RACOM
RADIO DATA NETWORKS

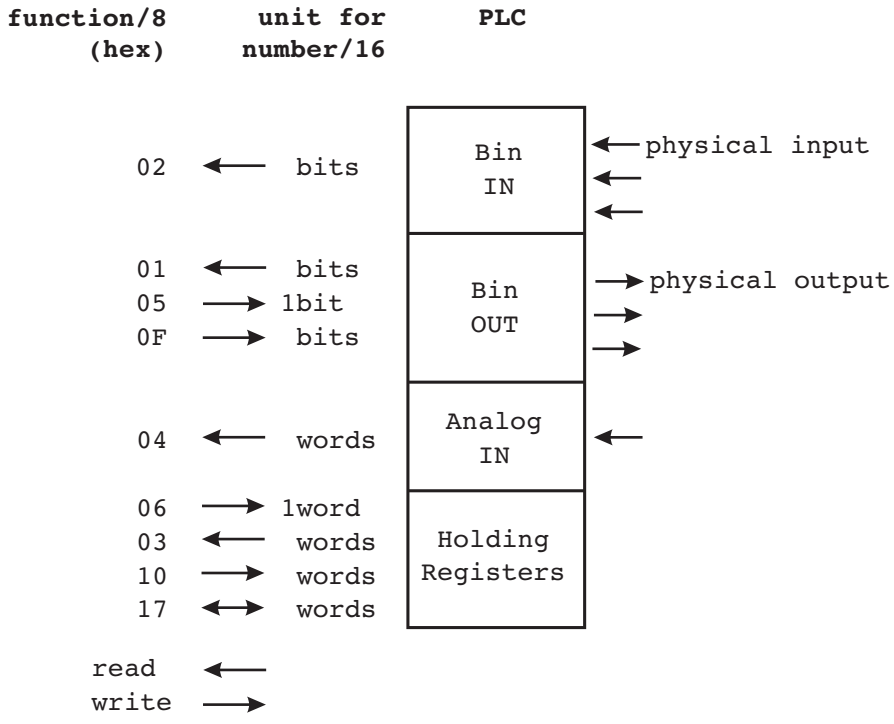
Format of MODBUS frame for MORSE

verzion x.xx
1/12/2011

1. Úvod

Modbus is a typical example of a family of protocols determined for buses implemented on RS485. It uses a 256 byte frame equipped with a 16-bit CRC. Because Modbus distinguishes the types of transmitted data (bit, byte, word), types of frames are established for distinguishing these variations. The Modbus type of frame is described by the number of the function implemented by the frame.

2. Overview of Modbus commands for reading and writing from various parts of the PLC memory:



3. Description of individual functions:

3.1. - 01 - (Read Output Status)

Reads output states (relays, transistor switches, etc.) from the Slave.

The basic unit read by means of this function is one output – for us this means one bit. Of course as the protocol is able to transfer one byte as the smallest element output bits are combined into bytes. The Master can address more bits at once. It then obtains them stacked into several bytes of the reply.

Typical forms of frames then look as follows:

request

```
| adr/8 | fce/8 | start/16 | number/16 | crc/16 |
```

reply

```
| adr/8 | fce/8 | cnt/8 | data/8 * cnt | crc/16 |
```

adr - address of PLC on Modbus, this address must be unique within the whole bus. The address is the same for a request as for a reply.

fce - function, which the PLC performs after receiving the frame

start - start address of data (output), which is to be processed

number - number of items (bits), which are to be processed

cnt - number of frame data bytes

data - actual data of the request aligned into 8 bits

adr - security word

Example:

If we have the state of outputs in our PLC from the zero address 0x1480. The PLC has an address on the Modbus of 0x10. Using the function 01 first we read everything at once and then the second part only.

request for the first case:
`1001 0000 0010 crc` – we want 16 outputs from the zero position reply:

odpověď:
`1001 0214 80 crc` – all 16 outputs returned. It is necessary to point out that data is not aligned into an even number of bytes as is common in MORSE networks.

request for the second case:
`1001 0008 0008 crc` – we want 8 outputs from the eighth position

reply:

`1001 0180 crc` – 8 outputs returned. As is seen outputs are organised in Intel format (little endian).

3.2. - 02 - (Read Input Status)

It is absolutely identical to the previous function, but it reads inputs from the Slave PLC. The form of frames are also the same.

3.3. - 03- (Read Holding Registers)

16 bit memory registers return from the PLC. These registers are general purpose. In our case they are used for cache and packet mode. The frames then look as follows:

request

```
| adr/8 | fce/8 | start/16 | number/16 | crc/16 |
```

reply

```
| adr/8 | fce/8 | cnt/8 | data/8 * cnt | crc/16 |
```

adr - address of the PLC on the Modbus

fce - function, which the PLC performs after receiving the frame

start - start address of data (output), which is to be processed

number - number of items (words), which are to be processed

cnt - number of frame data bytes

data - content of required registers aligned into 16 bits

crc - security word

Example:

In our PLC from the previous example the contents of three registers from the zero address are `0x1480, 0x3450` and `0x4054`.

request:
`1003 0000 0003 crc`

reply:
`1003 0614 8034 5040 54 crc` – again the frame is not aligned into an even number of bytes.

3.4. - 04- (Read Input Registers)

In principle this is the same as the function for reading registers. The difference of course is that it returns the states of analog inputs.

3.5. - 05- (Force Single Output)

It sets one output, i.e. one bit. As the binary output can only be set or deleted the forms of commands are very simple.

request
`| adr/8 | fce/8 | start/16 | 0xFF00 | crc/16 |` – for setting 1 output
`| adr/8 | fce/8 | start/16 | 0x0000 | crc/16 |` – for deleting 1 output

reply
`| adr/8 | fce/8 | start/16 | 0xFF00 | crc/16 |` or

`| adr/8 | fce/8 | start/16 | 0x0000 | crc/16 |` – it is a simple copy of the request.

adr - address of the PLC on the Modbus

fce - function, which the PLC performs after receiving the frame

start - start address of data (output), which is to be processed

crc - security word

3.6. - 06- (Preset Single Register)

Sets the content of one register = 1 word. It is a similar function to the previous one however the state of the register appears instead of the state of the bit.

request
`| adr/8 | fce/8 | start/16 | data/16 | crc/16 |`

reply
`| adr/8 | fce/8 | start/16 | data/16 | crc/16 |` – again it is a simple copy of the request

adr - address of the PLC on the Modbus

fce - function, which the PLC performs after receiving the frame

start - start address of data which is to be processed

data - content of written register

crc - security word

3.7. - 0F hex- (Force Multiple Outputs)

Simultaneous setting of more outputs.

request
`| adr/8 | fce/8 | start/16 | number/16 | cnt/8 | data/8 * cnt | crc/16 |`

reply
`| adr/8 | fce/8 | start/16 | number/16 | crc/16 |`

adr - address of the PLC on the Modbus

fce - function, which the PLC performs after receiving the frame

start - start address of data (output), which is to be processed

number - number of bits for writing

cnt - number of bytes necessary for transmission of the addressed group of bits

data - states of written outputs, words contain swapped bytes in the form L,H, L,H, L,H, ...

crc - security word

3.8. - `10 hex-` (Preset Multiple Regs)

Similar to the previous function it sets more registers at once.

request
```
| adr/8 | fce/8 | start/16 | number/16 | cnt/8 | data/16 * numb | crc/16 |
```

reply
```
| adr/8 | fce/8 | start/16 | number/16 | crc/16 |
```

adr - address of the PLC on the Modbus

fce - function, which the PLC performs after receiving the frame

start - start address of data (output), which is to be processed

number - number of words for writing

cnt - number of bytes necessary for the transmission of the required group of words

data - states of written registers

crc - security word

3.9. – `17 hex–` (READ/WRITE HOLDING REGISTERS)

Joins the functions 03 read and 06 write.

request
```
| a/8 | f/8 | rst/16 | rno/16 | wst/16 | wno/16 | wcnt/8 | wdata/cnt*8
|crc/16 |
```

reply
```
| a/8 | f/8 | cnt/8 | data/cnt*8 | crc/16 |
```

a - slave address

f - function READ/WRITE HOLDING REGISTERS

rst - start of read area

rno - number of registers of the read area

wst - start of written area

wno - number of registers of written area

wcnt - number of bytes of data in the frame

wdata - data that should be written

If the Slave PLC does not understand something in a request/command it is obligated to return an exception. The exception should inform the Master about the situation in which it tried to work with either an unauthorised function for the given Slave or with data outside of the valid range for the given Slave PLC.

- reply to the request or command entered incorrectly

```
| adr/8 | 0x80+fce /8 | excode/8 | crc/16 |
```

adr - address of the PLC on the Modbus

fce - function, which develops the exception

excode - exception number, specifies where exactly and what error occurred

- 1 - erroneous function number
- 2 - erroneous data address
- 3 - erroneous data content
- 5 - confirmed receipt of command, whose execution is slow
- 6 - rejection, Slave is busy with the function of a slower command

crc - security word