ECEN 5813 - PRINCIPLES OF EMBEDDED SOFTWARE – PROJECT PROPOSAL

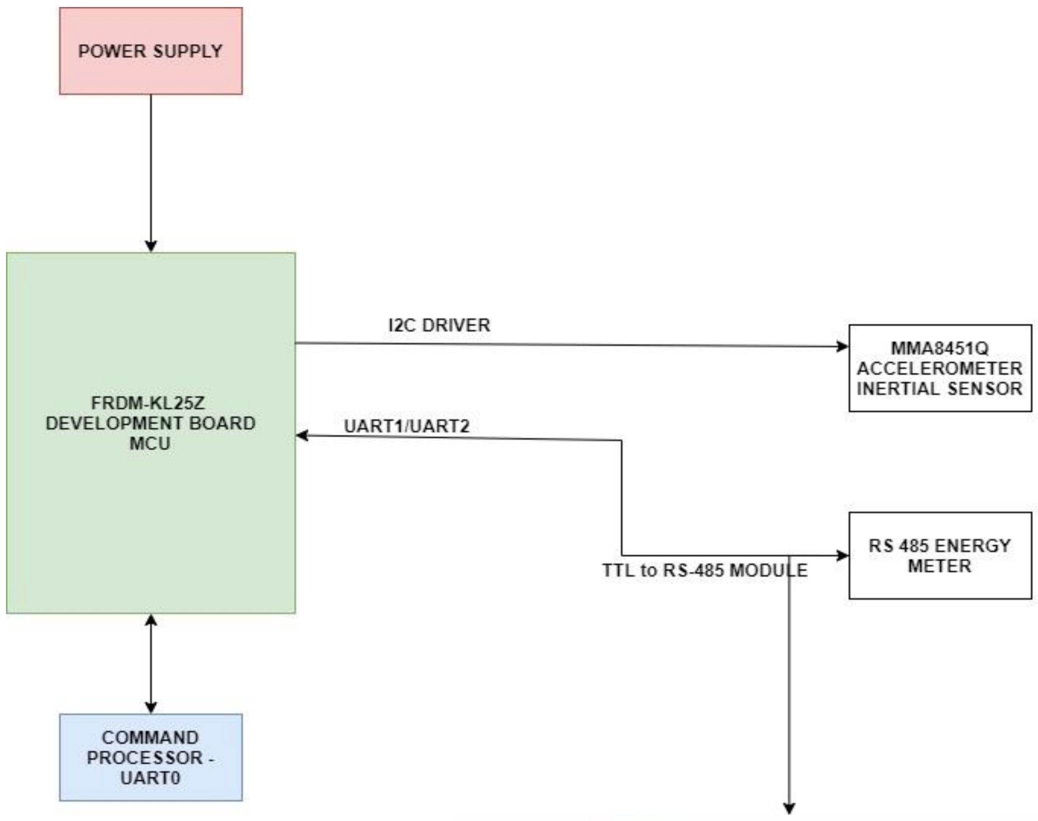
TITLE – COMMAND PROCESSOR TO READ AND DISPLAY ACCELEROMETER DATA AND MODBUS METER DATA

Objective:

Developing the command processor to get the data from MMA8451Q accelerometer sensor over I2C. The other part of the project includes interfacing UART1 or UART2 using RS-485 to get the data from 3 phase industrial energy meter.

The goal of the project is to read the data from live 3 phase industrial energy meter (Secure Elite 443) using RS-485 MODBUS protocol. It will also read register addressed provided by the manufacturer and display that on serial terminal. It will also show the data of accelerometer sensor on serial terminal.

General Block Diagram


POWER SUPPLY
FRDM-KL25Z DEVELOPMENT BOARD MCU
I2C DRIVER
MMA8451Q ACCELEROMETER INERTIAL SENSOR
UART1/UART2
RS 485 ENERGY METER
TTL to RS-485 MODULE
COMMAND PROCESSOR - UART0

Implementation Details (Functionality my project will demonstrate)

System Configuration

The following is a list of the software modules I will develop for this project:

- System clock either 48MHz or 24Mhz to drive the I2C and UART peripherals
- Delay functions to general a request and response frame needed for the RS-485 bus protocol
- UART0 for command processing to take input from user “I2Cdata”, “EMData” and perform lexical analysis and call the required function to display the data on serial terminal

- I2C initialization to read the Inertial sensor data of X,Y and Z plane
- UART1 or UART2 for RS-485 data polling from 3 phase meter to develop request-response based algorithm to read data from Modbus meter and display on terminal

Example of request response frame

REQUEST FRAME – WILL BE SENT FROM MASTER (MCU) TO SLAVE (ENERY METER)

SLAVE ID	FUNCTION CODE	MEMORY ADDRESS– LOW BYTE	MEMORY ADDRESS – HIGH BYTE	NO OF REGISTERS TO READ – LOW BYTE	NO OF REGISTERS TO READ – HIGH BYTE	16 BIT CRC – LOW BYTE	16 BIT CRC – HIGH BYTE

Demo frame to read the voltage from energy meter is as follow

{0x01, 0x03, 0x00, 0x63, 0x00, 0x02, 0xEF, 0x37}

Decoding the above request frame we see

0x01 – Slave Id (Can be changed in energy meter)

0x30 – Holding register

0x00 0x63 – Register Address as per the Elite manual attached here Elite 440 mapping-1.pdf

Here I am trying to read the first address 40100 that is 100 in decimal so when we convert 100 into hex we get 0x64. But we still see we use 0x63. The reason for that is in many of the energy meters there are offset of -1 to get the data correctly. Hence we send the frame with 0x0063 instead of 0x0064

0x00 0x02 – Size of how many registers of data to read (Each register is 16 bit) so we read total of 4 bytes. Because the voltage data as per data sheet is 32bit float which is 4 bytes

0xEF 0x36 – 16 bit CRC calculation

RESPONSE FRAME – WILL BE SENT FROM SLAVE (ENERGY METER) TO MASTER (MCU)

{0x01, 0x03, 0x43, 0x71, 0xBA, 0XE1, 0xXX, 0xXX}

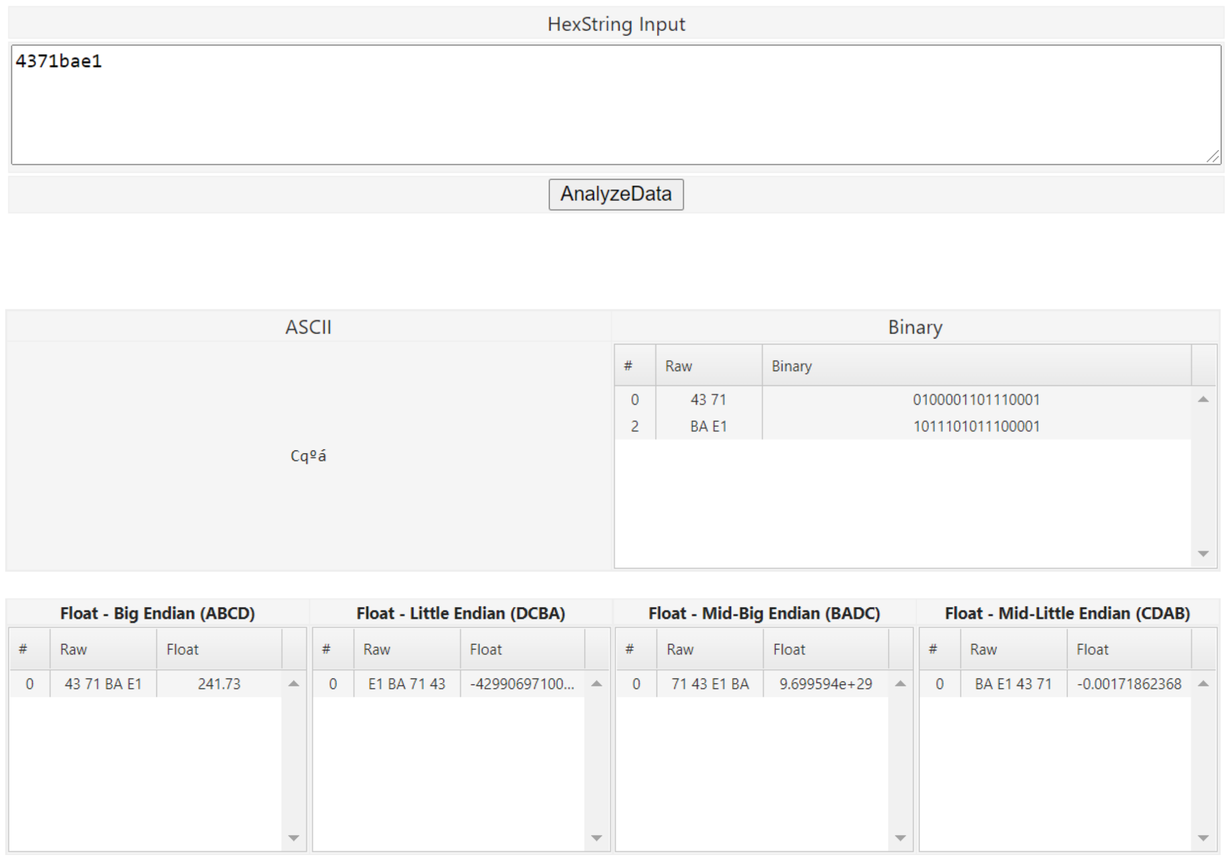
Decoding the above response frame we see

0x01 – Slave Id

0x30 – Holding register

0x43, 0x71, 0xBA, 0XE1 – 4 bytes of data of voltage

Conversion of data from hex to float is done using this link



```
4371bae1
```

ASCII

```
Cqºá
```

Binary

#	Raw	Binary
0	43 71	0100001101110001
2	BA E1	1011101011100001

Float - Big Endian (ABCD)

#	Raw	Float
0	43 71 BA E1	241.73

Float - Little Endian (DCBA)

#	Raw	Float
0	E1 BA 71 43	-42990697100...

Float - Mid-Big Endian (BADC)

#	Raw	Float
0	71 43 E1 BA	9.699594e+29

Float - Mid-Little Endian (CDAB)

#	Raw	Float
0	BA E1 43 71	-0.00171862368

Meter response can vary from manufacturer as the decoding works in either of the following byte placement (High byte first, High word first, etc)

0xXX 0xXX – 16 bit CRC from meter

Note: If the request frame CRC is wrong the meter won't respond.

Command processing (Reading input from user to read the values) – User interface

- Using the concept of A6, command processing technique will be created for the user to poll the data in interrupt mode. User can poll the I2C MMA data as well as Energy meter data by typing I2CData and EMData
- Displaying I2C accelerometer data either polling or interrupt. I2C driver will be developed to read the X,Y and Z axis data
- Displaying energy meter data using MAX485 IC from the meter using request response functions

Inertial Measurement using inbuilt accelerometer sensor – MMA8451Q I2C sensor

- Bare metal coding to initialize and read the accelerometer values

Serial communication

- Interfacing UART1 or UART2 with RS-485 module to read the meter data
- Developing UART driver to read and write the Modbus data (Read Holding Register)
- Develop 16 bit Modbus CRC checker to poll the data

Technologies will my project demonstrate

- Concept of command processor
- Concept of developing i2C driver
- UART0 driver to read user input – interrupt based
- UART1/2 driver for RS-485 protocol implementation

Anticipated learning to develop my project

- I2C driver development
- UART driver development
- 16 bit CRC development - Online CRC Calculator (tahapaksu.com)
- Modbus reference - Modbus Protocol (modbustools.com)
- Software sources - Modscan64 1.0 Download - ModScan64.exe (informer.com)
- Software sources - RTU/ASCII Master Test Software | Simply Modbus Software
- Difference between Modbus RTU and Modbus ASCII
- Internet of Things Industry

Additional Hardware used for my project

- USB to RS-485 converter
- TTL to RS-485 Module
- Secure Elite 3 phase industrial energy meter

I have access to all the extra hardware required for my project. I will have to test all of them because I am not sure if all are functional or not. If any of the hardware doesn't work, I will order the required hardware.

Testing Strategy

- Since this project includes many functions, it will use mix of automated and manual tests
- Automated test will include testing the sensor correction to data required including
- Manual test will include hardware switching test – shutting the sensor off, disconnecting and checking.