Intel® Quark™ microcontroller D2000 – How to communicate with MATLAB® over UART

Introduction

Intel® System Studio 2016 for Microcontrollers is an integrated tool suite for developing, optimizing, and debugging systems and firmware for the Intel® Quark™ microcontroller D2000 and Intel® Quark™ SE microcontroller development boards, which offers a microcontroller core to enable applications from device control to edge sensing IoT solutions.

Objective of this document

This document explains how to communicate with MATLAB® over UART of D2000. Most of IoT applications would collect sensory data and send it to back-end analytic server, but it is important to analyze actual sensor data before send it to server. During the development phase, developer would check the reliability of data acquisition from the sensor on serial terminal such as Putty or in Excel when user can import those data into spreadsheet. One of popular tool of those data analysis in visual figure is to use MATLAB® and this document shows how configure the Intel® Quark™ microcontroller D2000’s UART so that MATLAB® collect sensor data to plot these data in visual axes.

Figure 1 Visualization of data from Intel® Quark™ microcontroller D2000 over UART
# Table of Contents

Intel® Quark™ microcontroller D2000 – How to communicate with MATLAB® over UART ..................... 1

Introduction ....................................................................................................................................................... 1

Objective of this document ................................................................................................................................. 1

1. Intel® Quark™ microcontroller D2000 Application implementation ......................................................... 3
   1.1 Overview .................................................................................................................................................... 3
   1.2 Read a magneto sensor (BMC150) to log the heading ........................................................................... 4
   1.3 Read a ADC value from ambient light sensor ....................................................................................... 4
   1.4 Transfer ADC data in a string format to host machine ........................................................................ 5

2. MATLAB® Application which present sensor data in GUIDE™ axes ....................................................... 7
   2.1 MATLAB® scripts for serial opening to communicate with Intel® Quark™ microcontroller Intel® Quark™ microcontroller D2000 .............................................................................................................. 7
   2.2 MATLAB® scripts for parsing serial string and put into matrix .......................................................... 8
   2.3 Plot sensor data of light intensity which comes from ambient light sensor ....................................... 9

3. Hardware connection for UART Intel® Quark™ microcontroller D2000 .................................................. 10

4. Reference ..................................................................................................................................................... 11
1. Intel® Quark™ microcontroller D2000 Application implementation

1.1 Overview

The example of communication with MATLAB® would read sensory data and transfer it to host machine over UART channel, the MATLAB® script receives this data and fill in the matrix while plotting in axes in real time. You shall start from creating an reference application of “magneto “ and add ADC reading routine to check the ambient sensor according to the different heading of Intel® Quark™ microcontroller D2000. The Figure 1 shows implemented application of a “SunFlower” which read ambient light sensor values while swiping it using servo motor and send light intensity data, heading from magneto sensor over UART.

Figure 2 Light sensor monitoring application on Intel® Quark™ microcontroller D2000
1.2 Read a magneto sensor (BMC150) to log the heading

There is a 6-axis eCompass-BMC150 on Intel® Quark™ microcontroller D2000, it measures the earth’s geomagnetic field as well as dynamic and static acceleration in all three dimensions and outputs tilt-compensated heading or orientation information. Table 1 shows the magneto sensor reading routine which was copied when you create a reference example of “magneto” form Intel System Studio for Microcontrollers.

### Table 1 Magneto sensor reading routine

```c
static void read_magneto_callback(void)
{
    #ifdef USE_BMC_150
        double heading;
        bmc150_read_mag(&mag);
        heading = atan2(mag.y, mag.x);
        if (heading < 0) {
            heading += 2 * M_PI;
        }
        heading_in_degree = (int)(heading * 180 / M_PI);
    #else
        heading_in_degree = 0;
    #endif
    //QM_PRINTF("magnex %d y %d z %d deg %d direction %s\n", mag.x, mag.y,
            //    mag.z, heading_in_degree, degrees_to_direction(heading_in_degree));
}
```

1.3 Read a ADC value from ambient light sensor

The “SunFlower” reads the ambient light sensors accordance with different heading and checks which angle of position would be the best heading for getting intensive lighting. Table 2 shows the ADC reading routine which use two ADC channels to read ambient light sensors and simple average then store it to local variables.

### Table 2 ADC reading from two ambient light sensors

```c
static int read_adc_callback ( void )
{
    #ifdef USE_LIGHT_SENSOR
        qm_adc_xfer_t xfer;
        qm_adc_channel_t channels[] = {QM_ADC_CH_8, QM_ADC_CH_9};
```

4
/* Set up xfer */
xfer.ch = channels;
xfer.ch_len = NUM_CHANNELS;
xfer.samples = samples_polled;
xfer.samples_len = NUM_SAMPLES_POLLED;

/* Run the conversion */
if (qm_adc_convert(QM_ADC_0, &xfer)) {
    QM_PRINTF("ERROR: qm_adc_convert failed \r\n");
    return 1;
}

avg_light_value[0] = (unsigned int)samples_polled[0];
avg_light_value[1] = (unsigned int)samples_polled[1];

/* Print the values of the samples */
for (i = 0; i < NUM_SAMPLES_POLLED; i+=2) {
    // Get the average
    avg_light_value[0] = (avg_light_value[0] + (unsigned int)samples_polled[i]) / 2;
    avg_light_value[1] = (avg_light_value[1] + (unsigned int)samples_polled[i+1]) / 2;
}
#endif

return 0;  // success

1.4 Transfer ADC data in a string format to host machine

In the “SunFlower” application, there is 10mS timer which keep run in the whole life cycle then the timer callback can be utilized to check the serial message from the host machine to get user command and also read the magneto sensor and ambient light sensor. Finally it sends log sensory date over UART to plot those sensor data on MATLAB® axes on the fly. Table 3 shows the actual serial transferring routine which use UART channel to host machine which would receive those data in MATLAB®.

Table 3 Transfer sensory data over UART to host machine

<table>
<thead>
<tr>
<th>case STS_RUN:</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>adjust_pwm_value( fakeMotoOutput );</td>
<td></td>
</tr>
<tr>
<td>if( iDirection &gt; 0 ) fakeMotoOutput += 1;</td>
<td></td>
</tr>
<tr>
<td>else fakeMotoOutput -= 1;</td>
<td></td>
</tr>
<tr>
<td>if( fakeMotoOutput &gt;= MAX_PWM_DUTY_RATIO ) {</td>
<td></td>
</tr>
</tbody>
</table>
iDirection = -1;
}

if( fakeMotoOutput <= MIN_PWM_DUTY_RATIO ) {
    iDirection = 1;
}

read_magneto_callback();
read_adc_callback();

QM_PRINTF("%d, %d, %d, %d \r\n", ticks, fakeMotoOutput, \n    (int)heading_in_degree, (int)(avg_light_value[0]), \n    (int)(avg_light_value[1]) );

    // When we have button click event
if( button_clicked == 1 ) {
    iMainNext = STS_STOP;
    button_clicked = 0;
}

break;
2. MATLAB® Application which present sensor data in GUIDE™ axes

2.1 MATLAB® scripts for serial opening to communicate with Intel® Quark™ microcontroller

Intel® Quark™ microcontroller D2000

In MATLAB® workspace, the liens shows how to create serial object which will be opened to communicate with Intel® Quark™ microcontroller D2000 and due to RF serial module’s constraint of max baudrate, the 57600bps was selected.

Table 4 Create serial object in MATLAB®

Global SerialPortHandle;
SerialPortHandle = serial('COM5', 'BAUDRATE', 57600);

Then UI dialogue was implemented using “GUIDE” feature of the MATLAB to open & close serial port and start & stop timer which plot the ADC data on dedicated axes periodically. The following code block shows the call-back function of “Open” button in dialogue.

Table 5 Call-back function of the "Open" button

function pushbutton1_Callback(hObject, eventdata, handles)
    global SerialPortHandle;
    global bSerialOpen;
    % hObject    handle to pushbutton1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    disp('Open the COM port');
    axes(handles.LightSensor);
    try
        fopen(SerialPortHandle);
        bSerialOpen = 1;
    catch
        fclose(instrfind);
        disp('Error! Please check serial port is available');
    end

And next code block shows the call-back function of “Close” button in dialogue which only close the serial port when we successfully open the serial before.
Table 6 Call-back function of the "Close" button

```matlab
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global SerialPortHandle
global updateTimer
global bSerialOpen;

disp('Close the COM port');

if (bSerialOpen == 1)
    fclose(SerialPortHandle);
end
delete(SerialPortHandle);
delete(updateTimer);
```

2.2 MATLAB® scripts for parsing serial string and put into matrix

As a next step, you need to create a simple m file in the MATLAB® and Table 4 explain who parse the serial string data in MATLAB® and put it into integer matrix. This ‘SerialRxCallBack’ was registered as receive callback function when the serial port is initialized and called at every receiving of “terminator” character of “LF” then read full one line string. Then, it would convert formatted string to integer array and copy it into matrix which will be used when plot the sensory date in figure.

Table 7 Call-back function for receive characters from the serial

```matlab
function SerialRxCallBack(hObject, ~)
global LightSensorA;
global LightSensorA_avg;
global LightSensorB;
global LightSensorB_avg;
global HeadingInDegree;

OneLine = 'temp';
OneRow = (1:5);

OneLine = fscanf(hObject, '%s');
% disp(OneLine);
OneRow = sscanf(OneLine, '%d, %d, %d, %d, %d');
% OneRow(2) : 25 ~ 80
if OneRow(2) < 81 & OneRow(2) > 24
    index = OneRow(2)-24;
    HeadingInDegree(index) = OneRow(3);
    LightSensorA(index) = OneRow(4);
    LightSensorA_avg(index) = (LightSensorA_avg(index) + LightSensorA(index))/2
else
    LightSensorB(index) = OneRow(5);
end
```
2.3 Plot sensor data of light intensity which comes from ambient light sensor

When the “SunFlower” application run on Intel® Quark™ microcontroller D2000, it transfers sensory data over UART at every 10mS and Figure 3 shows real-time data of intensity of light and heading from magneto sensor. As you can see, it’s easy to analyze the trending data and compare it with other values of sensor data in the fly. It would help developer to check if those data is expected one and also do more post processing using plenty of signal processing method which was provided by the MATLAB®.

Figure 3 Real-time data of intensity of light accordance with heading
3. Hardware connection for UART Intel® Quark™ microcontroller D2000

In the ISSM 2016 getting started document, you can check the required hardware configuration for the Intel® Quark™ microcontroller D2000 reference platform with the following information.

A. Make sure that your JTAG-Jumpers are set properly (white lines)

B. Connect to your host machine with a USB cable to power up your board and connect with JTAG

C. Optionally, attach the UART-A pins to a FTDI cable to get the serial output of your board
   - Connect GND (brown) on the serial to the board’s GND pin
   - Connect TXD (red) on the serial cable to the board’s RX pin
   - Connect RXD (yellow) on the serial cable to the board’s TX pin.

Figure 4 Serial connection on Intel® Quark™ microcontroller D2000
4. Reference

- Intel® System Studio for Microcontrollers Getting start

- Intel® Quark™ microcontroller D2000 Microcontroller CRB User Guide:

- Intel® Quark™ microcontroller D2000 Microcontroller Datasheet: