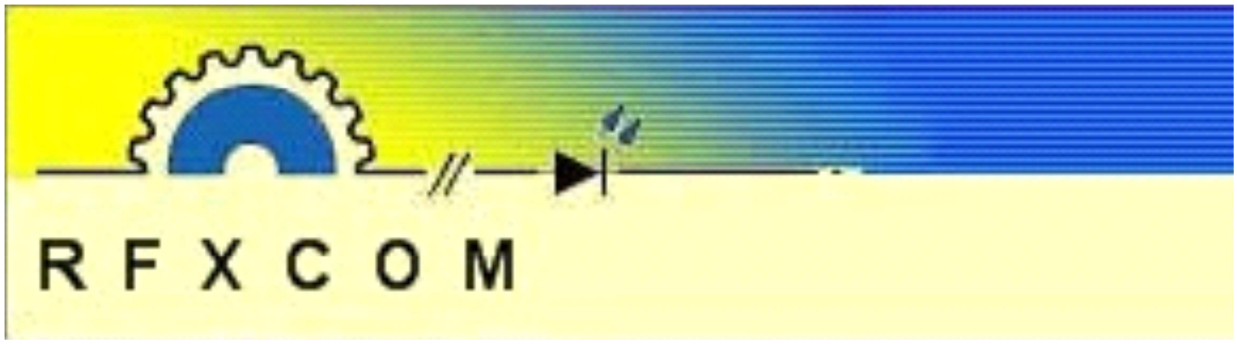


RFXtrx

Software Development Kit

www.rfxcom.com

1. Table of Contents

2. Protocol License

IMPORTANT: read the Copyright message at the end of this document.

3. Warning:

RF signals are possible disturbed and it has not been justified for this equipment at uses in circumstances where life-threatening or dangerous situations are possible.

4. Introduction

This document describes the communication protocols for the RFXtrx transceivers.
The RFXtrx315 and RFXtrx433 are available.
The RFXtrx868 is in development.
The RFXmngr program has implemented all functions to translate the received commands and to send commands to the device.
The VB.NET source of the RFXmngr program is available for free on request.

5. LAN interface **(not yet available)**

The LAN interface can be flashed with TCP/IP or xPL (UDP) firmware.
This document describes the interface with the use of TCP/IP. For use of the xPL protocol see the document “RFXCOM implementation xPL”.

Communication with the 433.92MHz transceiver runs over TCP/IP - IP port 10001.

Communication with the I/O port runs over TCP/IP - IP port 30704.

6. USB interface

The USB interface is using a COM Port Redirector with 38400 baud, 8 bits, No parity, 1 stop bit.

7. Bootloader

The RFXflash program has to be used to flash the transceiver or receiver.

The transceiver/receiver can enter the bootload mode in three different ways:

1. A bootload command is received within 5 seconds after power up,
2. A bootload command is received during normal operation,
3. Force bootload mode. Connect the BOOT point on the transceiver/ receiver PCB to ground and power up the transceiver/receiver.

Normally step 2 can be used. But if program memory has been destroyed by an incomplete flash or another unforeseen reason step 1 can be used.
If step 1 and 2 do not work step 3 must be used to force the device to bootload mode. After the flash operation remove the connection between BOOT and ground and power cycle the device to start normal operation mode.

8. Implemented RF protocols

See the RFXtrx User guide.

9. Transceiver/receiver message format

There are 3 kinds of messages. These are:

- Configuration and other commands for the interface,
- Messages to the transmitter
- Messages from the receiver

Every message consists of 5 fields:

- "Packet-Length" 1 byte that gives the number of bytes that will follow.
- "Packet-Type" 1 byte that indicates the message type.
- "Packet-Subtype" 1 byte that indicates the message sub type.
- "Sequence number" 1 byte for a sequence number so that the application knows which response belongs to which command sent. If not used by the application, leave it zero.
  Receiver messages have their own sequence number.
- "Packet-Data" 1 or more bytes with the information.

Packet length in bytes of: Packet-Type + Packet Subtype + Sequence number + Packet data	Packet Type	Packet Subtype	Sequence number	Packet Data

Some messages of the receiver and transmitter are identical or almost identical.
For example, a message from an X10 remote is almost identical to the message for the transmitter. The RSSI and battery level fields are present in both messages but are not used by the transmitter.

10. Communication protocol with the application

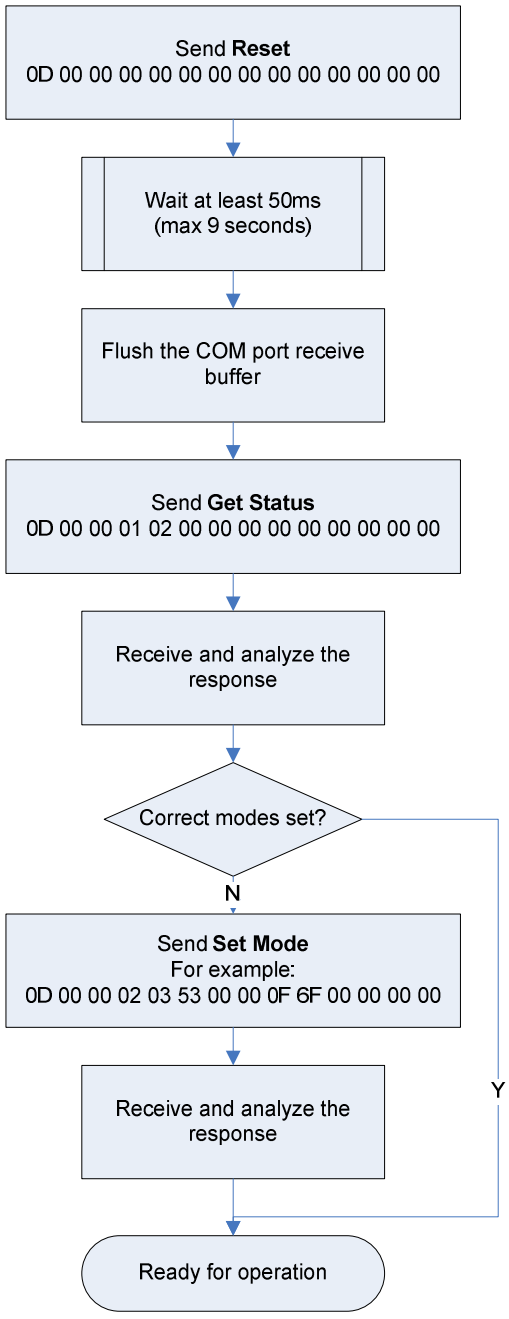
This is the procedure to start the communication between the application and the RFXCOM transceiver:

- Send a 14 byte Interface command – Reset:
  packet (hex 0D 00 00 00 00 00 00 00 00 00 00 00 00 00)
  The RFXCOM will now stop the RF receive for 10 seconds. This period is terminated by sending a Status Request.
- Wait at least 50 milliseconds (max 9 seconds) than clear the COM port receive buffers.
- Send a 14 byte Interface command – Get Status:
  packet (hex 0D 00 00 01 02 00 00 00 00 00 00 00 00 00)
  The RFXCOM will respond with the status and the 10 seconds reset timeout is terminated.
- If necessary send a select frequency selection command. The 433.92MHz transceiver does not have a frequency select and operates always on 433.92MHz.

The RFXtrx is now ready to receive RF data and to receive commands from the application for the transmitter.

IMPORTANT
The command buffer can hold a maximum of 400 bytes which is equivalent to about 40 commands.
Never send more than these 400 bytes without waiting for a response on the commands send.
All Interface commands (except reset) and all transmitter commands will be acknowledged by a response.

10.1. Flowchart of the initialization protocol



10.2. 0x00: Interface Control

The RFXCOM interface is controlled by control commands. All commands, except Reset, will be responded by the interface with status message.

10.2.1. 0x00: Mode Command

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   cmnd;
        BYTE   msg1;   // Select receiver/transceiver frequency
        BYTE   msg2;   // RFU
        BYTE   msg3;   // mode select bits
        BYTE   msg4;   // mode select bits
        BYTE   msg5;   // mode select bits
        BYTE   msg6;   // RFU
        BYTE   msg7;   // RFU
        BYTE   msg8;   // RFU
        BYTE   msg9;   // RFU
} ICMND;
```

packetlength:
Packet length (this byte not included) = 0x0D

packettype:
0x00 = interface command

subtype:
0x00 = mode command

seqnbr:
Sequence number, this is not used by the RFXCOM. This can be used in the application to synchronize the commands sent with response messages. If not used in the application leave it zero.

cmnd:
0x00 = Reset the receiver/transceiver. No answer is transmitted!
0x01 = not used.
0x02 = Get Status, return firmware versions and configuration of the interface.
0x03 = Set Mode msg1-msg5, return also the firmware version and configuration of the interface.
0x06 = save receiving modes of the receiver/transceiver in non-volatile memory
0x07 = not used

0x08 = T1 – for internal use by RFXCOM
0x09 = T2 – for internal use by RFXCOM

0x50 = select 310MHz in the 310/315 transceiver
0x51 = select 315MHz in the 310/315 transceiver
0x52 = not used
0x53 = not used
0x54 = not used
0x55 = select 868.00MHz in the 868 transceiver
0x56 = select 868.00MHz FSK in the 868 transceiver
0x57 = select 868.30MHz in the 868 transceiver
0x58 = select 868.30MHz FSK in the 868 transceiver
0x59 = select 868.35MHz in the 868 transceiver
0x5A = select 868.35MHz FSK in the 868 transceiver
0x5B = select 868.95MHz in the 868 transceiver

msg1 – msg5:
These bytes are only used by a 03 - Set Mode command.

msg1:
Select receiver/transceiver frequency.
The 4 RFXtrx types use different hardware.
It is for example not possible to select 315MHz in an RFXtrx433.

RFXtrx315:
0x50 = 310MHz
0x51 = 315MHz
RFXrec433:
0x52 = 433.92MHz
RFXtrx433:
0x53 = 433.92MHz
RFXtrx868:
0x55 = 868.00MHz
0x56 = 868.00MHz FSK
0x57 = 868.30MHz
0x58 = 868.30MHz FSK
0x59 = 868.35MHz
0x5A = 868.35MHz FSK
0x5B = 868.95MHz

msg2:
not used

msg3-msg5:
Select operation modes

msg3:
Set a bit to 1 to enable the protocol in the RFXtrx

bit	7	6	5	4	3	2	1	0
protocol	Enable display of undecoded	RFU6	Byron SX	RSL	Lighting4	FineOffset/Viking	Rubicson	AE Blyss
			433.92	433.92	433.92	433.92	433.92	433.92

msg4:
Set a bit to 1 to enable the protocol in the RFXtrx

bit	7	6	5	4	3	2	1	0
protocol	BlindsT1/T2/T3/T4	BlindsT0	ProGuard	FS20	La Crosse	Hideki/UPM	AD LightwaveRF	Mertik
	433.92	433.92	868.35 FSK	868.35	433.92 / 868.30	433.92	433.92	433.92

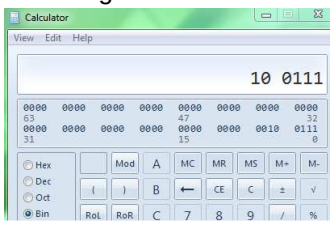
msg5:
Set a bit to 1 to enable the protocol in the RFXtrx

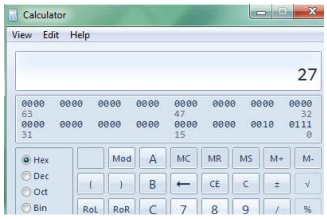
bit	7	6	5	4	3	2	1	0
protocol	Visonic	ATI	Oregon Scientific	Meiantech	HomeEasy EU	AC	ARC	X10
	315 / 868.95	433.92	433.92	433.92	433.92	433.92	433.92	310 / 433.92

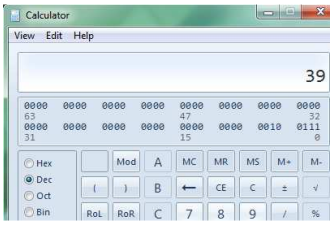
Example of a Set Mode command:
433.92MHz
Display un-decoded messages
Protocols Enabled: La Crosse, Oregon, AC, ARC and X10
0D 00 00 00 03 53 00 80 08 27 00 00 00 00

msg6 – msg9:
not used

Explanation: The Set mode command needs the mode bits encoded in msg3, 4 and 5.
To calculate the values for msg3, 4 and 5 you can use for example the Windows calculator in View - programmer mode.
As an example for msg5 enable Oregon, AC, ARC and X10.
Bit configuration is: 0 0 1 0 0 1 1 1 = hex 27 and if you want to use decimal this is 39







10.2.2. Remarks on Interface Control commands

Acknowledges
Every command, except reset, will be acknowledged by an Interface Response packet. (packettype 0x01, subtype 0x00)

Reset the receiver/transceiver
The reset command will reset the receiver and transmitter and it stops the receiver for 10 seconds. Receiving modes are set from non-volatile memory.
No response packet is transmitted on a reset command. This allows the application to flush the receive buffers after a few seconds before it starts processing received data.
Commands transmitted after the reset command are accepted and acknowledged and the 10 seconds time out is stopped.

Set Mode
Modes can be set using individual commands or using the Set mode command. This command is normally used by the application during start-up.

Save receiving modes
To configure the receiver for your purpose send an “enable all receiving modes” and after that disable the protocols not used. After this send the “save receiving modes” command to save this configuration in non-volatile memory. This configuration is loaded after power up and reset.
Important: Do not send the save command very often because there is a maximum of 10,000 write cycles to non-volatile memory!

10.3. 0x01: Interface Message

Responses or status messages send by the interface.

10.3.1. 0x00, 0xFF: Response on a command

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   cmnd;
        BYTE   msg1;
        BYTE   msg2;
        BYTE   msg3;
        BYTE   msg4;
        BYTE   msg5;
        BYTE   msg6;
        BYTE   msg7;
        BYTE   msg8;
        BYTE   msg9;
} IRESPONSE;
```

packetlength:
Packet length (this byte not included) = 0x0D

packettype:
0x01 = interface message

subtype:
0x00 = response on a mode command.
0xFF = wrong command received from the application

seqnbr:
Sequence number, this is not used by the RFXCOM receiver or transmitter. This field can be used in the application to synchronize the commands sent with response messages.

cmnd:
This field contains the command type to which this is the response.
See Interface Control – Mode command for the list of command codes.

msg1:
receiver/transceiver type
0x50 = 310MHz
0x51 = 315MHz
0x52 = 433.92MHz receiver only
0x53 = 433.92MHz transceiver
0x55 = 868.00MHz
0x56 = 868.00MHz FSK
0x57 = 868.30MHz
0x58 = 868.30MHz FSK
0x59 = 868.35MHz
0x5A = 868.35MHz FSK
0x5B = 868.95MHz

msg2:
Firmware version of the transceiver / receiver firmware

msg3:
Enabled protocols in the transceiver / receiver

bit	7	6	5	4	3	2	1	0
protocol	Enable display of undecoded	RFU6	Byron SX	RSL	Lighting4	FineOffset/Viking	Rubicson	AE Blyss

msg4:
Enabled protocols in the transceiver / receiver

bit	7	6	5	4	3	2	1	0
protocol	BlindsT1	BlindsT0	ProGuard	FS20	La Crosse	Hideki/UPM	AD LightwaveRF	Mertik

msg5:
Enabled protocols in the transceiver / receiver

bit	7	6	5	4	3	2	1	0
protocol	Visonic	ATI	Oregon Scientific	Meiantech	HomeEasy EU	AC	ARC	X10

Example wrong command received:

```
-----------------------------------------------
0D01FF0241533E000C2F01000000
Packettype        = Interface Message
subtype           = Wrong command received from application
Sequence nbr      = 2
```

10.4. 0x02: Receiver/Transmitter Message

Responses or messages send by the receiver or transmitter.

10.4.1. 0x00 – 0x01: messages

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   msg;
} RXRESPONSE;
```

packetlength:
Packet length (this byte not included) = 0x04

packettype:
0x02 = receiver/transmitter message

subtype:
0x00 = error, receiver did not lock
        msg not used

0x01 = transmitter response.
        msg 0x00 = ACK, transmit OK
        msg 0x01 = ACK, but transmit started after 3 seconds delay anyway with RF receive data
        msg 0x02 = NAK, transmitter did not lock on the requested transmit frequency
        msg 0x03 = NAK, AC address zero in id1-id4 not allowed

seqnbr:
Sequence number, this is not used by the RFXCOM. This can be used in the application to synchronize the commands sent with response messages. If not used in the application leave it zero.

10.5. 0x03: Undecoded RF Message

Undecoded RF message. (for internal use only)

10.5.1. 0x00 – 0x12: message type

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   msg1;
        …
        BYTE   msg33;
} UNDECODED;
```

packetlength:
Packet length (this byte not included) = max 36 (depends on number of msg bytes)

packettype:
0x03 = undecoded RF message

subtype:
0x00 = ac
0x01 = arc
0x02 = ati
0x03 = hideki/upm
0x04 = lacrosse/viking
0x05 = ad
0x06 = mertik
0x07 = oregon1
0x08 = oregon2
0x09 = oregon3
0x0A =proguard
0x0B = visonic
0x0C = nec
0x0D = fs20
0x0E = reserved
0x0F = blinds
0x10 = rubicson
0x11 = ae
0x12 = fineoffset

seqnbr:
Sequence number, this is not used by the RFXCOM. This can be used in the application to synchronize the commands sent with response messages. If not used in the application leave it zero.

10.6. 0x10, Lighting1

10.6.1. 0x00-0x09: X10, ARC, ELRO, Waveman, EMW200, IMPULS, RisingSun, Philips, Energenie, GDR2

Used by: transmitter (only X10 and ARC are received)

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   housecode;
        BYTE   unitcode;
        BYTE   cmnd;
        BYTE   filler  : 4;
        BYTE   rssi : 4;
} LIGHTING1;
```

packetlength:
Packet length (this byte not included) = 0x07

packettype:
0x10 = lighting1

subtype:
0x00 = X10 lighting
0x01 = ARC
0x02 = ELRO AB400D (Flamingo)
0x03 = Waveman
0x04 = Chacon EMW200
0x05 = IMPULS
0x06 = RisingSun
0x07 = Philips SBC
0x08 = Energenie ENER010
0x09 = Energenie 5-gang
0x0A = COCO GDR2-2000R

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

housecode:
0x41 to 0x50 = house code A to P
Except:
EMW200 0x41 to 0x43 = house code A, B, C
RisingSun, COCO 0x41 to 0x44 = house code A, B, C, D

unitcode:
0x01 to 0x10 = unit 1 to 16
Except:
AB400 and IMPULS = unit 1 to 64,
EMW200, RisingSun, Energenie, COCO = unit 1 to 4
Philips = unit 1 to 8
Energenie 5-gang = unit 1 to 10

cmnd:

	AB400	Wave man	EMW 200	Impuls	Rising Sun	Energ enie5	CoCo
Off	0x00	0x00	0x00	0x00	0x00	0x00	0x00
On	0x01	0x01	0x01	0x01	0x01	0x01	0x01

	X10	ARC	Philips	Ener genie
Off	0x00	0x00	0x00	0x00
On	0x01	0x01	0x01	0x01
Dim	0x02			
Bright	0x03			
All/group Off	0x05	0x05	0x05	0x05
All/group On	0x06	0x06	0x06	0x06
Chime		0x07*		
Illegal cmnd received	0xFF	0xFF		

* unit code is not used.

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

ARC protocol
ARC is the protocol used by different brands with units having code wheels A-P/1-16:
KlikAanKlikUit, NEXA, CHACON, HomeEasy, Proove, DomiaLite, InterTechno, AB600

Example X10 lighting

```
----------------------------------------------
071000B7490A0160
Packettype    = Lighting1
subtype       = X10
Sequence nbr  = 183
housecode     = I
unitcode      = 10
Command       = On
Signal level  = 6
----------------------------------------------
071000E0490C0060
Packettype    = Lighting1
subtype       = X10
Sequence nbr  = 224
housecode     = I
unitcode      = 12
Command       = Off
Signal level  = 6
```

Example ARC

```
----------------------------------------------
0710010E430E0180
Packettype    = Lighting1
subtype       = ARC
Sequence nbr  = 14
housecode     = C
unitcode      = 14
Command       = On
Signal level  = 8
```

10.7. 0x11, Lighting2 (AC,HEU)

10.7.1. 0x00-0x02: AC, HomeEasy EU, ANSLUT

Used by: transmitter and receiver

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1 : 2;
        BYTE   filler1 : 6;
        BYTE   id2;
        BYTE   id3;
        BYTE   id4;
        BYTE   unitcode;
        BYTE   cmnd;
        BYTE   level;
        BYTE   filler2  : 4;
        BYTE   rssi : 4;
} LIGHTING2;
```

packetlength:
Packet length (this byte not included) = 0x0B

packettype:
0x11 = lighting2

subtype:
0x00 = AC
0x01 = HomeEasy EU
0x02 = ANSLUT

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1 – id4:
remote/switch/unit ID. (id1 is high byte)
The unit ID can be 0x00 00 00 01 to 0x03 FF FF FF

unitcode:
0x01 to 0x10 = unit 1 to 16

cmnd:
0x00 = Off
0x01 = On
0x02 = set level
0x03 = group Off
0x04 = group On
0x05 = Set group level

level:
0x0 to 0xF = 0% to 100% dim level

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

AC protocol
AC is the protocol used by different brands with units having a learning mode button:
KlikAanKlikUit, NEXA, CHACON, HomeEasy UK

Example AC (Chacon/NEXA/HomeEasy/KlikAanKlikUit/Intertechno…)

```
----------------------------------------------
0B11000600109B520B000080
Packettype    = Lighting2
subtype       = AC
Sequence nbr  = 6
ID            = 0109B52
Unit          = 11
Command       = Off
Signal level  = 8
```

10.8. 0x12, Lighting3

10.8.1. 0x00-: Koppla

Used by: transmitter

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   system;
        BYTE   channel8_1;
        BYTE   channel10_9;
        BYTE   cmnd;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} LIGHTING3;
```

packetlength:
Packet length (this byte not included) = 0x08

packettype:
0x12 = lighting3

subtype:
0x00= Ikea Koppla

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

system:
0x0 to 0xF = system code 1 to 16

channel8_1:
0x0 to 0xFF = bit for channel 8 7 6 5 4 3 2 1

channel10_9:
0x0 to 0x03 = bit for channel x x x x x 10 9

cmnd:

Bright	= 0x00
Dim	= 0x08
On	= 0x10
level 1	= 0x11
level 2	= 0x12
level 3	= 0x13
level 4	= 0x14
level 5	= 0x15
level 6	= 0x16
level 7	= 0x17
level 8	= 0x18
level 9	= 0x19
Off	= 0x1A
Program	= 0x1C

battery_level: (is 0x0 for transmitter command)
0x9 = full
0x0 = empty

rssi: (is 0x0 for transmitter command)
Signal strength. 0x0 to 0xF = weak to strong

10.9. 0x13, Lighting4

10.9.1. 0x00: PT2262

Used by: transmitter and receiver

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   cmd1;
        BYTE   cmd2;
        BYTE   cmd3;
        BYTE   pulseHigh;
        BYTE   pulseLow;
        BYTE   filler : 4;
        BYTE   rssi : 4;
} LIGHTING4;
```

packetlength:
Packet length (this byte not included) = 0x09

packettype:
0x13 = lighting4

subtype:
0x00 = PT2262

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

cmd1, cmd2, cmd3:
24 bits for PT2262

pulseHigh, pulseLow:
Pulse timing / 4.
This value is mostly 1400 / 4 = 350 (usec.)

rssi: (is 0x0 for transmitter command)

10.10. 0x14, Lighting5 (AD)

10.10.1. 0x00-0x04: LightwaveRF, Siemens, EMW100, BBSB, MDREMOTE, RSL2, Livolo, RGB

Used by: transmitter and receiver (EMW100, MDREMOTE and Livolo transmit only)

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   id3;
        BYTE   unitcode;
        BYTE   cmnd;
        BYTE   level;
        BYTE   filler  : 4;
        BYTE   rssi : 4;
} LIGHTING5;
```

packetlength:
Packet length (this byte not included) = 0x0A

packettype:
0x14 = lighting5

subtype:
0x00 = LightwaveRF, Siemens
0x01 = EMW100 GAO/Everflourish
0x02 = BBSB new types
0x03 = MDREMOTE LED dimmer
0x04 = Conrad RSL2
0x05 = Livolo
0x06 = RGB TRC02

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1 – id3:
remote/switch/unit ID. (id1 is high byte)
For subtype 0x00, 0x06: The unit ID can be 0x00 00 01 to 0xFF FF FF
For subtype 0x01: The unit ID can be 0x00 00 01 to 0x00 3F FF
For subtype 0x02: The unit ID can be 0x00 00 01 to 0x07 FF FF
For subtype 0x03, 0x05: The unit ID can be 0x00 00 01 to 0x00 FF FF
For subtype 0x04: The unit ID can be 0x00 00 01 to 0xFF FF FF

unitcode:
For subtype 0x00: 0x01 to 0x10 = unit 1 to 16
For subtype 0x01: 0x01 to 0x04 = unit 1 to 4
For subtype 0x02: 0x01 to 0x06 = unit 1 to 6
For subtype 0x03, 0x05, 0x06: not used
For subtype 0x04: 0x01 to 0x10 = unit 1 to 16

cmnd:

	LightwaveRF	EMW100	BBSB	RSL
Off	0x00	0x00	0x00	0x00
On	0x01	0x01	0x01	0x01
Group Off	0x02		0x02	0x02
Learn		0x02		
Group On			0x03	0x03
mood1	0x03			
mood2	0x04			
mood3	0x05			
mood4	0x06			
mood5	0x07			
reserved	0x08			
reserved	0x09			
unlock	0x0A			
lock	0x0B			
all lock	0x0C			
close (inline relay)	0x0D			
stop (inline relay)	0x0E			
open (inline relay)	0x0F			
set level	0x10			
Colour Palette	0x11			
Colour Tone	0x12			
Colour Cycle	0x13			

	MDREMOTE
Power	0x00
Light	0x01
Bright	0x02
Dim	0x03
100%	0x04
50%	0x05
25%	0x06
Mode+	0x07
Speed-	0x08
Speed+	0x09
Mode-	0x0A

	TRC02
Off	0x00
On	0x01
Bright	0x02
Dim	0x03
Color+	0x04
Color-	0x05
Select color	0x06 - 0x84

	Livolo
Group Off	0x00
On/Off dimmer or Gang1	0x01
Dim+ or Gang2 on/off	0x02
Dim- or Gang3 on/off	0x03

Livolo Dimmer:

This module accepts:

Group Off (is an OFF command)
Toggle On/Off
Dim+ (is a Bright command)
Dim- (is a dim command)

To switch this module ON you can use 2 methods. Send a Group Off followed by a Toggle command OR 6 or a few more DIM+ commands.

The dimmer has 7 states: Off, 17%, 33%, 50%, 67%, 83%, 100%

Livolo Appliance module (1 Gang):

This module accepts:

Group Off (is an OFF command)
Toggle On/Off

To be sure this module is switched ON send a Group Off followed by a Toggle command.

Livolo Appliance module (3 Gang):
It is not advised to use these modules because control is not easy to do.
This module accepts:

- Group Off (is an OFF command for all 3 gang!!)
- Toggle On/Off Gang 1
- Toggle On/Off Gang 2
- Toggle On/Off Gang 3

The problem with this type of module is that you don't know the state (as with all Livolo modules) but to get it in a known state you can only switch off all 3 gang and after that execute a toggle for the gang you want to switch ON. It is not possible to switch OFF only one gang.

level:
0x0 to 0x1F = 0% to 100% level (only used for LightwaveRF)
For LWRF Colour commands:
0x00 or 0x01 command toggle to indicate new command

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

10.11. 0x15, Lighting6 (AE)

10.11.1. 0x00: Blyss

Used by: transmitter and receiver

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   groupcode;
        BYTE   unitcode;
        BYTE   cmnd;
        BYTE   cmndseqnbr;
        BYTE   seqnbr2;
        BYTE   filler  : 4;
        BYTE   rssi : 4;
} LIGHTING6;
```

packetlength:
Packet length (this byte not included) = 0x0B

packettype:
0x15 = lighting6

subtype:
0x00 = Blyss

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1 – id2:
remote/switch/unit ID. (id1 is high byte)

groupcode:
0x41 to 0x50 = group A to P

unitcode:
0x01 to 0x05 = unit 1 to 5

cmnd:
0x00 = On
0x01 = Off
0x02 = group On
0x03 = group Off

cmndseqnbr:
0x00 to 0x04
To be incremented after each command.

seqnbr2:
The seqnbr2 is created by the RFXtrx433 if seqnbr2 = 0x00.

0x01 to 0x91 (1 to 145 decimal)
To be incremented after each command.

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

Example Blyss commands

```
----------------------------------------------
0B150005D950450101011D80
Packettype    = Lighting6
subtype       = Blyss
Sequence nbr  = 5
ID            = D950
groupcode     = E
unitcode      = 1
Command       = Off
Command seqnbr= 1
seqnbr2       = 29
Signal level  = 8
----------------------------------------------
0B150006D950450100021E80
Packettype    = Lighting6
subtype       = Blyss
Sequence nbr  = 6
ID            = D950
groupcode     = E
unitcode      = 1
Command       = On
Command seqnbr= 2
seqnbr2       = 30
Signal level  = 8
```

It seems the cmndseqnbr and seqnbr2 can be used system wide.
So use one variable for cmndseqnbr and one for seqnbr2 and use these in any Blyss command.

Note that if a Blyss device is controlled by a Blyss remote that the Blyss device will not respond immediately but it needs to get a few commands. After this the Blyss device will not immediately respond to an RFXtrx433 command because the sequence numbers are out of sync.
You can however increment the cmndseqnbr and seqnbr2 on received Blyss remote commands so that the RFXtrx433 is kept in sync.

10.12. 0x16, Chime

10.12.1. 0x00: Byron SX

Used by: transmitter and receiver

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   sound;
        BYTE   filler  : 4;
        BYTE   rssi : 4;
} CHIME;
```

packetlength:
Packet length (this byte not included) = 0x07

packettype:
0x16 = chime

subtype:
0x00 = Byron SX

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1-id2:
chime ID. (id1 is high byte)
For SX21 The unit ID can be 0x00 to 0xFF

sound:
For SX21:
0x01, 0x0D = Tubular 3 notes
0x03, 0x0E = Big Ben
0x05, 0x06 = Tubular 2 notes
0x09, 0x02 = Solo

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

10.13. 0x17, Fan

10.13.1. 0x00: Siemens SF01

Used by: transmitter only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   id3;
        BYTE   cmnd;
        BYTE   filler  : 4;
        BYTE   rssi : 4;
} FAN;
```

packetlength:
Packet length (this byte not included) = 0x07

packettype:
0x17 = fan

subtype:
0x00 = Siemens SF01 - LF959RA50/LF259RB50/LF959RB50 extractor hood

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1-id2:
fan ID. (id1 is high byte)
For 0x00 The unit ID can be from 0x00 00 00 to 0x00 7F FF

cmnd:

	Siemens	
Timer	0x01	
-	0x02	
Learn	0x03	
+	0x04	
Confirm	0x05	
Light	0x06	
On		
Off		

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

10.14. 0x18, Curtain1

10.14.1. 0x00: Harrison

Used by: transmitter (special receive only for RollerTrol)

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   housecode;
        BYTE   unitcode;
        BYTE   cmnd;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} CURTAIN1;
```

packetlength:
Packet length (this byte not included) = 0x07

packettype:
0x18 = curtains

subtype:
0x00 = Harrison Curtain

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

housecode:
Harrison 0x41 to 0x50 = house code A to P

unitcode:
Harrison 0x01 to 0x10 = unit 1 to 16

cmnd:

	Harrison
Open	0x00
Close	0x01
Stop	0x02
Program	0x03

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

10.15. 0x19, Blinds1

10.15.1. 0x00-0x06: RollerTrol, Hasta, A-OK, Raex, Media Mount, DC,Forest

Used by: transmitter and receiver (BlindsT0 only with dedicated receive)

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   id3;
        BYTE   unitcode;
        BYTE   cmnd;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} BLINDS1;
```

packetlength:
Packet length (this byte not included) = 0x09

packettype:
0x19 = shades

subtype:
0x00 = BlindsT0 = RollerTrol, Hasta new
0x01 = BlindsT1 = Hasta old
0x02 = BlindsT2 = A-OK RF01
0x03 = BlindsT3 = A-OK AC114
0x04 = BlindsT4 = Raex YR1326
0x05 = BlindsT5 = Media Mount
0x06 = BlindsT6 = DC/RMF/Yooda
0x07 = BlindsT7 = Forest

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1 – id3:
unit ID. (id1 is high byte)
For subtype 0x00 and 0x01:
    The unit ID can be 0x00 00 01 to 0x00 FF FF
For subtype 0x02, 0x03, 0x04, 0x05, 0x06 and 0x07:
    The unit ID can be 0x00 00 01 to 0xFF FF FF

unitcode:
For subtype 0x00 and 0x01:
    0x01 to 0x0F = unit 1 to 15
    0x10 = all units
For subtype 0x02, 0x03, 0x04 and 0x05:
    unitcode = 0
For subtype 0x06, 0x07:
    0x01 to 0x0F = unit 1 to 15
    0x00 = all units

cmnd:

subtype	**0x00**	**0x01**	**0x02**	**0x03 0x06 0x07**	**0x04**	**0x05**
Open	0x00	0x00	0x00	0x00	0x00	0x00 (down)
Close	0x01	0x01	0x01	0x01	0x01	0x01 (up)
Stop	0x02	0x02	0x02	0x02	0x02	0x02
Confirm/Pair	0x03	0x03	0x03	0x03	0x03	
Set Limit	0x04	0x04			0x04 (upper)	
Set Lower Limit					0x05	
Delete limits					0x06	
Change direction					0x07	
Left					0x08 (not yet used)	
Right					0x09 (not yet used)	

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

Note: All other receive protocols are disabled if BlindsT0 receive is enabled

Example Raex YR1326 remote to control the T16 motor

```
----------------------------------------------
0919040600A21B010280
Packettype    = BLINDS1
subtype       = RAEX
Sequence nbr  = 6
id1-3         = 00A21B
Unit          = 1
Command       = Stop
Signal level  = 8
```

Example Media Mount

```
==============================================
Packettype    = BLINDS1
subtype       = Media Mount
Sequence nbr  = 2
id1-3         = 1A6280
Unit          = 1
Command       = Open
Signal level  = 0
----------------------------------------------
Blinds command:09 19 05 02 1A 62 80 01 00 00
```

Notes:
Media Mount is received as Lighting4 command!!!
Use address 1A 62 80 because the address is fixed.

10.16. 0x20, Security1

10.16.1. 0x00-0x08: X10, KD101, Visonic, Meiantech, SA30

Used by: receiver and transmitter

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   id3;
        BYTE   status;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} SECURITY1;
```

packetlength:
Packet length (this byte not included) = 0x08

packettype:
0x20 = security

subtype:
0x00 = X10 security door/window sensor
0x01 = X10 security motion sensor
0x02 = X10 security remote (no alive packets)
0x03 = KD101 (no alive packets)
0x04 = Visonic PowerCode door/window sensor – primary contact (with alive packets)
0x05 = Visonic PowerCode motion sensor (with alive packets)
0x06 = Visonic CodeSecure (no alive packets)
0x07 = Visonic PowerCode door/window sensor – auxiliary contact (no alive packets)
0x08 = Meiantech
0x09 = SA30 (no alive packets)

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1 – id3:
remote/sensor ID.
For X10 id2 is forced in the RFXtrx to comply with the X10 standard. id2 is therefore not used.
For KD101 id3 is forced in the RFXtrx to comply with the KD101 standard. id3 is therefore not used.
For SA30 id3 is not used.

status:

	subtype to be used in transmit for:						
	X10 security door-window sensor	X10 security motion sensor	X10 security remote	PowerCode door- window sensor	PowerCode motion sensor	Meian tech/ Atlantic/ Aidebao	KD101/ SA30
0x00 = normal	x			x			
0x01 = normal delayed	x						
0x02 = alarm	x			x			
0x03 = alarm delayed	x						
0x04 = motion		x			x		
0x05 = no motion		x			x		
0x06 = panic			x			x	x
0x07 = end panic			x				
0x08 = IR						x	
0x09 = arm away			x			x (gr2)	
0x0A = arm away delayed			x				
0x0B = arm home			x			x (gr3)	
0x0C = arm home delayed			x				
0x0D = disarm			x			x (gr1)	
0x10 = light 1 off			x				
0x11 = light 1 on			x				
0x12 = light 2 off			x				
0x13 = light 2 on			x				
0x14 = dark detected							
0x15 = light detected							
0x16 = batlow SD18, CO18							
0x17 = pair KD101/SA30							x
0x80 = normal + tamper	x			x			
0x81 = normal delayed + tamper	x						
0x82 = alarm + tamper	x			x			
0x83 = alarm delayed + tamper	x						
0x84 = motion + tamper		x			x		
0x85 = no motion + tamper		x			x		

To get the status without tamper: AND the status with 0x7F
Meiantech group commands are equal to security commands:disarm=group1, arm away=group2 and arm home = group3

battery_level: (is 0x0 for transmitter command)
0x9 = full
0x0 = empty
Note: KD101 and SA30 do not support battery level!

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

Notes:
X10 security sensors transmit about every 80 minutes an alive heart-beat packet with the current status of the sensor.
The KD101 and SA30 do not transmit an alive heart-beat.
Motion sensors and the primary contact of the Visonic PowerCode door/window sensors transmit about every 15 minutes an alive / heart-beat packet with the current status of the sensor.
The auxiliary contacts of the door/window sensors do NOT send an alive / heart-beat packet.

Example X10 security door/window sensor

```
------------------------------------------------
0820004DD3DC540089
Packettype    = Security1
subtype       = X10 security
Sequence nbr  = 77
id1-3         = D3DC54
status        = Normal
battery level = OK
Signal level  = 8
```

10.17. 0x28, Camera1

10.17.1. 0x00: X10 Ninja/Robocam

Used by: receiver and transmitter

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   housecode;
        BYTE   cmnd;
        BYTE   filler : 4;
        BYTE   rssi : 4;
} CAMERA1;
```

packetlength:
Packet length (this byte not included) = 0x06

packettype:
0x28 = camera

subtype:
0x00 = X10 Ninja

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

housecode:
0x41 to 0x50 = house code A to P

cmnd:
0x0 = Left
0x1 = Right
0x2 = Up
0x3 = Down
0x4 = Position 1
0x5 = Program Position 1
0x6 = Position 2
0x7 = Program Position 2
0x8 = Position 3
0x9 = Program Position 3
0xA = Position 4
0xB = Program Position 4
0xC = Center
0xD = Program Center Position
0xE = Sweep
0xF = Program Sweep

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

Note: the Ninja camera has to be selected using an X10 ON command before a command is send to the camera.

10.18. 0x30, Remote control and IR

10.18.1. 0x00-0x04: ATI, Medion, PC Remote

Used by: transmitter (not ATI RW2) and receiver

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id;
        BYTE   cmnd;
        BYTE   toggle : 1;
        BYTE   cmndtype : 3;
        BYTE   rssi : 4;
} REMOTE;
```

packetlength:
Packet length (this byte not included) = 0x06

packettype:
0x30 = remote control and IR commands

subtype:
0x00 = ATI Remote Wonder
0x01 = ATI Remote Wonder Plus
0x02 = Medion Remote
0x03 = X10 PC Remote
0x04 = ATI Remote Wonder II (receive only)

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id:
remote ID.

cmnd:
See the appendix 13.1 for the cmnd description.

toggle:
Only used for ATI Remote Wonder Plus and ATI Remote Wonder II. This bit toggles for every button press.

cmndtype:
Only used by ATI Remote Wonder II.
0 = PC
1 = AUX1
2 = AUX2
3 = AUX3
4 = AUX4

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

Example ATI Remote Wonder button pressed = 1

```
----------------------------------------------
063000040F0D82
Packettype    = Remote control & IR
subtype       = ATI Remote Wonder
Sequence nbr  = 4
ID            = 15
Command       = 1
Signal level  = 8
```

Example Medion Remote button pressed = 1

```
----------------------------------------------
0630000E000D80
Packettype    = Remote control & IR
subtype       = ATI Remote Wonder
Sequence nbr  = 14
ID            = 0
Command       = 1
Signal level  = 8
```

Example ATI Remote Wonder Plus button pressed = 1 (twice)

```
----------------------------------------------
063001060F0D72
Packettype    = Remote control & IR
subtype       = ATI Remote Wonder Plus
Sequence nbr  = 6
ID            = 15
Command       = 1  (button press = even)
Signal level  = 7
----------------------------------------------
063001070F0D73
Packettype    = Remote control & IR
subtype       = ATI Remote Wonder Plus
Sequence nbr  = 7
ID            = 15
Command       = 1  (button press = odd)
Signal level  = 7
```

Example ATI Remote Wonder II button pressed = 1 (twice)

```
----------------------------------------------
0630040B000D81
Packettype    = Remote control & IR
subtype       = ATI Remote Wonder II
Sequence nbr  = 11
ID            = 0
Command type  = PC
Command       = 1  (button press = odd)
Signal level  = 8
----------------------------------------------
0630040C000D80
Packettype    = Remote control & IR
subtype       = ATI Remote Wonder II
Sequence nbr  = 12
ID            = 0
Command type  = PC
Command       = 1  (button press = even)
Signal level  = 8
```

10.19. 0x40, Thermostat1

10.19.1. 0x00-0x01: Digimax

Used by: receiver and transmitter

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   temperature;
        BYTE   set_point;
        BYTE   status : 2;
        BYTE   filler : 5;
        BYTE   mode : 1;
        BYTE   filler1 : 4;
        BYTE   rssi : 4;
} THERMOSTAT1;
```

packetlength:
Packet length (this byte not included) = 0x09

packettype:
0x40 = Thermostat1

subtype:
0x00 = Digimax, TLX7506
0x01 = Digimax with short format (no set point)

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1-2:
Sensor ID.

temperature:
Current temperature

set_point:
Requested temperature

mode:
0x0 = heating
0x1 = cooling

status:
0x0 = no status available
0x1 = demand
0x2 = no demand
0x3 = initializing

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
The Digimax transmits about every 5 minutes an alive heart-beat packet with the current status of the sensor and at will send the packet on a status change.

Example Digimax, TLX7506

```
----------------------------------------------
0940001B6B1816150270
Packettype    = Thermostat1
subtype       = Digimax/TLX7506
Sequence nbr  = 27
ID            = 27416 ID1=6B ID2=18
Temperature   = 22 °C
Set           = 21 °C
Mode          = heating
Status        = no demand
Signal level  = 7
```

10.20. 0x41, Thermostat2

10.20.1. 0x00-0x01: HomeEasy HE105, RTS10

Used by: transmitter, receive not implemented.

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   unitcode;
        BYTE   cmnd;
        BYTE   filler : 4;
        BYTE   rssi : 4;
} THERMOSTAT2;
```

packetlength:
Packet length (this byte not included) = 0x06

packettype:
0x41 = thermostat2

subtype:
0x00 = HE105
0x01 = RTS10, RFS10, TLX1206

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

unitcode:

code	HE105 switches 1 2 3 4 5	code	HE105 switches 1 2 3 4 5
0x00	0 0 0 0 0	0x10	1 0 0 0 0
0x01	0 0 0 0 1	0x11	1 0 0 0 1
0x02	0 0 0 1 0	0x12	1 0 0 1 0
0x03	0 0 0 1 1	0x13	1 0 0 1 1
0x04	0 0 1 0 0	0x14	1 0 1 0 0
0x05	0 0 1 0 1	0x15	1 0 1 0 1
0x06	0 0 1 1 0	0x16	1 0 1 1 0
0x07	0 0 1 1 1	0x17	1 0 1 1 1
0x08	0 1 0 0 0	0x18	1 1 0 0 0
0x09	0 1 0 0 1	0x19	1 1 0 0 1
0x0A	0 1 0 1 0	0x1A	1 1 0 1 0
0x0B	0 1 0 1 1	0x1B	1 1 0 1 1
0x0C	0 1 1 0 0	0x1C	1 1 1 0 0
0x0D	0 1 1 0 1	0x1D	1 1 1 0 1
0x0E	0 1 1 1 0	0x1E	1 1 1 1 0
0x0F	0 1 1 1 1	0x1F	1 1 1 1 1

RTS10 unitcode = 0x00 – 0xFF

cmnd:
0x00 = Off
0x01 = On
0x02 = program RTS10

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

10.21. 0x42, Thermostat3

10.21.1. 0x00-0x01: Mertik-Maxitrol G6R-H4 type

Used by: transmitter and receiver

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   unitcode1;
        BYTE   unitcode2;
        BYTE   unitcode3;
        BYTE   cmnd;
        BYTE   filler : 4;
        BYTE   rssi : 4;
} THERMOSTAT3;
```

packetlength:
Packet length (this byte not included) = 0x08

packettype:
0x42 = thermostat3

subtype:
0x00 = Mertik G6R-H4T1
0x01 = Mertik G6R-H4TB / G6-H4T / G6R-H4T21-Z22

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

unitcode1 to unitcode3:
G6R-H4T1 unitcode1 & 2 = 0, unitcode3 = 0x00 to 0xFF
G6R-H4TB unitcode = 0x0 to 0x3FFFF

cmnd:

	G6R-H4T1	G6R-H4TB	G6R-H4TB
0x0	Off	Off	Off
0x1	On	On	On
0x2	Up	Up	Up
0x3	Down	Down	Down
0x4	Run Up	2nd Off	
0x5	Run Down	2nd On	
0x6	Stop		

rssi: (is 0x0 for transmitter command)
Signal strength.
0x0 to 0xF = weak to strong

Example G6R-H4TB / G6R-H4T

```
----------------------------------------------
08420101019FAB0281
Packettype    = Thermostat3
subtype       = Mertik G6R-H4TB
Sequence nbr  = 1
ID            = 0x019FAB
Command       = Up
Signal level  = 8
```

10.22. 0x4E, BBQ Temperature sensors

10.22.1. 0x01: BBQ1

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   sensor1high;
        BYTE   sensor1low;
        BYTE   sensor2high;
        BYTE   sensor2low;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} BBQ;
```

packetlength:
Packet length (this byte not included) = 0x0A

packettype:
0x4E = BBQ temperature sensors

subtype:
0x01 = BBQ1 is Maverick ET-732

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1-2:
Sensor ID.

sensor1high - sensor1low:
temperature of sensor 1 (Food)

sensor2high – sensor2low:
temperature of sensor 2 (BBQ)

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Example BBQ1 is ET732:

```
----------------------------------------------
0A4E010000000019001789
Packettype    = BBQ
subtype       = BBQ1 - Maverick ET-732
Sequence nbr  = 0
ID            = 0
Sensor1 temp  = 25 °C
Sensor2 temp  = 23 °C
Signal level  = 8
Battery       = OK
```

10.23. 0x4F: Temperature and rain sensors

10.23.1. 0x01: TR1

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   temperaturehigh : 7;
        BYTE   temperaturesign : 1;
        BYTE   temperaturelow;
        BYTE   raintotal1;        //high byte
        BYTE   raintotal2;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} TEMP_RAIN;
```

packetlength:
Packet length (this byte not included) = 0x0A

packettype:
0x4F = temperature+rain sensors

subtype:
0x01 = TR1 is Alecto WS1200

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
Sensor ID.

temperaturehigh - temperaturelow:
7 bits high byte and 8 bits low byte = temperature * 10

temperaturesign:
0 = positive temperature
1 = negative temperature

raintotal1-2:
Total rain fall in mm * 10

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

10.24. 0x50, Temperature sensors

10.24.1. 0x01-0x0A: TEMP1-TEMP10

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   temperaturehigh : 7;
        BYTE   temperaturesign : 1;
        BYTE   temperaturelow;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} TEMP;
```

packetlength:
Packet length (this byte not included) = 0x08

packettype:
0x50 = temperature sensors

subtype:
0x01 = TEMP1 is THR128/138, THC138
0x02 = TEMP2 is THC238/268,THN132,THWR288,THRN122,THN122,AW129/131
0x03 = TEMP3 is THWR800
0x04 = TEMP4 is RTHN318
0x05 = TEMP5 is La Crosse TX2, TX3, TX4, TX17
0x06 = TEMP6 is TS15C. UPM temp only
0x07 = TEMP7 is Viking 02811
0x08 = TEMP8 is La Crosse WS2300
0x09 = TEMP9 is Rubicson
0x0A = TEMP10 is TFA 30.3133

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1-2:
Sensor ID.

temperaturehigh - temperaturelow:
7 bits high byte and 8 bits low byte = temperature * 10

temperaturesign:
0 = positive temperature
1 = negative temperature

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
Oregon Scientific sensors transmit about every 1 to 2 minutes the current status of the sensor.
La Crosse sensors transmit about every minute the current status of the sensor.

The selected channel of the Oregon sensor is in id2.

Example TEMP1 is THR128/138, THC138:

```
----------------------------------------------
08500110000180BC69
Packettype    = TEMP
subtype       = TEMP1 - THR128/138, THC138
                channel 1
Sequence nbr  = 16
ID            = 1
Temperature   = -18,8 °C
Signal level  = 6
Battery       = OK
```

Example TEMP2 is THC238/268,THN132,THWR288,THRN122,THN122,AW129/131

```
----------------------------------------------
0850021DFB0100D770
Packettype    = TEMP
subtype       = TEMP2 - THC238/268,THN132,THWR288,THRN122,THN122,AW129/131
                channel 1
Sequence nbr  = 29
ID            = 64257
Temperature   = 21,5 °C
Signal level  = 7
Battery       = Low
```

TEMP5 is La Crosse TX2, TX3, TX4, TX17

```
----------------------------------------------
08500502770000D389
Packettype    = TEMP
subtype       = TEMP5 - LaCrosse TX2, TX3, TX4, TX17
Sequence nbr  = 2
ID            = 30464
Temperature   = 21,1 °C
Signal level  = 8
Battery       = OK
```

TEMP9 is Rubicson

```
----------------------------------------------
0850091A00C3800689
Packettype    = TEMP
subtype       = TEMP9 - RUBiCSON 48695
Sequence nbr  = 26
ID            = 195
Temperature   = -0,6 °C
Signal level  = 8
Battery       = OK
----------------------------------------------
0850097200C300E089
Packettype    = TEMP
subtype       = TEMP9 - RUBiCSON 48695
Sequence nbr  = 114
ID            = 195
Temperature   = 22,4 °C
Signal level  = 8
Battery       = OK
```

10.25. 0x51: Humidity sensors

10.25.1. 0x01: TX3

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   humidity;
        BYTE   humidity_status;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} HUM;
```

packetlength:
Packet length (this byte not included) = 0x08

packettype:
0x51 = humidity sensors

subtype:
0x01 = HUM1 is LaCrosse TX3
0x02 = HUM2 is LaCrosse WS2300

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
Sensor ID.

humidity:
relative humidity 0 – 100% RH

humidity_status:
0x00 = normal
0x01 = comfort
0x02 = dry
0x03 = wet

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
LaCrosse sensors transmit about every minute the current status of the sensor.

Example HUM1 is LaCrosse TX3

```
------------------------------------------------
085101027700360189
Packettype    = HUM
subtype       = HUM1 - LaCrosse TX3
Sequence nbr  = 2
ID            = 30464
Humidity      = 54
Status        = Comfortable
Signal level  = 8
Battery       = OK
```

10.26. 0x52: Temperature and humidity sensors

10.26.1. 0x01-0x09: TH1-TH11

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   temperaturehigh : 7;
        BYTE   temperaturesign : 1;
        BYTE   temperaturelow;
        BYTE   humidity;
        BYTE   humidity_status;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} TEMP_HUM;
```

packetlength:
Packet length (this byte not included) = 0x0A

packettype:
0x52 = temperature+humidity sensors

subtype:
0x01 = TH1 is THGN122/123, THGN132, THGR122/228/238/268
0x02 = TH2 is THGR810, THGN800
0x03 = TH3 is RTGR328
0x04 = TH4 is THGR328
0x05 = TH5 is WTGR800
0x06 = TH6 is THGR918/928, THGRN228, THGN500
0x07 = TH7 is TFA TS34C, Cresta
0x08 = TH8 is WT260,WT260H,WT440H,WT450,WT450H
0x09 = TH9 is Viking 02035,02038 (02035 has no humidity)
0x0A = TH10 is Rubicson
0x0B = TH11 is EW109

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
Sensor ID.

temperaturehigh - temperaturelow:
7 bits high byte and 8 bits low byte = temperature * 10

temperaturesign:
0 = positive temperature
1 = negative temperature

humidity:
relative humidity 0 – 100% RH

humidity_status:
0x00 = normal
0x01 = comfort
0x02 = dry
0x03 = wet

battery_level:
0x9 = full
0x0 = empty

Note that the THGR918/928 reports a battery level:
0x9 = 100% full
0x8 = 90%
0x7 = 80%
0x6 = 70%
0x5 = 60%
0x4 = 50%
0x3 = 40%
0x2 = 30%
0x1 = 20%
0x0 = 10% empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
Oregon Scientific sensors transmit about every 1 to 2 minutes the current status of the sensor.

The selected channel of the Oregon and Esic sensor is in id2.

The selected channel of a TH7 sensor is in id1:
Value 0x20 to 0x3F = channel 1
Value 0x40 to 0x5F = channel 2
Value 0x60 to 0x7F = channel 3
Value 0xA0 to 0xBF = channel 4
Value 0xC0 to 0xDF = channel 5

Example TH2 is THGR810, THGN800, THGR810

```
----------------------------------------------
0A520211700200A72D0089
Packettype    = TEMP_HUM
subtype       = TH2 - THGN800,TGHN801,THGR810
                channel 2
Sequence nbr  = 17
ID            = 28674
Temperature   = 16,7 °C
Humidity      = 45
Status        = Normal
Signal level  = 8
Battery       = OK
```

Example TH5 is WTGR800

```
----------------------------------------------
0A5205D42F000082590379
Packettype    = TEMP_HUM
subtype       = TH5 - WTGR800
Sequence nbr  = 212
ID            = 12032
Temperature   = 13 °C
Humidity      = 89
Status        = Wet
Signal level  = 7
Battery       = OK
```

10.27. 0x53: Barometric sensors

10.27.1. 0x01: reserved for future use

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   baro1;
        BYTE   baro2;
        BYTE   forecast;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} BARO;
```

packetlength:
Packet length (this byte not included) = 0x09

packettype:
0x53 = barometric sensors

subtype:
0x01 =

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
Sensor ID.

baro:
Barometric value in hPa.

forecast:
0x00 = no forecast available
0x01 = sunny,
0x02 = partly cloudy,
0x03 = cloudy,
0x04 = rain.

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
Oregon Scientific sensors transmit about every 1 to 2 minutes the current status of the sensor.

10.28. 0x54: Temperature, humidity and barometric sensors

10.28.1. 0x01-0x02: THB1-THB2

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   temperaturehigh : 7;
        BYTE   temperaturesign : 1;
        BYTE   temperaturelow;
        BYTE   humidity;
        BYTE   humidity_status;
        BYTE   baro1;
        BYTE   baro2;
        BYTE   forecast;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} TEMP_HUM_BARO;
```

packetlength:
Packet length (this byte not included) = 0x0D

packettype:
0x54 = temperature+humidity+barometric sensors

subtype:
0x01 = THB1 is BTHR918
0x02 = THB2 is BTHR918N, BTHR968

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
Sensor ID.

temperaturehigh - temperaturelow:
7 bits high byte and 8 bits low byte = temperature * 10

temperaturesign:
0 = positive temperature
1 = negative temperature

humidity:
relative humidity 0 – 100% RH

humidity_status:
0x00 = normal
0x01 = comfort
0x02 = dry
0x03 = wet

baro1-2:
Barometric value in hPa.
baro1 is the high and baro2 the lower part of the value

forecast:
0x00 = no forecast available
0x01 = sunny,
0x02 = partly cloudy,
0x03 = cloudy,
0x04 = rain.

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
Oregon Scientific sensors transmit about every 1 to 2 minutes the current status of the sensor**.**

Example BTHR918N/BTHR968

```
----------------------------------------------
0D54020EE90000C9270203E70439
Packettype    = TEMP_HUM_BARO
subtype       = THB2 - BTHR918N, BTHR968
                channel 0
Sequence nbr  = 14
ID            = 59648
Temperature   = 20,1 °C
Humidity      = 39
Status        = Dry
Barometer     = 999 hPa
Forecast      = Rain
Signal level  = 3
Battery       = OK
```

10.29. 0x55: Rain sensors

10.29.1. 0x01-0x05: RAIN1-RAIN6

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   rainratehigh;
        BYTE   rainratelow;
        BYTE   raintotal1;      //high byte
        BYTE   raintotal2;
        BYTE   raintotal3;      //low byte
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} RAIN;
```

packetlength:
Packet length (this byte not included) = 0x0B

packettype:
0x55 = rain sensors

subtype:
0x01 = RAIN1 is RGR126/682/918/928
0x02 = RAIN2 is PCR800
0x03 = RAIN3 is TFA
0x04 = RAIN4 is UPM RG700
0x05 = RAIN5 is WS2300
0x06 = RAIN6 is La Crosse TX5

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1-2:
Sensor ID.

rainrate: (only used by RAIN1 and RAIN2)
Rain fall rate:
for RAIN1 in mm/hr
for RAIN2 in mm * 100 / hr

raintotal1-3:
For RAIN1 to RAIN5: Total rain fall in mm * 10
For RAIN6: raintotal3 is a flip counter 0 to 15. Each flip is 0.266mm

battery_level:
0x9 = full
0x0 = empty

RAIN1:
0x9 = 100% full
0x8 = 90%
0x7 = 80%
0x6 = 70%
0x5 = 60%
0x4 = 50%
0x3 = 40%

0x2 = 30%
0x1 = 20%
0x0 = 10% empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
Oregon Scientific sensors transmit about every 1 to 2 minutes the current status of the sensor.

The RAIN6 La Crosse TX5 transmits only a flip count. The totalrain3 field contains the flipcounter. The total rain must be calculated in the application for this sensor.

Example RAIN2 is PCR800

```
----------------------------------------------
0B550217B6000000004D3C69
Packettype    = RAIN
subtype       = RAIN2 - PCR800
Sequence nbr  = 23
ID            = 46592
Rain rate     = 0 mm/h
Total rain    = 1977,2 mm
Signal level  = 6
Battery       = OK
```

10.30. 0x56: Wind sensors

10.30.1. 0x01-0x06: WIND1-WIND6

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   directionhigh;
        BYTE   directionlow;
        BYTE   av_speedhigh;
        BYTE   av_speedlow;
        BYTE   gusthigh;
        BYTE   gustlow;
        BYTE   temperaturehigh : 7;
        BYTE   temperaturesign : 1;
        BYTE   temperaturelow;
        BYTE   chillhigh : 7;
        BYTE   chillsign : 1;
        BYTE   chilllow;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
}WIND;
```

packetlength:
Packet length (this byte not included) = 0x10

packettype:
0x56 = wind sensors

subtype:
0x01 = WIND1 is WTGR800
0x02 = WIND2 is WGR800
0x03 = WIND3 is STR918, WGR918, WGR928
0x04 = WIND4 is TFA
0x05 = WIND5 is UPM WDS500
0x06 = WIND6 is WS2300

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
Sensor ID.

direction:
wind direction in degrees

av_speed: (not used in WIND5)
average wind speed in meters per second * 10

gust:
wind speed in meters per second * 10

temperaturehigh - temperaturelow: (only used in WIND4)
7 bits high byte and 8 bits low byte = temperature * 10

temperaturesign: (only used in WIND4)
0 = positive temperature
1 = negative temperature

chillhigh - chilllow: (only used in WIND4)
7 bits high byte and 8 bits low byte = chill temperature * 10

chillsign: (only used in WIND4)
0 = positive chill temperature
1 = negative chill temperature

battery_level:
0x9 = full
0x0 = empty

WIND3:
0x9 = 100% full
0x8 = 90%
0x7 = 80%
0x6 = 70%
0x5 = 60%
0x4 = 50%
0x3 = 40%
0x2 = 30%
0x1 = 20%
0x0 = 10% empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
Oregon Scientific sensors transmit about every 1 to 2 minutes the current status of the sensor.

Example WIND1 is WTGR800

```
----------------------------------------------
105601122F000087000000140049000079
Packettype    = WIND
subtype       = WIND1 - WTGR800
Sequence nbr  = 18
ID            = 12032
Direction     = 135 degrees  SE
Average speed = 0 mtr/sec = 0 km/hr = 0 mph
Wind gust     = 2 mtr/sec = 7,2 km/hr = 4,47 mph
Signal level  = 7
Battery       = OK
```

10.31. 0x57: UV sensors

10.31.1. 0x01-0x03: UV1-UV3

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   uv;
        BYTE   temperaturehigh : 7;
        BYTE   temperaturesign : 1;
        BYTE   temperaturelow;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
}UV;
```

packetlength:
Packet length (this byte not included) = 0x09

packettype:
0x57 = UV sensors

subtype:
0x01 = UV1 is UVN128, UV138
0x02 = UV2 is UVN800
0x03 = UV3 is TFA

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1-2:
Sensor ID.

uv:
UV value * 10

temperaturehigh - temperaturelow: (only used in UV3)
7 bits high byte and 8 bits low byte = temperature * 10

temperaturesign: (only used in UV3)
0 = positive temperature
1 = negative temperature

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
Oregon Scientific sensors transmit about every 1 to 2 minutes the current status of the sensor.

10.32. 0x58: Date/time sensors

10.32.1. 0x01: DT1

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   yy;
        BYTE   mm;
        BYTE   dd;
        BYTE   dow;
        BYTE   hr;
        BYTE   min;
        BYTE   sec;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
}DT;
```

packetlength:
Packet length (this byte not included) = 0x0D

packettype:
0x58 = Date/Time sensors

subtype:
0x01 = DT1 is RTGR328N

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
Sensor ID.

dow:
day of week: 1 to 7.

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
Oregon Scientific sensors transmit about every 1 to 2 minutes the current status of the sensor.

10.33. 0x59: Current sensors

10.33.1. 0x01: ELEC1

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   count;
        BYTE   ch1_high;
        BYTE   ch1_low;
        BYTE   ch2_high;
        BYTE   ch2_low;
        BYTE   ch3_high;
        BYTE   ch3_low;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
}CURRENT;
```

packetlength:
Packet length (this byte not included) = 0x0D

packettype:
0x59 = Current sensors

subtype:
0x01 = ELEC1 is CM113, Electrisave, cent-a-meter

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
sensor ID.

count:
count field increments with each transmission.

ch1:
high = high byte, low = low byte. Ampere channel 1 * 10

ch2:
high = high byte, low = low byte. Ampere channel 2 * 10

ch3:
high = high byte, low = low byte. Ampere channel 3 * 10

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
This sensor transmits about every 1 to 2 minutes the current status of the sensor and on a value change.

Example CM113, Electrisave, cent-a-meter

```
------------------------------------------------
0D59010F860004001D0000000049
Packettype    = CURRENT
subtype       = ELEC1 - OWL CM113, Electrisave, cent-a-meter
Sequence nbr  = 15
ID            = 34304
Count         = 4
Channel 1     = 2,9 ampere
Channel 2     = 0 ampere
Channel 3     = 0 ampere
Signal level  = 4
Battery       = OK
```

10.34. 0x5A: Energy usage sensors

10.34.1. 0x01-0x02: ELEC2-ELEC3

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   count;
        BYTE   instant1;
        BYTE   instant2;
        BYTE   instant3;
        BYTE   instant4;
        BYTE   total1;
        BYTE   total2;
        BYTE   total3;
        BYTE   total4;
        BYTE   total5;
        BYTE   total6;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
}ENERGY;
```

packetlength:
Packet length (this byte not included) = 0x11

packettype:
0x5A = Energy usage sensors

subtype:
0x01 = ELEC2 is CM119/160
0x02 = ELEC3 is CM180

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
sensor ID.

count:
packet counter

instant:
instant1 = high byte, instant4 = low byte. Current usage in Watts

total:
total1 = high byte, total6 = low byte. Must be divided by 223.666 to get the total usage in Wh

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength. 0x0 to 0xF = weak to strong

Notes;
This sensor transmits about every 1 to 2 minutes the current status of the sensor and on value change.

Example CM119 / CM160:

```
-----------------------------------------------
115A01071A7300000003F600000000350B89
Packettype    = ENERGY
subtype       = ELEC2 - OWL CM119, CM160
Sequence nbr  = 7
ID            = 6771
Count         = 0
Instant usage = 1014 Watt
total usage   = 60,7 Wh
Signal level  = 8
Battery       = OK
```

10.35. 0x5B: Current + ENERGY sensors

10.35.1. 0x01: ELEC4

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   count;
        BYTE   ch1_high;
        BYTE   ch1_low;
        BYTE   ch2_high;
        BYTE   ch2_low;
        BYTE   ch3_high;
        BYTE   ch3_low;
        BYTE   total1;
        BYTE   total2;
        BYTE   total3;
        BYTE   total4;
        BYTE   total5;
        BYTE   total6;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
} CURRENT_ENERGY;
```

packetlength:
Packet length (this byte not included) = 0x13

packettype:
0x5B = Current with power sensors

subtype:
0x01 = ELEC4 is CM180i

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
sensor ID.

count:
count field increments with each transmission.

ch1:
high = high byte, low = low byte. Ampere channel 1 * 10

ch2:
high = high byte, low = low byte. Ampere channel 2 * 10

ch3:
high = high byte, low = low byte. Ampere channel 3 * 10

total:
This field is only valid if count is zero.
total1 = high byte, total6 = low byte. Must be divided by 223.666 to get the total usage in Wh

battery_level:
0x9 = full
0x0 = empty

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Note:
The OWL Intuition-e and OWL Intuition-lc kits make use of the CM180i.
The CM180i is a current sensor with 3 channels and a total power value.
Unlike the CM113 this sensor does not immediately transmit a packet if the current on a channel changes but reports at fixed intervals of 1 minute.
If the count field is not equal to zero the packet contains only the value of the current sensors and the total power value is zero.
If the count field is zero the packet contains the value of the current sensors and the total power used value.

OWL sensor overview:
CM113 is an ampere meter for up to 3 CT's (current transformers) 3 ampere values are reported. The value is reported on a change or else every minute. The Electrisave and cent-a-meter are identical to the CM113

CM119 and CM160 are identical. This sensor reports the instant and total power used. It reports the total of all (max 3) CT's measured. It will immediately report the value if a change in power is measured.
A fixed value is set for the line voltage. It is therefore not a real power meter but for a normal household good enough to have a power usage indication.

The CM180i is a mix of the CM113 and CM119.
It reports every minute the ampere values of each CT.
Every 4 minutes it will also report the total power usage as done by the CM119. But it does not report an instant power. Instant power can be calculated as 230 x current measured.
Unlike the CM113 and CM119 this sensor does not immediately transmit a packet if the current on a channel changes but reports at fixed intervals of 1 minute.

Count = 0: current and total power

```
------------------------------------------------
135B0106B8000000160000000000006F148889
Packettype    = CURRENT_ENERGY
subtype       = ELEC4 - OWL CM180i
Sequence nbr  = 6
ID            = 47104
Count         = 0
Channel 1     = 2,2 ampere
Channel 2     = 0 ampere
Channel 3     = 0 ampere
total usage   = 32547,4 Wh
Signal level  = 8
Battery       = OK
```

Count = 2: only current

```
------------------------------------------------
135B014FB80002001D0000000000000000000079
Packettype    = CURRENT_ENERGY
subtype       = ELEC4 - OWL CM180i
Sequence nbr  = 79
ID            = 47104
Count         = 2
Channel 1     = 2,9 ampere
Channel 2     = 0 ampere
Channel 3     = 0 ampere
Signal level  = 7
Battery       = OK
```

10.36. 0x5C: Power sensors

10.36.1. 0x01: ELEC5

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   voltage;
        BYTE   currentH;
        BYTE   currentL;
        BYTE   powerH;
        BYTE   powerL;
        BYTE   energyH;
        BYTE   energyL;
        BYTE   pf;
        BYTE   freq;
        BYTE   filler : 4;
        BYTE   rssi : 4;
} POWER;
```

packetlength:
Packet length (this byte not included) = 0x0F

packettype:
0x5C = Current with power sensors

subtype:
0x01 = ELEC5 is Revolt

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
sensor ID.

voltage:
Power Line voltage

currentH-currentL:
Ampere * 100

powerH-powerL:
Instant power * 10 (W)

energy-energyL:
total power * 100 (kWh)

pf:
power factor

freq:
power line frequency in Hz

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Example ELEC5 Revolt

```
----------------------------------------------
0F5C0103002DE4000000000003003280
Packettype    = POWER
subtype       = ELEC5 - Revolt
Sequence nbr  = 3
ID            = 45
Voltage       = 228 Volt
Current       = 0 Ampere
Instant power = 0 Watt
total usage   = 0,03 kWh
power factor  = 0
Frequency     = 50
Signal level  = 8
----------------------------------------------
0F5C0104002DE40002002F0003643280
Packettype    = POWER
subtype       = ELEC5 - Revolt
Sequence nbr  = 4
ID            = 45
Voltage       = 228 Volt
Current       = 0,02 Ampere
Instant power = 4,7 Watt
total usage   = 0,03 kWh
power factor  = 1
Frequency     = 50
Signal level  = 8
----------------------------------------------
0F5C0105002DE3001401BD0003643280
Packettype    = POWER
subtype       = ELEC5 - Revolt
Sequence nbr  = 5
ID            = 45
Voltage       = 227 Volt
Current       = 0,2 Ampere
Instant power = 44,5 Watt
total usage   = 0,03 kWh
power factor  = 1
Frequency     = 50
Signal level  = 8
----------------------------------------------
0F5C0106002DE30005005700034D3280
Packettype    = POWER
subtype       = ELEC5 - Revolt
Sequence nbr  = 6
ID            = 45
Voltage       = 227 Volt
Current       = 0,05 Ampere
Instant power = 8,7 Watt
total usage   = 0,03 kWh
power factor  = 0,77
Frequency     = 50
Signal level  = 8
```

10.37. 0x5D: Weighting scale

10.37.1. 0x01-0x02: WEIGHT1-WEIGHT2

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   weighthigh;
        BYTE   weightlow;
        BYTE   battery_level : 4;
        BYTE   rssi : 4;
}WEIGHT;
```

packetlength:
Packet length (this byte not included) = 0x08

packettype:
0x5D = weighting scale

subtype:
0x01 = WEIGHT1 – BWR101/102
0x02 = WEIGHT2 – GR101

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

Id1-2:
sensor ID.

weighthigh-weightlow:
(weight * 10) kg

battery_level:
0x9 = full
0x0 = empty
Note: not supported by Oregon scales and will always be 0.

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
This sensor transmits on a value change.

10.38. 0x5E: Gas usage sensors

10.38.1. 0x01: reserved

Used by: receiver

10.39. 0x5F: Water usage sensors

10.39.1. 0x01: reserved

Used by: receiver

10.40. 0x70: RFXsensor

10.40.1. 0x00-0x03: Temp, Voltage, A/D, message

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id;
        BYTE   msg1;
        BYTE   msg2;
        BYTE   filler : 4;
        BYTE   rssi : 4;
}RFXSENSOR;
```

packetlength:
Packet length (this byte not included) = 0x07

packettype:
0x70 = RFXSensor

subtype:
0x00 = RFXSensor temperatuur
0x01 = RFXSensor A/D
0x02 = RFXSensor voltage
0x03 = RFXSensor message

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id:
sensor ID.

msg1-msg2 for subtype 0x00
measured value in Celsius * 100.
Bit 7 in msg1 is a sign bit. For a negative temperature this bit is 1.

msg1-msg2 for subtype 0x01 and 0x02
measured value in mV

msg1-msg2 for subtype 0x03
0x0001 = sensor addresses incremented
0x0002 = battery low detected
0x0081 = no 1-wire device connected
0x0082 = 1-Wire ROM CRC error
0x0083 = 1-Wire device connected is not a DS18B20 or DS2438
0x0084 = no end of read signal received from 1-Wire device
0x0085 = 1-Wire scratchpad CRC error

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
This sensor transmits every 1 or 5 minutes on a value change and depending on the interval rate selected in the RFXSensor. And this sensor transmits about every 80 minutes an alive heart-beat packet with the current status of the sensor.

Example RFXSensor temperatuur

```
----------------------------------------------
077000E92802E170
Packettype    = RFXSensor
subtype       = Temperature
Sequence nbr  = 233
ID            = 40
msg           = 7,37 °C
Signal level  = 7
```

Example RFXSensor negative temperatuur

```
----------------------------------------------
0770000208809650
Packettype    = RFXSensor
subtype       = Temperature
Sequence nbr  = 2
ID            = 8
msg           = -1,5 °C
Signal level  = 5
```

Example RFXSensor voltage

```
----------------------------------------------
077002EA2801D870
Packettype    = RFXSensor
subtype       = Voltage
Sequence nbr  = 234
ID            = 40
msg           = 472 mV
Signal level  = 7
```

Example RFXSensor A/D

```
----------------------------------------------
077001EB28018170
Packettype    = RFXSensor
subtype       = A/D
Sequence nbr  = 235
ID            = 40
msg           = 385 mV
Signal level  = 7
```

10.41. 0x71: RFXMeter

10.41.1. 0x00-0x0F: counter, messages

Used by: receiver only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   id1;
        BYTE   id2;
        BYTE   count1;
        BYTE   count2;
        BYTE   count3;
        BYTE   count4;
        BYTE   filler : 4;
        BYTE   rssi : 4;
}RFXMETER;
```

packetlength:
Packet length (this byte not included) = 0x0A

packettype:
0x71 = RFXMeter

subtype:
0x00 normal data packet
0x01 new interval time set.
Count3:
0x01 30 seconds
0x02 1 minute
0x04 6 minutes (RFXPower = 5 minutes)
0x08 12 minutes (RFXPower = 10 minutes)
0x10 15 minutes
0x20 30 minutes
0x40 45 minutes
0x80 60 minutes
0x02 calibrate value in <count> in µsec. (count2, 2 high bits indicated the channel)
0x03 new address set
0x04 counter value reset mode within 5 seconds (count2, 2 high bits indicated the channel)
0x0B counter value reset executed
0x0C set interval mode within 5 seconds
0x0D calibration mode within 5 seconds (count2, 2 high bits indicated the channel)
0x0E set address mode within 5 seconds
0x0F identification packet
Count3 = firmware version
0x00 – 0x3F = RFXPower
0x40 – 0x7F = RFU
0x80 – 0xBF = RFU
0xC0 – 0xFF = RFXMeter
Count4 = interval time (see subtype 0x01)

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

id1-2:
sensor ID.

count:
counter value 0x00 to 0xFFFFFF.

rssi:
Signal strength.
0x0 to 0xF = weak to strong

Notes;
This sensor transmits on a fixed interval depending on the interval rate selected in the RFXMeter.

Example RFXMeter

```
----------------------------------------------
0A71003708F8008A646770
Packettype    = RFXMeter
subtype       = RFXMeter counter
Sequence nbr  = 55
ID            = 2296
Counter       = 9069671
if RFXPwr     = 9069,671 kWh
Signal level  = 7
```

10.42. 0x72: FS20

10.42.1. 0x00-0x01: FS20, FHT 8V, FHT80

Used by: transmitter and receiver

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   hc1;
        BYTE   hc2;
        BYTE   addr;
        BYTE   cmd1;
        BYTE   cmd2;
        BYTE   filler : 4;
        BYTE   rssi : 4;
}FS20;
```

packetlength:
Packet length (this byte not included) = 0x09

packettype:
0x72 = FS20

subtype:
0x00 FS20
0x01 FHT8V valve
0x02 FHT80 door/window sensor

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

hc1-2:
House code.

addr:
Device address.
High nibble is group address 0x0 to 0xE. 0xF is Master address
Low nibble = sub address 0x0 to 0xE. 0xF is Local Master address

Translate address codes.
To translate an address entered with the FS20 remote subtract 1 from the button code, this will give you a two bits code and can be translated to a hex value. For example:
The house code entered on the FS20 remote = 2 3 1 4 1 3 4 2,
Subtract 1 from each button code = 1 2 0 3 0 2 3 1
Translate in bits: 01 10 00 11 00 10 11 01
Translate in hex: 6 3 2 D

cmd1:

Command codes FS20.

Bit 7: if 0 it is a command, if 1 it is response from a receiver
Bit 6: bidirectional command
Bit 5: additional command byte present

Bit 4-0: command. See the command table below.

0x00	Off	cmd2 optional
0x01	dim level 1 = 6.25%	cmd2 optional
0x02	dim level 2 = 12.5%	cmd2 optional
0x03	dim level 3 = 18.75%	cmd2 optional
0x04	dim level 4 = 25%	cmd2 optional
0x05	dim level 5 = 31.25%	cmd2 optional
0x06	dim level 6 = 37.5%	cmd2 optional
0x07	dim level 7 = 43.75%	cmd2 optional
0x08	dim level 8 = 50%	cmd2 optional
0x09	dim level 9 = 56.25%	cmd2 optional
0x0A	dim level 10 = 62.5%	cmd2 optional
0x0B	dim level 11 = 68.75%	cmd2 optional
0x0C	dim level 12 = 75%	cmd2 optional
0x0D	dim level 13 = 81.25%	cmd2 optional
0x0E	dim level 14 = 87.5%	cmd2 optional
0x0F	dim level 15 = 93.75%	cmd2 optional
0x10	On (100%)	cmd2 optional
0x11	On (at last dim level set)	cmd2 optional
0x12	Toggle between Off and On (last dim level set)	cmd2 optional
0x13	Bright one step	
0x14	Dim one step	
0x15	Start dim cycle	
0x16	Program timer	cmd2 present
0x17	Request status from a bidirectional device	
0x18	Off for timer period	cmd2 present
0x19	On (100%) for timer period	cmd2 present
0x1A	On (at last dim level set) for timer period	cmd2 present
0x1B	Reset	

Command codes FHT 8V valve.

Bit 7: if 0 it is a new command, if 1 it is a repeated command
Bit 6: bidirectional command
Bit 5: 1 (additional command byte present)
Bit 4: enable battery empty beep

Bit 3-0: command. See the command table below.

0x0	Synchronize now (cmd2 is valve position 0x00-0xFF = 0 to 100%)
0x1	open valve
0x2	close valve
0x6	open valve at percentage level (cmd2 0x00-0xFF = 0 to 100%)
0x8	relative offset (cmd2 bit 7=direction, bit 5-0 offset value)
0xA	decalcification cycle (cmd2 is end position 0x00-0xFF = 0 to 100%)
0xC	synchronization active (cmd2 bit 7-1 is count down in seconds, bit 0=1)
0xE	test, drive valve and produce an audible signal
0xF	pair valve (cmd2 bit 7-1 is count down in seconds, bit 0=1)

Command codes FHT80 door/window sensor.

Bit 7: if 0 it is a new command, if 1 it is a repeated command

Bit 3-0: command. See the command table below.

0x1	sensor opened
0x2	sensor closed
0xC	synchronization active

cmd2:
Additional command byte for FS20.
For commands 0x00 to 0x12, 0x18, 0x19 and 0x1A this cmd2 byte gives the timer value in steps of 0.25 seconds. The maximum time is 255 * 0.25 = 63.75 seconds.

For command 0x16 this cmd2 byte is: 2 ^ high nibble * low nibble * 0.25 seconds.
The high nibble can have a maximum value of 0xC.
The maximum time is 15360 seconds = 4Hr and 16 minutes

Additional command byte for FHT 8V valve.
For command 0x0: this cmd2 byte contains the valve position 0x00-0xFF = 0 to 100%
For command 0x1: 0x00
For command 0x2: 0x00
For command 0x6: this cmd2 byte contains the valve position 0x00-0xFF = 0 to 100%
For command 0x8: this cmd2 byte contains in bit 5-0 the offset valve position. Bit 7 is the direction.
For command 0xA: this cmd2 byte contains the valve position 0x00-0xFF = 0 to 100%
For command 0xC: this cmd2 byte contains in bit 7-1 the countdown value in seconds. Bit 0=1
For command 0xE: 0x00
For command 0xF: this cmd2 byte contains in bit 5-0 the offset valve position. Bit 7 is the direction.

Additional command byte for FHT80 door/window sensor
Not use and always 0x00

rssi:
Signal strength.
0x0 to 0xF = weak to strong

FHT 8V valve operations
The FHT 8V will go in receive mode for a few seconds only with an interval period of about 2 minutes.
The interval period is defined by the lower 3 bits of the house code using this formula:
115010 + ((hc2 & 0x07) * 0.5) seconds.
The interval period will be 115.01 to 118.51 seconds depending on the house code.

10.43. 0x7F: RAW transmit

10.43.1. 0x00: RAW transmit

Used by: transmitter only

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   repeat;
        struct {
                BYTE   uint_msb;
                BYTE   uint_lsb;
        } pulse[125];
}RAW;
```

packetlength:
Packet length (this byte not included) = number of uint-msb + number of uint-lsb + 4
Max packetlength is 125 * 2 + 4 = 254

packettype:
0x7F = RAW transmit

subtype:
0x00 RAW transmit

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

repeat:
repeat count for RF transmit. Max value is 255 but do not use a too high repeat count to avoid other sensor signal disturbance. A repeat of 5 to 10 is acceptable.

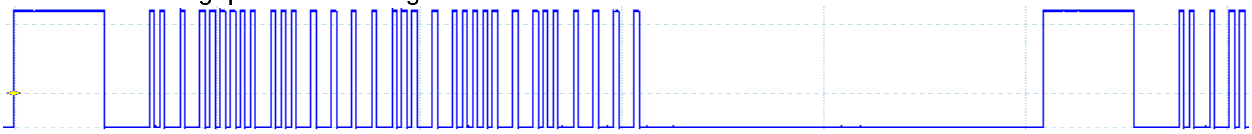
uint:
uint_msb and uint_lsb = pulse timing in usec.
Odd value is for pulse high, even value is for pulse low.
A multiple of 4 bytes is mandatory.

Example:
9000usec high (is hex 0x2328)
4500usec low (is hex 0x1194)

Transmit bytes:
0x23
0x28
0x11
0x94

Example X10 A2-OFF command:

An X10 RF A2-OFF command is:
0x60 0x9F 0x30 0xCF = 0110 0000 1001 1111 0011 0000 1100 1111
A 0 bit is 500usec high – 500usec low
A 1 bit is 500usec high – 1500usec low
The inter record gap is 500usec high – 40000usec low



```
------------------------------------------------
RAW transmit command:8C 7F 00 0A 05 23 28 11 94 01 F4 01 F4 01 F4 05 DC 01 F4 05 DC 01 F4 01 F4
01 F4 01 F4 01 F4 01 F4 01 F4 01 F4 01 F4 01 F4 01 F4 05 DC 01 F4 01 F4 01 F4 01 F4 01 F4 05 DC
01 F4 05 DC 01 F4 05 DC 01 F4 05 DC 01 F4 05 DC 01 F4 01 F4 01 F4 01 F4 01 F4 05 DC 01 F4 05 DC
01 F4 01 F4 01 F4 01 F4 01 F4 01 F4 01 F4 01 F4 01 F4 05 DC 01 F4 05 DC 01 F4 01 F4 01 F4 01 F4
01 F4 05 DC 01 F4 05 DC 01 F4 05 DC 01 F4 05 DC 01 F4 9C 40
------------------------------------------------
0402010A00
Packettype          = Receiver/Transmitter Message
subtype             = Transmitter Response
Sequence nbr        = 10
response            = ACK, data correct transmitted
```

A2_OFF.txt file contents with decimal numbers can be used in RXmngr for a RAW transmit command:

- First value is RF repeat count = 5
- 9000 – 4500 = pulse 9000usec high and 4500usec low, is hex 0x23 0x28 0x11 0x94
- 500 – 500 = pulse 500usec high and 500usec low (is a zero RF bit) , is hex 0x01 0xF4 0x01 0xF4
- 500 – 1500 = pulse 500usec high and 1500usec low (is a one RF bit), is hex 0x01 0xF4 0x05 0xDC

5
9000
4500
500
500
500
1500
500
1500
500
500
500
500
500
500
500
500
500
500
500
1500
500
500
500
500
500
1500
500
1500
500
1500
500
1500
500
1500

500
500
500
500
500
1500
500
1500
500
500
500
500
500
500
500
500
500
1500
500
1500
500
500
500
500
500
1500
500
1500
500
1500
500
1500
500
40000

11. Message format for I/O lines.

11.1. Types 0x80: I/O lines

11.1.1. 0x01: Configure I/O lines

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   direction;
        BYTE   pull_up;
}IO_CONFIG;
```

packetlength:
Packet length (this byte not included) = 0x05

packettype:
0x80 = Configure I/O lines

subtype:
0x01 = Configur I/O lines

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

direction:
Direction of IO7 to IO0.
Input, bit=1
Output, bit =0

pull_up:
bit = 1, pull_up enabled.

11.1.2. 0x02: Control output lines

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   mask;
        BYTE   io_value;
}IO_OUTPUT;
```

packetlength:
Packet length (this byte not included) = 0x04

packettype:
0x80 = control I/O lines

subtype:
0x02 = Control output line(s)

mask:
I/O7 to I/O0.
bit=1, control this line
bit =0, do not change the output line state

io_value:
0 = set line low
1 = set line high.

11.1.3. 0x03: Get I/O line status

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;
        BYTE   direction;
        BYTE   io_value;
}IO_STATE;
```

packetlength:
Packet length (this byte not included) = 0x05

packettype:
0x80 = I/O lijnen

subtype:
0x03 = I/O line status

seqnbr:
Sequence number. This field contains a sequence number from 0x00 to 0xFF.

direction:
Direction of IO7 - IO0.
Input, bit=1
Output, bit =0

io_value:
The value is 0x00 when transmitted.
The response contains the status of IO7 – IO0.
0 = the line is low.
1 = the line is high.

Notes;
Input I/O lines send a packet on value change.
The status can be polled to see if the I/O interface is still on-line.

12. Message format reserved

12.1. Types 0x90 – 0xEF: reserved

12.1.1. 0x01: to be defined

```
struct {
        BYTE   packetlength;
        BYTE   packettype;
        BYTE   subtype;
        BYTE   seqnbr;

        to be defined!

        BYTE   battery_level : 4;
        BYTE   rssi : 4;
}?;
```

packettype:
0x90 = to be defined!

13. Appendix.

13.1. Remote commands

13.1.1. X10 RF Remote

Dec	Hex	Button
2	02	0
18	12	8
34	22	4
56	38	Rewind
58	3A	Info
64	40	CHAN+
66	42	2
82	52	Ent
96	60	VOL+
98	62	6
99	63	Stop
100	64	Pause
112	70	Cursor-left
113	71	Cursor-right
114	72	Cursor-up
115	73	Cursor-down
116	74	Cursor-up-left
117	75	Cursor-up-right
118	76	Cursor-down-right
119	77	Cursor-down-left
120	78	left mouse
121	79	left mouse-End
123	7B	Drag
124	7C	right mouse
125	7D	right mouse-End
130	82	1
146	92	9
160	A0	MUTE
162	A2	5
176	B0	Play
182	B6	Menu
184	B8	Fast Forward
186	BA	A+B
192	C0	CHAN-
194	C2	3
201	C9	Exit
209	D1	MP3
210	D2	DVD
211	D3	CD
212	D4	PC / Shift-4
213	D5	Shift-5
214	D6	Shift-Ent
215	D7	Shift-Teletext
216	D8	Text
217	D9	Shift-Text
224	E0	VOL-
226	E2	7
242	F2	Teletext
255	FF	Record

13.1.2. ATI Remote Wonder

Dec	Hex	Button
0	00	A
1	01	B
2	02	power
3	03	TV
4	04	DVD
5	05	?
6	06	Guide
7	07	Drag
8	08	VOL+
9	09	VOL-
10	0A	MUTE
11	0B	CHAN+
12	0C	CHAN-
13	0D	1
14	0E	2
15	0F	3
16	10	4
17	11	5
18	12	6
19	13	7
20	14	8
21	15	9
22	16	txt
23	17	0
24	18	snapshot ESC
25	19	C
26	1A	^
27	1B	D
28	1C	TV/RADIO
29	1D	<
30	1E	OK
31	1F	>
32	20	<-
33	21	E
34	22	v
35	23	F
36	24	Rewind
37	25	Play
38	26	Fast forward
39	27	Record
40	28	Stop
41	29	Pause
44	2C	TV
45	2D	VCR
46	2E	RADIO
47	2F	TV Preview
48	30	Channel list
49	31	Video Desktop
50	32	red
51	33	green
52	34	yellow
53	35	blue
54	36	rename TAB
55	37	Acquire image
56	38	edit image
57	39	Full screen
58	3A	DVD Audio
112	70	Cursor-left
113	71	Cursor-right
114	72	Cursor-up
115	73	Cursor-down
116	74	Cursor-up-left
117	75	Cursor-up-right
118	76	Cursor-down-right
119	77	Cursor-down-left
120	78	V
121	79	V-End
124	7C	X
125	7D	X-End

13.1.3. ATI Remote Wonder Plus

Dec	Hex	Button
0	00	A
1	01	B
2	02	power
3	03	TV
4	04	DVD
5	05	?
6	06	Guide
7	07	Drag
8	08	VOL+
9	09	VOL-
10	0A	MUTE
11	0B	CHAN+
12	0C	CHAN-
13	0D	1
14	0E	2
15	0F	3
16	10	4
17	11	5
18	12	6
19	13	7
20	14	8
21	15	9
22	16	txt
23	17	0
24	18	Open Setup Menu
25	19	C
26	1A	^
27	1B	D
28	1C	FM
29	1D	<
30	1E	OK
31	1F	>
32	20	Max/Restore Window
33	21	E
34	22	v
35	23	F
36	24	Rewind
37	25	Play
38	26	Fast forward
39	27	Record
40	28	Stop
41	29	Pause
42	2A	TV2
43	2B	Clock
44	2C	TV
45	2D	ATI
46	2E	RADIO
47	2F	TV Preview
48	30	Channel list
49	31	Video Desktop
50	32	red
51	33	green
52	34	yellow
53	35	blue
54	36	rename TAB
55	37	Acquire image
56	38	edit image
57	39	Full screen
58	3A	DVD Audio
112	70	Cursor-left
113	71	Cursor-right
114	72	Cursor-up
115	73	Cursor-down
116	74	Cursor-up-left
117	75	Cursor-up-right
118	76	Cursor-down-right
119	77	Cursor-down-left
120	78	Left Mouse Button
121	79	V-End
124	7C	Right Mouse Button
125	7D	X-End

13.1.4. ATI Remote Wonder II

Dec	Hex	Button
0	00	A
1	01	B
2	02	power
3	03	TV
4	04	DVD
5	05	?
6	06	¿
7	07	Drag
8	08	VOL+
9	09	VOL-
10	0A	MUTE
11	0B	CHAN+
12	0C	CHAN-
13	0D	1
14	0E	2
15	0F	3
16	10	4
17	11	5
18	12	6
19	13	7
20	14	8
21	15	9
22	16	txt
23	17	0
24	18	Open Setup Menu
25	19	C
26	1A	^
27	1B	D
28	1C	TV/RADIO
29	1D	<
30	1E	OK
31	1F	>
32	20	Max/Restore Window
33	21	E
34	22	v
35	23	F
36	24	Rewind
37	25	Play
38	26	Fast forward
39	27	Record
40	28	Stop
41	29	Pause
45	2D	ATI
42	3B	PC
43	3C	AUX1
44	3D	AUX2
45	3E	AUX3
46	3F	AUX4
112	70	Cursor-left
113	71	Cursor-right
114	72	Cursor-up
115	73	Cursor-down
116	74	Cursor-up-left
117	75	Cursor-up-right
118	76	Cursor-down-right
119	77	Cursor-down-left
120	78	Left Mouse Button
124	7C	Right Mouse Button

13.1.5. Medion Remote

Dec	Hex	Button
0	00	Mute
1	01	B
2	02	power
3	03	TV
4	04	DVD
5	05	Photo
6	06	Music
7	07	Drag
8	08	VOL-
9	09	VOL+
10	0A	MUTE
11	0B	CHAN+
12	0C	CHAN-
13	0D	1
14	0E	2
15	0F	3
16	10	4
17	11	5
18	12	6
19	13	7
20	14	8
21	15	9
22	16	txt
23	17	0
24	18	snapshot ESC
25	19	DVD MENU
26	1A	^
27	1B	Setup
28	1C	TV/RADIO
29	1D	<
30	1E	OK
31	1F	>
32	20	<-
33	21	E
34	22	v
35	23	F
36	24	Rewind
37	25	Play
38	26	Fast forward
39	27	Record
40	28	Stop
41	29	Pause
44	2C	TV
45	2D	VCR
46	2E	RADIO
47	2F	TV Preview
48	30	Channel list
49	31	Video Desktop
50	32	red
51	33	green
52	34	yellow
53	35	blue
54	36	rename TAB
55	37	Acquire image
56	38	edit image
57	39	Full screen
58	3A	DVD Audio
112	70	Cursor-left
113	71	Cursor-right
114	72	Cursor-up
115	73	Cursor-down
116	74	Cursor-up-left
117	75	Cursor-up-right
118	76	Cursor-down-right
119	77	Cursor-down-left
120	78	V
121	79	V-End
124	7C	X
125	7D	X-End

13.1.6. Remote Code table

	Medion remote		ATI RW		ATI RW+		ATI RW2		X10 RF remote	
Dec	Hex	Button	Hex	Button	Hex	Button	Hex	Button	Hex	Button
0	0	Mute	0	A	0	A	0	A		
1	1	B	1	B	1	B	1	B		
2	2	power	2	power	2	power	2	power	2 0	
3	3	TV	3	TV	3	TV	3	TV		
4	4	DVD	4	DVD	4	DVD	4	DVD		
5	5	Photo	5	?	5	?	5	?		
6	6	Music	6	Guide	6	Guide	6	¿		
7	7	Drag	7	Drag	7	Drag	7	Drag		
8	8	VOL-	8	VOL+	8	VOL+	8	VOL+		
9	9	VOL+	9	VOL-	9	VOL-	9	VOL-		
10	0A	MUTE	0A	MUTE	0A	MUTE	0A	MUTE		
11	0B	CHAN+	0B	CHAN+	0B	CHAN+	0B	CHAN+		
12	0C	CHAN-	0C	CHAN-	0C	CHAN-	0C	CHAN-		
13	0D	1	0D	1	0D	1	0D	1		
14	0E	2	0E	2	0E	2	0E	2		
15	0F	3	0F	3	0F	3	0F	3		
16	10	4	10	4	10	4	10	4		
17	11	5	11	5	11	5	11	5		
18	12	6	12	6	12	6	12	6	12 8	
19	13	7	13	7	13	7	13	7		
20	14	8	14	8	14	8	14	8		
21	15	9	15	9	15	9	15	9		
22	16	txt	16	txt	16	txt	16	txt		
23	17	0	17	0	17	0	17	0		
24	18	snapshot ESC	18	snapshot ESC	18	Open Setup Menu	18	Open Setup Menu		
25	19	DVD MENU	19	C	19	C	19	C		
26	1A	^	1A	^	1A	^	1A	^		
27	1B	Setup	1B	D	1B	D	1B	D		
28	1C	TV/RADIO	1C	TV/RADIO	1C	FM	1C	TV/RADIO		
29	1D	<	1D	<	1D	<	1D	<		
30	1E	OK	1E	OK	1E	OK	1E	OK		
31	1F	>	1F	>	1F	>	1F	>		
32	20	<-	20	<-	20	Max/Restore Window	20	Max/Restore Window		
33	21	E	21	E	21	E	21	E		
34	22	v	22	v	22	v	22	v	22 4	
35	23	F	23	F	23	F	23	F		
36	24	Rewind	24	Rewind	24	Rewind	24	Rewind		
37	25	Play	25	Play	25	Play	25	Play		
38	26	Fast forward	26	Fast forward	26	Fast forward	26	Fast forward		
39	27	Record	27	Record	27	Record	27	Record		
40	28	Stop	28	Stop	28	Stop	28	Stop		
41	29	Pause	29	Pause	29	Pause	29	Pause		
42					2A	TV2				
43					2B	Clock				
44	2C	TV	2C	TV	2C	TV				
45	2D	VCR	2D	VCR	2D	ATI	2D	ATI		
46	2E	RADIO	2E	RADIO	2E	RADIO				
47	2F	TV Preview	2F	TV Preview	2F	TV Preview				
48	30	Channel list	30	Channel list	30	Channel list				
49	31	Video Desktop	31	Video Desktop	31	Video Desktop				
50	32	red	32	red	32	red				
51	33	green	33	green	33	green				
52	34	yellow	34	yellow	34	yellow				

	Medion remote		ATI RW		ATI RW+		ATI RW2		X10 RF remote	
Dec	Hex	Button	Hex	Button	Hex	Button	Hex	Button	Hex	Button
53	35	blue	35	blue	35	blue				
54	36	rename TAB	36	rename TAB	36	rename TAB				
55	37	Acquire image	37	Acquire image	37	Acquire image				
56	38	edit image	38	edit image	38	edit image			38	Rewind
57	39	Full screen	39	Full screen	39	Full screen				
58	3A	DVD Audio	3A	DVD Audio	3A	DVD Audio			3A	Info
59							3B	PC		
60							3C	AUX1		
61							3D	AUX2		
62							3E	AUX3		
63							3F	AUX4		
64									40	CHAN+
66									42	2
82									52	Ent
96									60	VOL+
98									62	6
99									63	Stop
100									64	Pause
112	70	Cursor-left	70	Cursor-left	70	Cursor-left	70	Cursor-left	70	Cursor-left
113	71	Cursor-right	71	Cursor-right	71	Cursor-right	71	Cursor-right	71	Cursor-right
114	72	Cursor-up	72	Cursor-up	72	Cursor-up	72	Cursor-up	72	Cursor-up
115	73	Cursor-down	73	Cursor-down	73	Cursor-down	73	Cursor-down	73	Cursor-down
116	74	Cursor-up-left	74	Cursor-up-left	74	Cursor-up-left	74	Cursor-up-left	74	Cursor-up-left
117	75	Cursor-up-right	75	Cursor-up-right	75	Cursor-up-right	75	Cursor-up-right	75	Cursor-up-right
118	76	Cursor-down-right	76	Cursor-down-right	76	Cursor-down-right	76	Cursor-down-right	76	Cursor-down-right
119	77	Cursor-down-left	77	Cursor-down-left	77	Cursor-down-left	77	Cursor-down-left	77	Cursor-down-left
120	78	V	78	V	78	Left Mouse Button	78	Left Mouse Button	78	left mouse
121	79	V-End	79	V-End	79	V-End			79	left mouse-End
123									7B	Drag
124	7C	X	7C	X	7C	Right Mouse Button	7C	Right Mouse Button	7C	right mouse
125	7D	X-End	7D	X-End	7D	X-End			7D	right mouse-End
130									82	1
146									92	9
160									A0	MUTE
162									A2	5
176									B0	Play
182									B6	Menu
184									B8	Fast Forward
186									BA	A+B
192									C0	CHAN-
194									C2	3
201									C9	Exit
209									D1	MP3
210									D2	DVD
211									D3	CD
212									D4	PC / Shift-4
213									D5	Shift-5
214									D6	Shift-Ent
215									D7	Shift-Teletext
216									D8	Text
217									D9	Shift-Text
224									E0	VOL-
226									E2	7
242									F2	Teletext
255									FF	Record

13.2. Harrison address conversion to switch settings

The address used is converted to the address selected in the Harrison curtain motor using the table below.

switch	1 2 3 4		5 6 7 8
	H H H H		X X X X
A	0 1 1 0	1	0 0 0 0
B	0 1 1 1	2	0 0 0 1
C	0 1 0 0	3	0 0 1 0
D	0 1 0 1	4	0 0 1 1
E	1 0 0 0	5	0 1 0 0
F	1 0 0 1	6	0 1 0 1
G	1 0 1 0	7	0 1 1 0
H	1 0 1 1	8	0 1 1 1
I	1 1 1 0	9	1 0 0 0
J	1 1 1 1	10	1 0 0 1
K	1 1 0 0	11	1 0 1 0
L	1 1 0 1	12	1 0 1 1
M	0 0 0 0	13	1 1 0 0
N	0 0 0 1	14	1 1 0 1
O	0 0 1 0	15	1 1 1 0
P	0 0 1 1	16	1 1 1 1

H H H H = House code
X X X X = device code

Switch position in the motor:
Up = 1
Middle = not used!!!!
Down = 0

Examples:
If you assign the address E7 (1000 0110) to the curtain motor then set the switches to: 1=up, 2=down, 3=down, 4=down, 5=down, 6=up, 7=up, 8=down

If you assign the address A2 (0110 0001) to the curtain motor then set the switches to: 1=down, 2=up, 3=up, 4=down, 5=down, 6=down, 7=down, 8=up

13.3. Flamingo, AB400, IMPULS switch settings

	1 2 3 4		5 6 7 8 9 10		5 6 7 8 9 10 <== switches
HC	==========	DC	==========	DC	==========
A	0 0 0 0	1	0 0 0 0 0 0	33	0 0 0 0 0 1
B	0 0 0 1	2	0 0 0 1 0 0	34	0 0 0 1 0 1
C	0 0 1 0	3	0 0 1 0 0 0	35	0 0 1 0 0 1
D	0 0 1 1	4	0 0 1 1 0 0	36	0 0 1 1 0 1
E	0 1 0 0	5	0 1 0 0 0 0	37	0 1 0 0 0 1
F	0 1 0 1	6	0 1 0 1 0 0	38	0 1 0 1 0 1
G	0 1 1 0	7	0 1 1 0 0 0	39	0 1 1 0 0 1
H	0 1 1 1	8	0 1 1 1 0 0	40	0 1 1 1 0 1
I	1 0 0 0	9	1 0 0 0 0 0	41	1 0 0 0 0 1
J	1 0 0 1	10	1 0 0 1 0 0	42	1 0 0 1 0 1
K	1 0 1 0	11	1 0 1 0 0 0	43	1 0 1 0 0 1
L	1 0 1 1	12	1 0 1 1 0 0	44	1 0 1 1 0 1
M	1 1 0 0	13	1 1 0 0 0 0	45	1 1 0 0 0 1
N	1 1 0 1	14	1 1 0 1 0 0	46	1 1 0 1 0 1
O	1 1 1 0	15	1 1 1 0 0 0	47	1 1 1 0 0 1
P	1 1 1 1	16	1 1 1 1 0 0	48	1 1 1 1 0 1
		17	0 0 0 0 1 0	49	0 0 0 0 1 1
		18	0 0 0 1 1 0	50	0 0 0 1 1 1
		19	0 0 1 0 1 0	51	0 0 1 0 1 1
		20	0 0 1 1 1 0	52	0 0 1 1 1 1
		21	0 1 0 0 1 0	53	0 1 0 0 1 1
		22	0 1 0 1 1 0	54	0 1 0 1 1 1
		23	0 1 1 0 1 0	55	0 1 1 0 1 1
		24	0 1 1 1 1 0	56	0 1 1 1 1 1
		25	1 0 0 0 1 0	57	1 0 0 0 1 1
		26	1 0 0 1 1 0	58	1 0 0 1 1 1
		27	1 0 1 0 1 0	59	1 0 1 0 1 1
		28	1 0 1 1 1 0	60	1 0 1 1 1 1
		29	1 1 0 0 1 0	61	1 1 0 0 1 1
		30	1 1 0 1 1 0	62	1 1 0 1 1 1
		31	1 1 1 0 1 0	63	1 1 1 0 1 1
		32	1 1 1 1 1 0	64	1 1 1 1 1 1

Examples:

A1	0 0 0 0 0 0 0 0 0 0
A15	0 0 0 0 1 1 1 0 0 0
N2	1 1 0 1 0 0 0 1 0 0
N11	1 1 0 1 1 0 1 0 0 0

0 = switch off
1 = switch on

13.4. Phenix, IDK YC-4000S switch settings

HC	switch 1 2 3 4	DC	switch 5 A B C D
A	0 0 0 0	1	0 0 0 0 0
B	0 0 0 1	2	0 0 0 1 0
C	0 0 1 0	3	0 0 1 0 0
D	0 0 1 1	4	0 0 1 1 0
E	0 1 0 0	5	0 1 0 0 0
F	0 1 0 1	6	0 1 0 1 0
G	0 1 1 0	7	0 1 1 0 0
H	0 1 1 1	8	0 1 1 1 0
I	1 0 0 0	9	1 0 0 0 0
J	1 0 0 1	10	1 0 0 1 0
K	1 0 1 0	11	1 0 1 0 0
L	1 0 1 1	12	1 0 1 1 0
M	1 1 0 0	13	1 1 0 0 0
N	1 1 0 1	14	1 1 0 1 0
O	1 1 1 0	15	1 1 1 0 0
P	1 1 1 1	16	1 1 1 1 0
		17	0 0 0 0 1
		18	0 0 0 1 1
		19	0 0 1 0 1
		20	0 0 1 1 1
		21	0 1 0 0 1
		22	0 1 0 1 1
		23	0 1 1 0 1
		24	0 1 1 1 1
		25	1 0 0 0 1
		26	1 0 0 1 1
		27	1 0 1 0 1
		28	1 0 1 1 1
		29	1 1 0 0 1
		30	1 1 0 1 1
		31	1 1 1 0 1
		32	1 1 1 1 1

13.5. C defines and structures

This section has been moved to a separate file RFXtrx.h

14. Copyright notice

15. Revision history.

Version 0.0 – July 1, 2011
- Initial version.

Version 1.0 – August 30, 2011
- Interface Response changed

Version 1.1 – September 7, 2011
- Transmitter response changed
- IMPULS device added in Lighting1
- Lighting1 and Lighting2 unit codes changed (0x01 to 0x10)

Version 1.2 – September 15, 2011
- Chapter “Communication protocol with the application” added

Version 2.0 – September 18, 2011
- seqnbr field added in all messages.
- C code added
- Lighting4 changed

Version 2.1 – September 22, 2011
- KD101 also supported by the transmitter

Version 2.2 – September 28, 2011
- Thermostat3 changed.

Version 2.3 – September 30, 2011
- Packet lengths corrected
- Interface command - Disable Mertik RF added

Version 2.4 – October 1, 2011
- Receiver error report added in Interface Response
- Battery status not used for Thermostat1

Version 2.5 – October 11, 2011
- WEIGHT, CAMERA1 and REMOTE structures changed
- Structure TRESPONSE renamed to RXRESPONSE
- pTypeInterfaceControl split into 3 types:
  - pTypeInterfaceControl
  - pTypeInterfaceMessage
  - pTypeRecXmitMessage
- Rain fall rate changed for RAIN2 to mm*100/hr.

Version 2.6 – October 30, 2011
- Information about max number of Interface control-Mode Save commands added.
- Notes about reporting and alive periods added at sensors in this document.
- Visonic auxiliary contact added as type.
- LightwaveRF added

Version 2.7 December 12, 2011
- Receiver types updated in Interface response
- LaCrosse added
- Frequency select commands added in Interface control
- Communication protocol with the application rewritten.
- TH7 = TFA TS34C added

Version 2.8 December 29, 2011
Undecoded added
Version 3.0 December 31, 2011
Visonic moved to Security1
Security1 length changed and id3 added
Version 4.0 January 2, 2012
The ICMND Interface Control Mode command has been extended with additional bytes to enable mode setting using only 1 command.
Set Mode command added.
Initialization protocol flowchart added.
Lighting2 – unit ID id1-id4 changed to format 0x3 FF FF FF.
Get Status values for msg1 corrected
Response on a mode command msg3 and 4 moved to msg4 and msg5, msg3 added
Field name IRESPONSE.msg changed to IRESPONSE.cmnd
Version 4.1 January 9, 2012
Mode command corrected with length 0x0D and msg removed
Corrected: Interface message type is 0x01
Version 4.2 January 10, 2012
RAIN1 battery level changed
Security1 changed! X10 motion and X10 remote subtypes added
Divide the CM119 total usage by 223.666 to get the kWh usage.
Version 4.3 January 16, 2012
Lighting3 changed.
id2 and id3 in Security1 changed
wind sensor packet length changed for WIND4 TFA temp values
RAIN3 added
UV3 added and temp added to UV structure
Version 4.4 January 18, 2012
Housecode in Camera1 changed to A-P
Version 4.5 January 24, 2012
Name of struct TRESPONSE changed to RXRESPONSE
Version 4.6 February 8, 2012
868.00 added to Interface Control and Interface Message
RFXSensor int msg changed to bytes
Novatys added
Version 4.7 February 13, 2012
LWRF renamed to AD
Version 4.8 February 22, 2012
Sequence number for transmit clarified.
Version 4.9 February 27, 2012
RFXrec433 added, defines changed and status added
recType43392 changed to trxType43392 !!
Version 4.10 March 5, 2012
Table added: subtype to be used for X10 security transmit commands
Version 4.11 March 12, 2012
Selected channel for weather sensors added
Version 4.12 March 13, 2012
Lighting5 inline relay and Mood4-5 commands added
Lighting5 lock command codes changed
Version 4.13 March 14, 2012
Temperature-Humidity TH8 UPM Esic sensor added
Lighting6 – Novatis removed
Version 4.14 March 19, 2012
Lighting5 – set dim level added
Version 4.15 March 29, 2012
Lighting5 – EMW100 GAO-Everflourish added
Version 4.16 April 1, 2012
Lighting1 – AB400 and IMPULS unit numbers extended 64
Version 4.17 April 12, 2012
FS20 added

Version 4.18 April 19, 2012
Weight value is reported * 10
UPM wind and rain added
Version 4.19 April 24, 2012
Rain total values are multiplied by 10 (FW version 35)
TEMP6 - TS15C added
Version 4.20 May 4, 2012
Sensor list updated
Version 4.21 May 9, 2012
RisingSun added to Lighting1
Version 4.22 May 14, 2012
BLINDS1 - RollerTrol added
TEMP7 – Viking added
Version 4.23 June 14, 2012
TEMP8, HUM2, RAIN5, WIND6 for WS2300 added
Chapter 9.2.1 Set Mode example corrected
Version 4.24 June 24, 2012
Visonic PowerCode transmit added
Version 4.25 July 1, 2012
Lighting3 - Ikea koppla system code 1-16 added
Version 4.26 July 15, 2012
TH9 Viking 02035, 02038 added
TEMP9 RUBiCSON added
Disable Ikea Koppla receive replaced by Disable Meiantech
Enable Meiantech added
Security1 – tamper reporting changed (IMPORTANT CHANGE)
Security1 – Meiantech added
Version 4.27 July 23, 2012
Lighting1 – Philips SBC added
Lighting6 – Blyss added
Interface Control
- msg3 Rubicson bit added
- msg3 AE bit added
- msg3 FineOffset/Viking added
- msg4 Rollertrol renamed to BlindsT0
- msg4 BlindsT1 bit added
- msg4 Viking moved from LaCrosse to FineOffset
BLINDS1 – Type 0x01 (Hasta old) added
BLINDS1 – Set limit added
Version 4.28 August 18, 2012
Undecoded subtypes 0x0F – 0x12 added
Version 4.29 August 25, 2012
TEMP10 for TFA 30.3133 added
ATI Remote Wonder II added with cmndtype field.
Version 4.30 August 28, 2012
TEMP10 subtype corrected to 0x0A
Version 4.31 September 8, 2012
Lighting6 - Blyss unit 5 added
Energy – ELEC3 added
Version 4.32 October 8, 2012
Lighting5 - BBSB added
Version 4.33 October 16, 2012
Thermostat1 transmit added
Humidity status dry and normal exchanged
Version 4.34 October 18, 2012
BlindsT2 and BlindsT3 added
Version 4.35 November 4, 2012
Security – IR beam added
Version 5.00 November 9, 2012
Current_Energy CM180i added
Gas and Water sensors renumbered (not yet used)
Protocol tables extended with protocol to enable for receive.

Interface Control:
disable protocol commands 0x10 to 0x1C removed
enable all RF command removed
enable display undecoded messages command removed
RFXSensor msg1-msg2 subtypes corrected

Version 5.01 November 10, 2012
Blinds1 - ID and unitcode information added

Version 5.02 November 14, 2012
ELEC2/3 total usage is in Wh (not kWh)
RFXSensor packet length is 7 (not 6)

Version 5.03 January 1, 2013
A few packet examples added
Lighting5 – subtype 0x02 BBSB has unit code 1 to 6
Lighting4 receive added
Interface Control – Mode msg3 Lighting4 bit added
Interface Message – Response msg3 Lighting4 bit added
BLINDS1 - BlindsT4 Raex added
Interface Message - Wrong Command added

Version 5.04 January 4, 2013
RFXtrx433 Type1/Type2

Version 6.00 January 6, 2013
Chapter 2: Protocol License added

Version 6.01 January 9, 2013
Security status 0x83 text corrected to “alarm delayed + tamper”
Copyright message updated

Version 6.02 February 4, 2013
Lighting6 group code E added
TH10 Rubicson added
BlindT5 – Media Mount projection screen added

Version 6.03 February 27, 2013
BlindT5 – Media Mount ID can be 0x000001 to 0xFFFFFF

Version 6.04 March 13, 2013
Lighting6 Blyss Group code supports now A to P

Version 6.05 April 30, 2013
Remote example codes added

Version 6.06 June 8, 2013
BLINDST4 RAEX ID changed to 3 bytes, Unit code is 0
Lighting5 MDREMOTE LED dimmer added www.ultraleds.co.uk
Lighting5 Conrad RSL2 added
Lighting1 Energenie ENER010 added

Version 6.07 September 27, 2013
THGR918/928 report battery level
Lighting5 LWRF colour commands added
WS1200 - 0x4F: Temperature and rain sensors added
Byron SX - 0x16: Chime added

Version 6.08 October 23, 2013
Siemens SF01 - 0x17: extractor hood added
Lighting5 – Livolo Added

Version 6.09 November 2, 2013
0x4E BBQ Maverick ET-732 added

Version 6.10 November 15, 2013
Security1 – SA30 added
TEMP_HUM – TH11 = EW109 added
POWER = Revolt added

Version 6.11 December 5, 2013
Lighting6 – Blyss rfu replaced by seqnbr2
Lighting5 - RGB TRC02 added

Version 6.12 February 11, 2014
UPM temp only sensor decoded as TEMP6

Version 6.13 March 31, 2014
- Lighting1-Energenie 5-gang added
- Lighting1-COCO GDR2-2000R added
- BlindsT6 added DC106, YOODA, Röhrmotor24 RMF
- RAW transmit added

Version 6.14 April 7, 2014
- BlindsT7 – Forest added