Tibber Embedded Firmware Developer Assignment

Introduction

In this assignment, you will be implementing an international standard IEC 62056-21 (Mode C) client on an MCU of your choice that can communicate with a metering device. The hardware application for this is typically an IR transmitter/receiver but the hardware implementation is out of this project's scope.

The format of the communication is a simplex async usart (1 start bit, 7 data bits, 1 parity bit, 1 stop bit).

Assignment

Here is a link to the full standard, you don't need to read it or implement it fully. Look at the example data included in this document instead. It's just here for reference.

https://drive.google.com/file/d/1-3BGqSyLafuhalZK1DnqKgK1UiJlwyL7/view?usp=sharing

The 5:th byte in the id indicates the baud rate for the data to be sent. To confirm that baud rate you need to reply with the same identifier in a request for data as in the example exchange we provided

Supported baud rates:
0: 300
1: 600
2: 1200
3: 2400
4: 4800
5: 9600

The goal is to send the volts and the amps (parameters 32.7, 52.7, 72.7, 31.7, 51.7, 71.7) to your computer/server in the format you think is best, using your medium of choice. See the example output at the end of this document.

Setup

- You can choose any microcontroller, language, and toolchain that you are familiar with. But something that could still be used in production.

Bonus Points:

- Write the code or at least plan in mind that your application will run on a battery-powered device.
- Plan for the application to run on an RTOS with other tasks in parallel.

This assignment is the base of discussions in the next interview step. Take notes of your thoughts and ideas you get so we can discuss them after when you present your code.

Try to scope your time, and stop when you think you have enough to show us even if not fully complete.

Send us the code over mail when ready.

If there are any questions don't hesitate to send us an e-mail at jimmy@tibber.com and dagfinn@tibber.com

Have fun!

Example data

```
Send hello to the meter (300 baud):
/?!CRLF

Receive ID (300 baud):
/LGZ4ZMF100AC.M29CRLF


Request meter data in (300 baud):
ACK040CRLF

Receive data (4800 baud):
STX
F.F(00)CRLF
1.8.0(000052.337*kWh)CRLF
2.8.0(000376.432*kWh)CRLF
3.8.0(000145.875*kvarh)CRLF
4.8.0(000010.724*kvarh)CRLF
15.8.0(000428.769*kWh)CRLF
32.7(231*V)CRLF
52.7(231*V)CRLF
72.7(231*V)CRLF
31.7(00.000*A)CRLF
51.7(00.000*A)CRLF
71.7(00.000*A)CRLF
13.7(-.--)CRLF
14.7(50.0*Hz)CRLF
C.1.0(40799390)CRLF
0.0(40799390        )CRLF
C.1.1(        )CRLF
0.2.0(M29)CRLF
16.7(000.00*kW)CRLF
131.7(000.00*kVAr)CRLF
C.5.0(6402)CRLF
C.7.0(0064)CRLF
!CRLF
ETX
BS
```

Example data in hex

Send:
2F3F210D0A0D0A

Receive:
2F4C475A345A4D4631303041432E4D32390D0A

Send:
063034300D0A

Receive:
02
462E46283030290D0A
312E382E30283030303035322E3333372A6B5768290D0A
322E382E30283030303337362E3433322A6B5768290D0A
332E382E30283030303134352E3837352A6B76617268290D0A
342E382E30283030303031302E3732342A6B76617268290D0A
31352E382E30283030303432382E3736392A6B5768290D0A
33322E37283233312A56290D0A
35322E37283233302A56290D0A
37322E37283233302A56290D0A
33312E372830302E3030302A41290D0A
35312E372830302E3030302A41290D0A
37312E372830302E3030302A41290D0A
31332E37282D2E2D2D290D0A
31342E372835302E302A487A290D0A
432E312E3028343037393933930290D0A
302E30283430373939333930202020202020202020290D0A
432E312E31282020202020202020290D0A
302E322E30284D3239290D0A
31362E37283030302E30302A6B57290D0A
3133312E37283030302E30302A6B564172290D0A
432E352E302836343032290D0A
432E372E302830303634290D0A
210D0A
03
08

Example output in json

```
{
      “ID”: “LGZ4ZMF100AC.M29CRLF”,
      “L1”: {
            “volt”: 231.0,
            “amp: 0.0
      },
      “L2”: {
            “volt”: 231.0,
            “amp: 0.0
      },
      “L3”: {
            “volt”: 231.0,
            “amp: 0.0
      },
}
```