LP pulse water metering modem.

Data Frames. Ver 1.3

The Notification Data frame sent by modem is of the form:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Frame Start | Type | Modem ID | Status | Payload Size | Payload | CRC16 | Frame End |
| 1 byte | 1 byte | 4 bytes | 1 byte | 2 bytes | …….. | 2 bytes | 1 byte |
| 0x7E |  |  |  |  |  | CCITT | 0x7E |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Cover Opened | SIM removed | Low Power | 0 | 0 | 0 | 0 | 0 |

Long and short integer are stored with the little endian arrangement.

“Type” field will take “0” for DeviceInfo, “1” for PR7 Data and “2” for MBus Data.

Modem Status contains the following signal:

CRC16 is a of type CCITT (0x1021) calculated for all the frame except the frame End.

So far, we defined 3 types of payload:

1. DeviceInfo payload sent whenever modem starts (sent every startup), that contains modem and device info parameters,

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Time Stamp | IMEI | Modem Type | SIM ID | IMSI | Version | IP Adress | Local Port |
| 4 Bytes | 16 chars | 16 | 20 | 16 chars | 8 bytes | 16 chars | 2 bytes |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Reset count | Battery Voltage | Brand | Status | Signal Strength | Meter Type | Meter Count |
| 2 bytes | 2 bytes | 1 byte | 1 byte | 1 byte | 1 byte | 1 byte |

Meter Type: “0xFF” for MBus meter else for Pulse counter. 0x00 = K1:1, 0x01 = K1:10, 0x11 = K10:10,

0x02= K1:100, 0x12 = K10:100… etc

1. PR7 data payload:

is an hourly sampling data. It is usually 24 samples per day (per transfer)

|  |  |
| --- | --- |
| Sample count | Sampling Data Frames |
| 1 Byte usually = 24 | 24 frames = 17\*24 bytes |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Frame (Fixed size 17 Bytes) | | | | |
| Time Stamp | Forward Pulses | Reverse Pulses | Compensated Pulses | CTR |
| 4 Bytes | 4 Bytes | 4 Bytes | 4 Bytes | 1 Byte |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CTR field | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Passive | 0 | 0 | 0 | 0 | 0 | 0 | Tamper/Battery empty |

**Usually:**

Forward Pulses – Reverse Pulses = Compensated pulses.

PR7/6 pulse generator contains 4 signals

Pulse1P indicate a real flow where Pulse1D indicate the direction of the flow, from those signals Modem counts Forward Pulses and reverse pulses,

Pulse2 counts only the compensated flow (Forward - Reverse) where Pulse2C indicates if there is compensation or not. I do not use this signal in the data.

1. MBus Data payload,

Modem could be attached to multiple MBus meters, each meter should be read every hour, so there are usually 24 samples for each meter per day (per transfer).

The following frame represents sampling data array for multiple MBus meters, the first field indicates the overall data samples contained in the frame;

|  |  |  |  |
| --- | --- | --- | --- |
| Sample count | MBus Data sample 1 | MBus Data sample 2 | Etc … |
| 1 Byte | About 64 bytes |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MBus Data (non-fixed size) | | | | |
| Time Stamp | Meter ID | Length L | Data | Spacer |
| 4 Bytes | 1 byte | 1 Byte | L bytes | 0X7C |

II. Server Response & Command

After receiving of a data notification from the modem, server will respond by a UDP frame that contains a result of processing the notification and a command if user wants to perform a command to the modem. The response has the following format:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Start | Error Code | Accepted Samples | Time stamp | Cmd Size | Command | BCC | End |
| 0x7E | 1 byte | 2 bytes | 4 bytes | 2 bytes | N bytes | 1 byte | 0x7E |

Frame Start & End : 0x7E

Accepted Samples: is the number of correct received data samples, It is for future use.

Error code: indicate the error occurred during notification parsing,

sr\_Err\_None =0 // OK

sr\_Err\_Truncated =1 // uncompleted notification

sr\_Err\_Size =2 // error in payload size or data size

sr\_Err\_CRC16 =3 // wrong crc16

Command: is a char string, its size stored in the command size field. Multiple commands can be included by separating them by “\n” character.

Examples: will be shared later

Appendix: The data structure definition

#define SERVER\_FRAME\_VERSION 1.3

typedef struct {

uint8\_t CoverOpened:1;

uint8\_t SIMRemoved:1;

uint8\_t LowPower:1;

//uint8\_t Reserved:5;

}TModemStatus;

typedef enum{

sft\_DeviceInfo =0

,sft\_PR7Data =1

,sft\_MBusData =2

} TServerFrameType;

//Notification Frame

typedef struct

{

uint8\_t StartChar; // Frame Start

TServerFrameType Type; // Frame Type

uint32\_t ModemID; // Modem Identification Number

TModemStatus Status;

uint8\_t\* Payload;

uint16\_t PayloadSize; // Frame size in bytes

uint16\_t CRC16;

uint8\_t EndChar; // Frame End

} TServerFrame;

//DeviceInfo Notification

typedef struct

{

char IMEI[16];

char ModemType[16];

char SIMID[20];

char IMSI[16];

char Version[8];

char IP[16];

uint16\_t LocalPort;

uint16\_t ResetCount;

uint16\_t BatteryVoltage;

uint8\_t Brand;

TModemStatus Status;

uint8\_t SignalStrength; //CSQ

uint8\_t MeterType;

uint8\_t MeterCount;

// etc...

} TDeviceInfoPayload;

typedef struct

{

uint32\_t TimeStamp; //sampling Time value

uint32\_t FPulse1; //Forward pulse counter

uint32\_t RPulse1; //Reverse Pulse counter

uint32\_t Pulse2; //OverAll (Compensated) Pulse counter

uint8\_t Tamper; //Tamper Flag

}TPR7Record;

typedef struct

{

uint8\_t RecordCount; //usually 24

TPR7Record Record[…];

} TPR7Payload;

typedef struct

{

uint32\_t TimeStamp;

uint8\_t MeterID;

uint8\_t Size; // Size < 128-8

uint8\_t Buf[Size]; //TODO: later, we will configure the record size

uint8\_t Spacer; //0x7C

}TMBusRecord;

typedef struct

{

uint8\_t RecordCount; //usually 24 for each meter

TMBusRecord Record[…]; //only for one meter

} TMBusPayload;

typedef enum{

sr\_Err\_None =0

,sr\_Err\_Truncated =1

,sr\_Err\_Size =2

,sr\_Err\_CRC16 =3

} TSrvResponseErr;

//Notification response & Command Frame

typedef struct

{

uint8\_t Start; // Frame reference number

TSrvResponseErr Error; // Error kind

uint16\_t AcceptedSamples; // how much samples are correct counting from start of the sample list

uint16\_t TimeStamp; // Server Time

uint16\_t CmdSize; // Req size in bytes

uint8\_t\* Cmd;

uint8\_t BCC; // Block Check Character

uint8\_t End; // Frame End = 0x7E

} TSrvResponse;