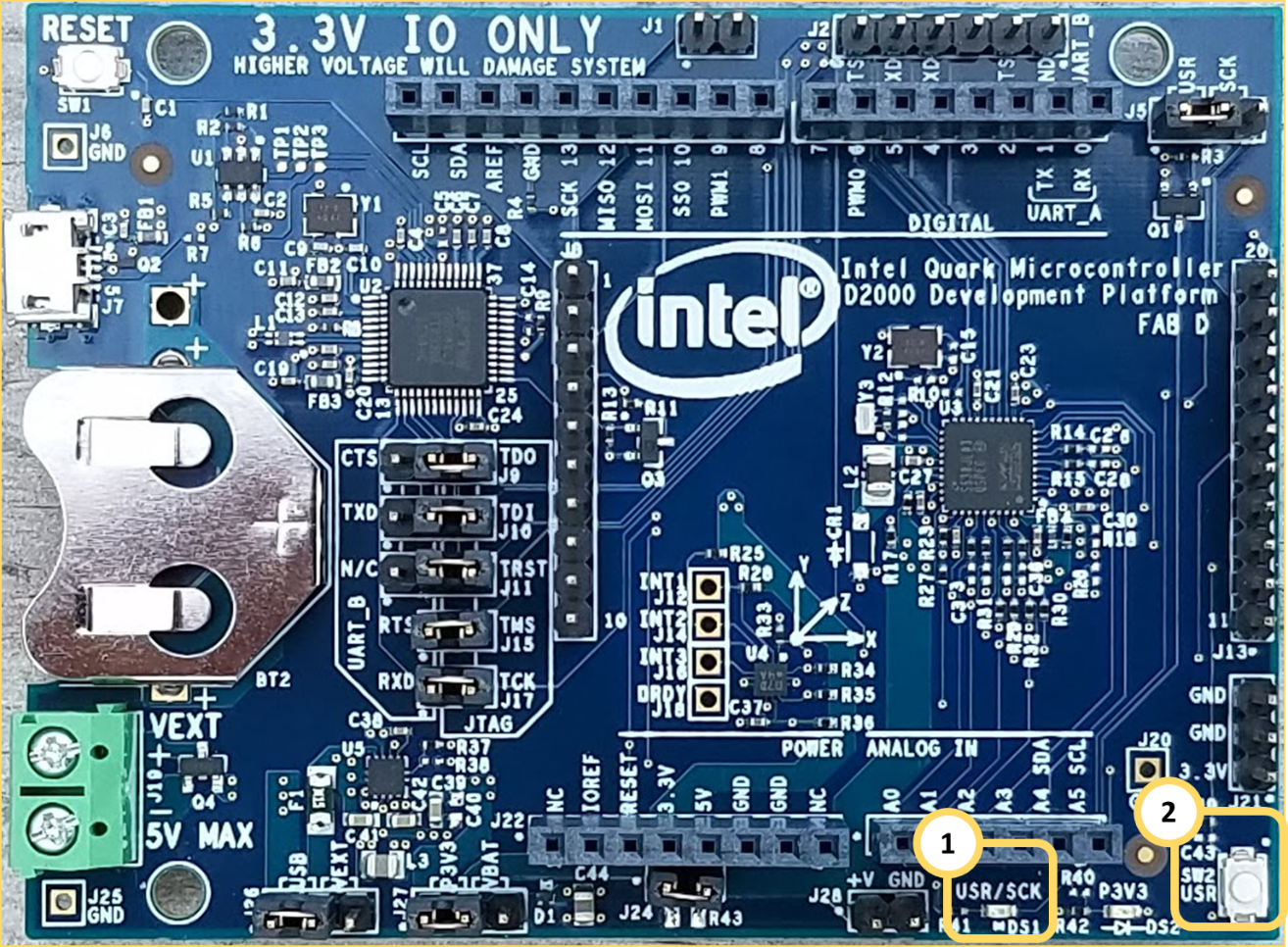


*Note: The flashing process takes some time. Look for the progress bar at the bottom right corner of the Intel® Systems Studio for Microcontrollers window:*

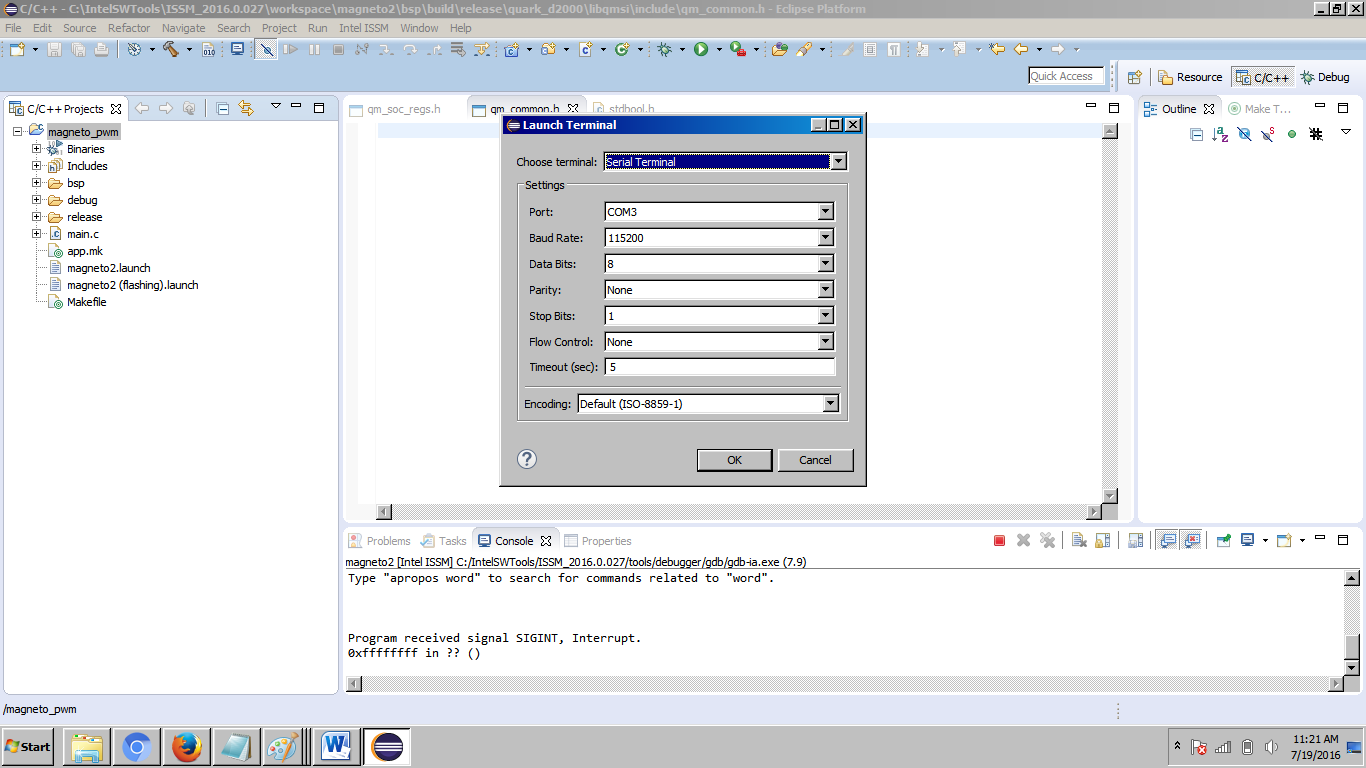
**

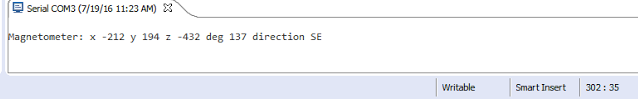
### Step 9: Test the project

Position the board horizontally. Observe that the user LED DS1 – marked as (1) on the picture is dim (or off). Rotate the board so that the USB connector points up or down. Observe that the user LED is bright.



Instead of using the Putty Terminal we used the ISSM terminal. Press Ctrl Alt T to open the ISSM launch terminal then select the right serial port. You should be seeing magnetometer readings being printed out in the terminal. Try to rotate the board. Notice how the magnetometer readings change.





## Lab Source Code

### How it works

The Magnetometer and PWM sample uses the on-board Bosch BMC150 magnetometer connected to the microcontroller using I2C interface, the PWM module integrated in Intel® Quark™ Microcontroller D2000, and the on-board user LED DS2. The LED is connected to the I/O pin number 24. The code defines and uses ***PIN\_LED*** constant to refer to that pin.

The sample begins with initializing and setting up the magnetometer:

|  |
| --- |
| /\* Initialize Magnetometer \*/  bmc150\_init(*BMC150\_J14\_POS\_0*);  bmc150\_mag\_set\_preset(*BMC150\_MAG\_PRESET\_HIGH\_ACCURACY*); |

The ***bmc150\_init*** function initializes the I2C and the magnetometer. The ***BMC150\_J14\_POS\_0*** parameter specifies the address of the magnetometer (the J14 switch was available on CRB boards).

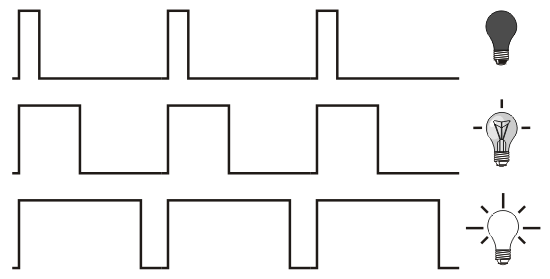
The **bmc150\_mag\_set\_preset** sets values for resulting magnetic field output noise and magnetometer part current consumption.

Next the sample initializes a periodic RTC interrupt for 0.1 seconds period, and sets up the callback for that interrupt – **print\_magneto\_callback**.

|  |
| --- |
| **#define** ALARM\_PERIOD (QM\_RTC\_ALARM\_SECOND / 10)  /\* Set up RTC alarm interrupt \*/  rtc\_cfg.init\_val = 0;  rtc\_cfg.alarm\_en = true;  rtc\_cfg.alarm\_val = ALARM\_PERIOD;  rtc\_cfg.callback = read\_accel\_callback;  qm\_irq\_request(QM\_IRQ\_RTC\_0, qm\_rtc\_isr\_0);  clk\_periph\_enable(*CLK\_PERIPH\_RTC\_REGISTER* | *CLK\_PERIPH\_CLK*);  qm\_rtc\_set\_config(*QM\_RTC\_0*, &rtc\_cfg); |

The **print\_magneto\_callback reads** the magnetometer data using ***bmc150\_read\_mag*** function, prints the data on the serial terminal using ***QM\_PRINTF*** function, calculates the LED brightness depending on the acceleration on X axis, and calls ***set\_pwm*** function to set up PWM accordingly.

The ***set\_pwm*** function configures a PWM output on Intel® Quark™ Microcontroller D2000.

*Note: PWM (pulse-width modulation) is a technique used by microcontrollers to get an analog signal using a digital output. It uses the ratio between on and off cycles to control the signal amplitude (this ratio is known as duty cycle).*

The ***set\_pwm*** function performs one of two actions depending on the duty cycle setting:

1. If the ***duty\_cycle*** parameter is between 1 and 254, it configures the PWM module, and multiplexes the output of that module to the I/O pin. The PWM module uses two timers: The first timer - ***hi\_count*** determines the number of system clock cycles the output stays in logic ‘1’ state. The second timer – ***lo\_count*** determines the number of system clock cycles the output stays in logic ‘0’ state. The ***set\_pwm*** function calculates these parameters using the specified period and the ***duty\_cycle*** parameters:

|  |
| --- |
| /\* Configure PWM \*/  /\* Calculate low time and high counts \*/  pwm\_cfg.hi\_count = period \* duty\_cycle / 256;  pwm\_cfg.lo\_count = period - pwm\_cfg.hi\_count; |

1. Alternatively, if the ***duty\_cycle*** parameter is 0 (always off) or 255 (always on), it multiplexes the GPIO module to that I/O pin, configures it in the output mode, and sets the pin value to logic ‘0’ or logic ‘1’ respectively.

|  |
| --- |
| **if** (0 == duty\_cycle) {  qm\_gpio\_clear\_pin(*QM\_GPIO\_0*, pin);  } **else** {  qm\_gpio\_set\_pin(*QM\_GPIO\_0*, pin);  } |

The code used for the pwm is shown in the box bellow. Depending on the angle of degree we set the LED to be on for a certain amount of the period cycle and off for the rest of the period cycle. As long as the direction did not change the LED cycle stayed the same. Once the degree of direction changes the same code is used to determine the pwm for that direction.

|  |
| --- |
| qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);  clk\_sys\_udelay(1000);  qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);  clk\_sys\_udelay(1000); |

### Source Code

|  |
| --- |
| /\* Magnetometer and PWM sample \*/  **#include "qm\_soc\_regs.h"**  **#include "qm\_gpio.h"**  **#include "qm\_scss.h"**  **#define LED\_BIT 24**  **static qm\_gpio\_port\_config\_t cfg;**  **#ifndef \_UNISTD\_H\_**  **#define \_UNISTD\_H\_**  **# include <sys/unistd.h>**  **#endif /\* \_UNISTD\_H\_ \*/**  **#include "math.h"**  **#define MAX\_LED\_BLINKS (3)**  **#include "qm\_interrupt.h"**  **#include "qm\_rtc.h"**  **#include "bmc150/bmc150.h"**  **#define ALARM (QM\_RTC\_ALARM\_SECOND >> 3)**  **#define M\_PI 3.14159265358979323846**  **static const char \*degrees\_to\_direction(unsigned int deg)**  **{**  **QM\_PUTS("Starting: Led blink\n");**  **cfg.direction = BIT(LED\_BIT);**  **qm\_gpio\_set\_config(QM\_GPIO\_0, &cfg);**  **if (deg >= 360) {**  **deg %= 360;**  **}**  **int counter = 0 ;**  **if (deg >= 338 || deg < 23) {**  **/\* qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT); \*/**  **while (counter < MAX\_LED\_BLINKS) {**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(1000);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(1000);**  **counter++;**  **}**  **return "N";**  **}**  **else if (deg < 68)**  **{**  **while (counter < MAX\_LED\_BLINKS) {**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(500);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(500);**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(2500);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(2500);**  **counter++;**  **}**  **return "NE";**  **} else if (deg < 113) {**  **while (counter < MAX\_LED\_BLINKS) {**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(2500);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(6500);**  **counter++;**  **}**  **return "E";**  **} else if (deg < 158) {**  **while (counter < MAX\_LED\_BLINKS) {**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(12500);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(12500);**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(6500);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(6500);**  **counter++;**  **}**  **return "SE";**  **} else if (deg < 203)**  **{**  **while (counter < MAX\_LED\_BLINKS) {**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(18500);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(18500);**  **counter++;**  **}**  **/\* qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT); \*/**  **return "S";**  **} else if (deg < 248) {**  **while (counter < MAX\_LED\_BLINKS) {**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(185000);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(18500);**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(24500);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(24500);**  **counter++;**  **}**  **return "SW";**  **} else if (deg < 293) {**  **while (counter < MAX\_LED\_BLINKS) {**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(28500);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(28500);**  **counter++;**  **}**  **return "W";**  **} else {**  **while (counter < MAX\_LED\_BLINKS) {**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(28500);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(500);**  **qm\_gpio\_set\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(500);**  **qm\_gpio\_clear\_pin(QM\_GPIO\_0, LED\_BIT);**  **clk\_sys\_udelay(28500);**  **counter++;**  **}**  **return "NW";**  **}**  **}**  **static void print\_magneto\_callback(void)**  **{**  **bmc150\_mag\_t mag = {0};**  **double heading;**  **int deg;**  **bmc150\_read\_mag(&mag);**  **heading = atan2(mag.y, mag.x);**  **if (heading < 0) {**  **heading += 2 \* M\_PI;**  **}**  **deg = (int)(heading \* 180 / M\_PI);**  **QM\_PRINTF("mag x %d y %d z %d deg %d direction %s\n", mag.x, mag.y,**  **mag.z, deg, degrees\_to\_direction(deg));**  **qm\_rtc\_set\_alarm(QM\_RTC\_0, (QM\_RTC[QM\_RTC\_0].rtc\_ccvr + ALARM));**  **}**  **int main(void)**  **{**  **qm\_rtc\_config\_t rtc;**  **qm\_rc\_t rc;**  **QM\_PUTS("Magnetometer example app\n");**  **rtc.init\_val = 0;**  **rtc.callback = print\_magneto\_callback;**  **qm\_irq\_request(QM\_IRQ\_RTC\_0, qm\_rtc\_isr\_0);**  **clk\_periph\_enable(CLK\_PERIPH\_RTC\_REGISTER | CLK\_PERIPH\_CLK);**  **rc = bmc150\_init(BMC150\_J14\_POS\_0);**  **if (rc != QM\_RC\_OK) {**  **return rc;**  **}**  **rc = bmc150\_mag\_set\_power(BMC150\_MAG\_POWER\_ACTIVE);**  **if (rc != QM\_RC\_OK) {**  **return rc;**  **}**  **rc = bmc150\_mag\_set\_preset(BMC150\_MAG\_PRESET\_HIGH\_ACCURACY);**  **if (rc != QM\_RC\_OK) {**  **return rc;**  **}**  **qm\_rtc\_set\_config(QM\_RTC\_0, &rtc);**  **return rc;**  **}** |