Intel® Quark™ Microcontroller Developer Kit
D2000 Accelerometer, Magnetometer, Temperature : Lab 5 Guide

Contents

Return To Top

Overview

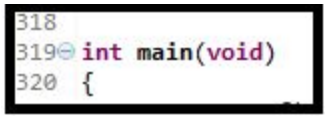
The purpose of this lab is to show how to program and combine the magnetometer, accelerometer and temperature applications with PWM. These are integrated in Intel® Quark™ Microcontroller Developer Kit D2000 board using Quark Microcontroller Software Interface (QMSI) and Intel® System Studio 2016 for Microcontrollers. The sample project used in this lab reads the acceleration, magnetometer, and temperature sensor and prints it to serial port, and also changes on-board user LED brightness using PWM with pauses between each sensor to demonstrate which one is currently being displayed on the serial port.

Lab instructions

Since this lab is based on combining the code for all three sensors (accelerometer, magnetometer, and temperature), this lab assumes that the code for the individual sensors are already functional. To install the drivers and software necessary to run any of the sensors on the Intel Quark D2000 microcontroller follow the individual instruction for the installation of the accelerometer (lab 2), magnetometer (lab3), or temperature sensor (lab 4).

Go to the file “main.c” and look for the main function as shown in Figure 1.0.

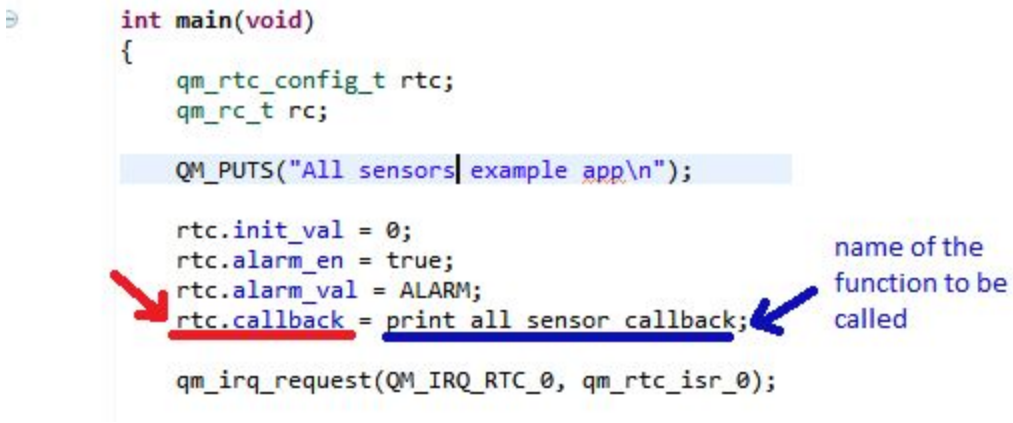
Figure 1.0



Under the main function there is a line of code that starts with ”rtc.callback” underlined in red as shown in Figure 1.1. The area highlighted in blue on the other side of the assignment operator from “rtc.callback” is the function that will be called in order to display the values gathered from each of the sensors.

Return To Top

Figure 1.1

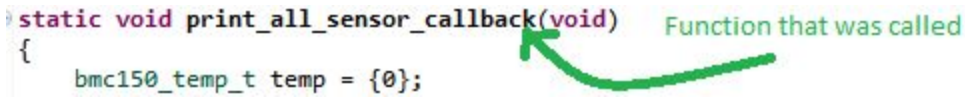


In your case, the function being called may not be called “print_all_sensor_callback” but there should still be a function being called.

3. Finding the definition of the print function being called……...3

Go to the function being called(In this case “print_all_sensor_callback” Figure 1.2)

Figure 1.2



4. Making objects from each struct representing a sensor……...3

This function, an object needs to be created for each struct that represents a sensor. The Figure 1.2 shows an example of how a temperature object is created. The “bmc150_temp_t” is the name of the struct and it is defining an object known as “temp.” Each struct representing a sensor is defined in the “bmc_150.h” file (Figure 1.3). Each object created has a read function declared in the “bmc_150.h” file (Figure 1.4). The read function needs to be called in order to initialize the parameters of each object. An example of the temp read function being called is shown in FIgure 1.5.

Return To Top

bmc_150.h (FIgure 1.3)

```
80  typedef struct {
81      int16_t x, y, z;
82  } bmc150_accel_t;
83
84  typedef struct {
85      int x, y, z;
86  } bmc150_mag_t;
87
88
89
90  typedef struct{
91      int8_t temp_data;
92  }bmc150_temp_t;
```

bmc_150.h (Figure 1.4)

```
95  qm_rc_t bmc150_read_temp(bmc150_temp_t *const temp);
96  qm_rc_t bmc150_read_accel(bmc150_accel_t *const accel);
97  qm_rc_t bmc150_read_mag(bmc150_mag_t *const mag);
98
```

main.c (Figure 1.5)

```
bmc150_read_temp(&temp);/*function initializes the values in the temp object*/
```


With the exception of the magnetometer, the values of the temperature and accelerometer can then be printed out using the QM_PRINTF() built in function. With the magnetometer, a heading must be created in order to get a degree(Figure 1.6 blue arrow). The heading makes sure that the x and y values obtained from the magnetometer sensor do not result in a negative reading in the unit circle in which the degree is based on.Once the heading is calculated, the degrees can be obtained(Figure 1.6 green arrows). From there the values of the magnetometer sensor can be printed using the QM_PRINTF() buit in function as well as the degrees and direction.

Return To Top

```
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("Magnetometer: x %d y %d z %d deg %d direction %s", mag.x, mag.y,
      mag.z, deg, degrees_to_direction(deg));
QM_PRINTF("\n\n");
```

read function

heading created

degrees obtained

Calling function to print direction

main.c (Figure 1.6)

If there is a function being called in the “print_all_sensor_call_back” function that is not currently in the main.c, refer to the individual magnetometer, temperature, or accelerometer labs and templates to obtain(copy and paste) those functions that need to be in the “main.c” file.

Once everything is functional and output is being displayed, the modulation is needed to differentiate the output between each sensor using the LED. Figure 1.7 demonstrates how to turn on the LED with a delay when it’s on then how to turn it off with a delay when its off. In Figure 1.7 the delay is 1,000,000 cycles.

main.c (Figure 1.7)

```
qm_gpio_set_pin(QM_GPIO_0, LED_BIT);/* turns on LED*/
clk_sys_udelay(1000000);/*LED on delay*/

qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);/* turns off LED*/
clk_sys_udelay(1000000);/*LED off delay*/
```


Place the on-off LED code before the declaration of each object to ensure that the LED will turn off and on with a 1,000,000 cycle delay between the readings of each sensor.

Return To Top

At this point, the code is complete.We have provided the complete code for the combined sensors below.

Source Code

```
30 #include <unistd.h>
31 #include <math.h>
32
33 #include "qm_interrupt.h"
34 #include "qm_scss.h"
35 #include "qm_rtc.h"
36
37 #include "bmc150/bmc150.h"
38 #include "qm_gpio.h"
39
40 #define ALARM (QM_RTC_ALARM_SECOND >> 3
41
42 #define M_PI 3.14159265358979323846
43
44 #define LED_BIT 24
45 #define MAX_LED_BLINKS (1);
46 static qm_gpio_port_config_t cfg;
47 static void accel_blinky(double x);
48
```

Return To Top

```
49  static const char *degrees_to_direction(unsigned int deg)
50  {
51
52      qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
53              clk_sys_udelay(1000000);
54
55              qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
56              clk_sys_udelay(1000000);
57      if (deg >= 360) {
58          deg %= 360;
59      }
60
61      if (deg >= 338 || deg < 23) {
62          qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
63                  clk_sys_udelay(100);
64
65                  qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
66                  clk_sys_udelay(100);
67                  qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
68                          clk_sys_udelay(100);
69
70                          qm_gpio_clear_pin(QM_GPIO_0, LED_B
71                          clk_sys_udelay(100);
72          return "N";
73      } else if (deg < 68) {
74          qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
75                  clk_sys_udelay(1000);
76
77                  qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
78                  clk_sys_udelay(1000);
79
80                  qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
81                                  clk_sys_udelay(1000);
82
```

Return To Top

```
                                qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
                                clk_sys_udelay(1000);
        return "NE";
    } else if (deg < 113) {
        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);

                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);
        return "E";
    } else if (deg < 158) {
        qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
                        clk_sys_udelay(4500);

                        qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
                        clk_sys_udelay(4500);

                        qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
                                        clk_sys_udelay(4500);

                                        qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
                                        clk_sys_udelay(4500);
        return "SE";
    } else if (deg < 203) {
        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);
```

Return To Top

```
                        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);
        return "S";
    } else if (deg < 248) {
        qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
                        clk_sys_udelay(8500);

                        qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
                        clk_sys_udelay(8500);

                        qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
                                        clk_sys_udelay(8500);

                                        qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
                                        clk_sys_udelay(8500);
        return "SW";
    } else if (deg < 293) {
        qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
                        clk_sys_udelay(10500);

                        qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
                        clk_sys_udelay(10500);

                        qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
                                        clk_sys_udelay(10500);

                                        qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
                                        clk_sys_udelay(10500);
        return "W";
    } else {
        qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
                        clk_sys_udelay(12500);
```

Return To Top

```
154
155                         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
156                         clk_sys_udelay(12500);
157
158                         qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
159                                         clk_sys_udelay(12500);
160
161                                         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
162                                         clk_sys_udelay(12500);
163         return "NW";
164     }
165 }
166 static void temperature_blink(double temp)
167 {
168     qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
169                             clk_sys_udelay(1000000);
170
171                             qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
172                             clk_sys_udelay(1000000);
173     qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
174                 clk_sys_udelay(1000000);
175
176                 qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
177                 clk_sys_udelay(1000000);
178
179     QM_PUTS("\n");
180     cfg.direction = BIT(LED_BIT);
181     qm_gpio_set_config(QM_GPIO_0, &cfg);
182
183     if(temp < 0)
184     {
185
186         qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
187         clk_sys_udelay(-1 * temp * 10000);
188
189         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
```

Return To Top

```
189         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
190         clk_sys_udelay(-1 * temp * 10000);
191         qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
192                 clk_sys_udelay(-1 * temp * 10000);
193
194                 qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
195                 clk_sys_udelay(-1 * temp * 10000);
196                 qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
197                         clk_sys_udelay(-1 * temp * 10000);
198
199                         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
200                         clk_sys_udelay(-1 * temp * 10000);
201
202
203     }
204     else{
205         qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
206         clk_sys_udelay(temp * 1000);
207
208         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
209         clk_sys_udelay(temp * 1000);
210         qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
211                 clk_sys_udelay(temp * 1000);
212
213                 qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
214                 clk_sys_udelay(temp * 1000);
215                 qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
216                         clk_sys_udelay(temp * 1000);
217
218                         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
219                         clk_sys_udelay(temp * 1000);
220     }
221
222 }
```

Return To Top

```
223 static void accel_blinky(double x)
224 {
225
226     if((x > 0 && x <= 5) || (x < 0 && x >= -5))
227     {
228         qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
229         clk_sys_udelay(1000);
230         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
231         clk_sys_udelay(1000);
232         qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
233                 clk_sys_udelay(1000);
234                 qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
235                 clk_sys_udelay(1000);
236                 qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
237                         clk_sys_udelay(1000);
238                         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
239                         clk_sys_udelay(1000);
240     }
241     if((x > 5 && x <= 10) || (x < -5 && x >= -10))
242     {
243         qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
244         clk_sys_udelay(70000);
245         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
246         clk_sys_udelay(70000);
247         qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
248                 clk_sys_udelay(70000);
249                 qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
250                 clk_sys_udelay(70000);
251                 qm_gpio_set_pin(QM_GPIO_0, LED_BIT);
252                         clk_sys_udelay(70000);
253                         qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);
254                         clk_sys_udelay(70000);
255     }
256
257
258 }
```

Return To Top

```
259 static void print_all_sensor_callback(void)
260 {
261
262     qm_gpio_set_pin(QM_GPIO_0, LED_BIT);/* turns on LED*/
263     clk_sys_udelay(1000000);/*LED on delay*/
264
265     qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);/* turns off LED*/
266     clk_sys_udelay(1000000);/*LED off delay*/
267
268     bmc150_temp_t temp = {0};/* temperature object created*/
269     bmc150_read_temp(&temp);/*function initializes the values in the temp object*/
270     temperature_blink(temp.temp_data);/*temperature blink function*/
271     QM_PRINTF("Temperature data: %d C\t", temp.temp_data);
272     QM_PRINTF("\n\n");
273
274
275
276     qm_gpio_set_pin(QM_GPIO_0, LED_BIT);/* turns on LED*/
277     clk_sys_udelay(1000000);/*LED on delay*/
278
279     qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);/* turns off LED*/
280     clk_sys_udelay(1000000);/*LED off delay*/
281
282     bmc150_accel_t accel = {0};/* accelerometer object created*/
283     bmc150_read_accel(&accel);/* accelerometer object initialized*/
284
285     QM_PRINTF("Accelerometer: x %d y %d z %d", accel.x, accel.y, accel.z);
286     accel_blinky(accel.x);/*accelerometer blink function*/
287     QM_PRINTF("\n\n");
288
289
290     qm_gpio_set_pin(QM_GPIO_0, LED_BIT);/* turns on LED*/
291     clk_sys_udelay(1000000);/*LED on delay*/
292
293     qm_gpio_clear_pin(QM_GPIO_0, LED_BIT);/* turns off LED*/
294     clk_sys_udelay(1000000);/*LED off delay*/
295
```

Return To Top

```
296     bmc150_mag_t mag = {0};
297     double heading;
298     int deg;
299
300     bmc150_read_mag(&mag);
301
302     heading = atan2(mag.y, mag.x);
303
304     if (heading < 0) {
305         heading += 2 * M_PI;
306     }
307
308     deg = (int)(heading * 180 / M_PI);
309
310     QM_PRINTF("Magnetometer: x %d y %d z %d deg %d direction %s", mag.x, mag.y,
311           mag.z, deg, degrees_to_direction(deg));
312     QM_PRINTF("\n\n");
313
314   clk_sys_udelay(1000000);
315
316     qm_rtc_set_alarm(QM_RTC_0, (QM_RTC[QM_RTC_0].rtc_ccvr + ALARM));
317 }
318
```

Return To Top

```
319 int main(void)
320 {
321     qm_rtc_config_t rtc;
322     qm_rc_t rc;
323
324     QM_PUTS("All sensors example app\n");
325
326     rtc.init_val = 0;
327     rtc.alarm_en = true;
328     rtc.alarm_val = ALARM;
329     rtc.callback = print_all_sensor_callback;
330
331     qm_irq_request(QM_IRQ_RTC_0, qm_rtc_isr_0);
332
333     clk_periph_enable(CLK_PERIPH_RTC_REGISTER | CLK_PERIPH_CLK);
334
335     rc = bmc150_init(BMC150_J14_POS_0);
336     if (rc != QM_RC_OK) {
337         return rc;
338     }
339
340     rc = bmc150_mag_set_power(BMC150_MAG_POWER_ACTIVE);
341     if (rc != QM_RC_OK) {
342         return rc;
343     }
344
345     rc = bmc150_mag_set_preset(BMC150_MAG_PRESET_HIGH_ACCURACY);
346     if (rc != QM_RC_OK) {
347         return rc;
348     }
349
350     qm_rtc_set_config(QM_RTC_0, &rtc);
351
352     return rc;
353 }
```

Return To Top