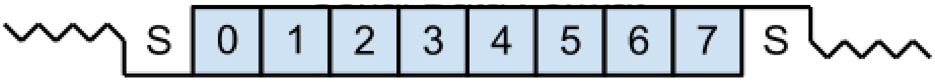
Wireless Protocol

FSK Demodulator Serial Output

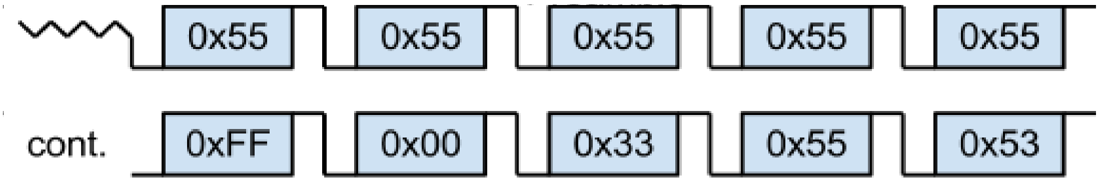
The FSK signal is modulated directly with a signal that has the same characteristics as a 38400 baud RS232 signal with one start, one stop and eight data bits. Each byte of the Preamble, End Of Block and Manchester encoded message is transmitted in this manner.



Serial Data Format

Message Preamble

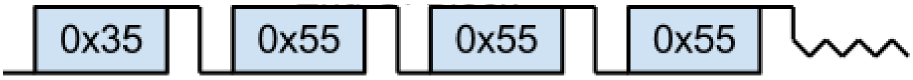
Each message is prefixed with a ten byte preamble. It is suggested that a receiver should synchronise to the sequence 0x33, 0x55, 0x53. The Preamble contains values that deliberately break the Manchester encoding scheme.



Preamble

End Of Block

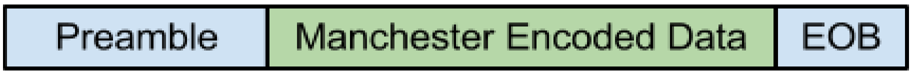
The end of message is signalled by the value 0x35. It is suggested that trailing 0x55 bytes are ignored. The End Of Block contains a value that deliberately breaks the Manchester encoding scheme.



End Of Block

Manchester Encoded Payload

Each message is contained within a preamble and end of block [to be confirmed that more than one message cannot be concatenated by some other means].



Encoded Message Block

The Manchester encoding turns each bit of original data into two bits in the encoded data block. The encoding is such that the average DC level of the signal in the FSK modem is zero.

The following table shows how to decode a pair of Manchester encoded bits to obtain one output bit.

Input	Decoded
10	0
01	1
00 or 11	Not allowed

The most straightforward way to decode the incoming bytes is to use a lookup table to convert each input byte into a four bit value. All other input values are invalid Manchester codes and the message should be considered corrupt if any other values are present.

Input	Output	Input	Output
0xAA	0x0	0x6A	0x8
0xA9	0x1	0x69	0x9
0xA6	0x2	0x66	0xA
0xA5	0x3	0x65	0xB
0x9A	0x4	0x5A	0xC
0x99	0x5	0x59	0xD
0x96	0x6	0x56	0xE
0x95	0x7	0x55	0xF

Received bytes between the Preamble and End Of Block are processed in pairs, forming the most significant and least significant nibbles of each Manchester decoder output byte. Therefore, a valid message must include an even number of bytes between the Preamble and End Of Block.

Example: input bytes 0x55, 0xA5 decode to 0xF3.

The result of this decoding process is a message that is half the length of the original Manchester encoded bytes.

Message Format

Checksum

The last byte of the Manchester decoded message is a checksum.

$$\sum_{i=0}^{len} received[i] \; \& \; 0xFF = 0x00$$

The modulo 255 sum of the bytes between the Preamble and End of Block should be zero. If this is not the case, the message should be considered corrupt.

Byte Order and Data Formats

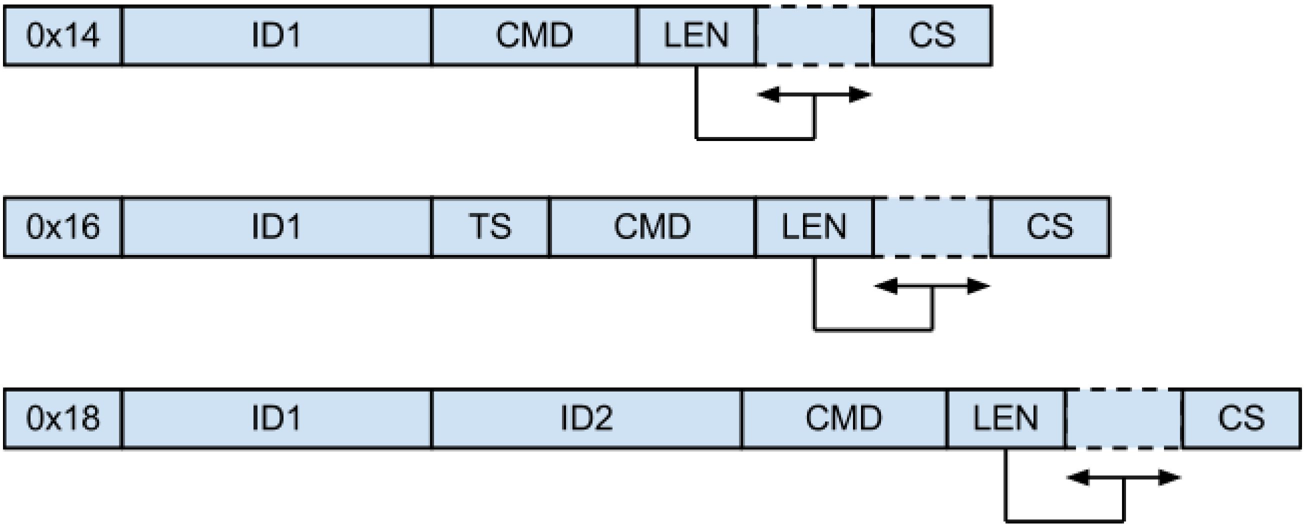
For each multi-byte (16 bit unsigned, 16 bit signed, 24 bit) values the byte order is little-endian (MSB first).

The data format for temperatures is a 16 bit signed integer value, the units are $\frac{1}{100}$ degrees Celsius.

Message Header

The header byte determines which additional content is encoded after the message header.

Three different values of the header byte have been observed:



IDn	24 bit device ID
CMD	16 bit command
LEN	8 bit message length
TS	8 bit minutes counter timestamp
CS	8 bit checksum

The meaning of the encoding of the header byte is unclear, however the three formats above have been observed.

Monitoring a wider variety of equipment may reveal other header values.

Messages

Message 0x0008

Sent from CM67z, purpose unknown.

Example Message	`16 33 3d 48 c4 00 08 02 00 00 64`
	`H  [  id  ] TS [cmd] L  [ ? ] CS`

H	Header byte
id	Originating Device Id
TS	Timestamp, incrementing in minutes
cmd	Command type
L	Length
?	Unknown
CS	Checksum

Second example, source id unknown. Maybe a neighbours device? Occurs very infrequently.

Example Message	`14 33 d2 3e 00 08 02 00 00 9f`
	`H  [  id  ] [cmd] L  [ ? ] CS`

H	Header byte
id	Originating Device Id
cmd	Command type
L	Length
?	Unknown
CS	Checksum

Message 0x0009

Sent from CM67z, purpose unknown.

Example Message	`16 33 4a b3 7b 00 09 03 00 00 ff 34`
	`H  [  id  ] TS [cmd] L  [   ?  ] CS`

H	Header byte
id	Originating Device Id
TS	Timestamp, incrementing in minutes
cmd	Command type
L	Length
?	Unknown
CS	Checksum

Message 0x000a

The CM67z sends operating parameters to the HR80's. Each zone receives its own settings, example message shows two zones.

Example Message	`16 33 4a b3 ca 00 0a 0c 01 02 01 f4 0b b8 02 02 01 f4 0b b8 63`
	`H  [  id  ] TS [cmd] L  Z  FL [min] [max] Z  FL [min] [max] CS`

H	Header byte
id	Originating Device Id
TS	Timestamp, incrementing in minutes
cmd	Command type
L	Length
Z	Zone
FL	Flags, individual bits to be confirmed (local override enable, window open....)
min	Zone minimum temperature
max	Zone maximum temperature
CS	Checksum

Message 0x1100

Sent from CM67z, purpose unknown.

Example Message	16 33 3d 48 91 11 00 08 00 18 04 04 00 00 96 01 d1 16 33 4a b3 7a 11 00 08 00 18 08 08 00 00 96 01 68 H [id] TS [cmd] L [?] CS
H	Header byte
id	Originating Device Id
TS	Timestamp, incrementing in minutes
cmd	Command type
L	Length
?	Unknown
CS	Checksum

Message 0x1f09

From CM67z, meaning unknown. Could be global parameters sent from CM67z to HR80/Boiler relay, values do not seem to change. There are lots of settings in the CM67z, this could be the means of broadcasting them.

Example Message	16 33 4a b3 43 1f 09 03 00 0a 64 de H [id] TS [cmd] L [params] CS
H	Header byte
id	Originating Device Id
TS	Timestamp, incrementing in minutes
cmd	Command type
L	Length
Params	3 bytes, encoding unknown
CS	Checksum

Message 0x2249

Information about the current and upcoming temperature setpoints, sent from CM67z. A minutes countdown accompanies the "next" temperature to allow the HR80 optimisation function to work.

Two examples, one where the CM67z controls two zones, a second where the CM67z is configured to be the zone temperature sensor for zone 1. In this case the message only conveys now/next information for zone 2.

Example Message	`16 33 3d 48 90 22 49 0e 01 04 4c 07 08 02 7d 02 05 dc 07 6c 00 61 93`
	`H  [  id  ] TS [cmd] L  Z  temp  temp  count Z  temp  temp  count CS`
	`                           now   next  down     now   next  down`
	
	`16 33 4a b3 6b 22 49 07 02 07 08 03 20 00 3f 6a`
	`H  [  id  ] TS [cmd] L  Z  temp  temp  count CS`
	`                           now   next  down`

H	Header byte
id	Originating Device Id
TS	Timestamp, incrementing in minutes
cmd	Command type
L	Length
Z	Zone
temp now	Current setpoint in programmer schedule
temp next	Next setpoint in programmer schedule
countdown	Number of minutes until next setpoint takes effect, counting down
CS	Checksum

Message 0x30c9

Sent from both CM67z and HR80, giving the actual temperature as measured. Two different forms, one from CM67z and one from HR80.

A CM67z that is set as zone temperature sensor gives a valid temperature reading, otherwise the CM67z puts 0x7fff in the temperature field.

Example Message	`From HR80`
	`18 11 c9 c1 11 c9 c1 30 c9 03 00 07 b4 fb`
	`H  [  id1 ] [  id2 ] [cmd] L  ?? temp  CS`
	
	`From CM67z`
	`16 33 4a b3 57 30 c9 03 01 08 85 d9`
	`16 33 3d 48 19 30 c9 03 01 7f ff 9e`
	`H  [  id1 ] TS [cmd] L  ?? temp  CS`

H	Header byte
id1	Originating Device Id
id2	HR80 sends it's own ID, twice
TS	Timestamp, incrementing in minutes
cmd	Command type
L	Length
??	HR80s set this byte to 0x00, CM67z set to 0x01
temp	Current measured temperature, 0x7fff for no measurement.
CS	Checksum

Message 2309:

Zone setpoint setting. Sent by a device that is controlling a temperature, either a HR80, or a CM67z acting as zone temperature sensor. The temperature indicated is either the setpoint as determined by the programmer schedule, or one overridden locally.

Example Message	From HR80 18 11 c9 c2 33 4a b3 23 09 03 01 03 20 c9 H [id] [id2] [cmd] L Z setpt CS From CM67z 16 33 4a b3 6e 23 09 03 01 03 20 f9 H [id] TS [cmd] L Z setpt CS
H	Header byte
id1	Originating Device Id
id2	Programmer that HR80 is bound to
TS	Timestamp, incrementing in minutes
cmd	Command type
L	Length
setpt	Current measured temperature, 0x7ff for no measurement.
CS	Checksum

Message 3b00:

Sent from CM67z, possibly only sent when “system timing master” is enabled in the CM67z settings.

Example Message	16 33 4a b3 03 3b 00 02 00 c8 b2 H [id] TS [cmd] L ?? ?? CS
H	Header byte
id	Originating Device Id
TS	Timestamp, incrementing in minutes
cmd	Command type
L	Length
??	Unknown, always seems to be 00 c8
CS	Checksum

Message 3150

Heat demand sent by HR80. Each HR80 sends a demand value to the boiler relay. These values are aggregated and the boiler relay determines the mark-space ratio required to fulfill the overall demand.

Example Message	18 11 c9 d0 33 4a b3 31 50 02 01 00 8a H [id1] [id2] [cmd] L Z D CS
H	Header byte
id1	Originating Device Id
id2	Programmer that HR80 is bound to
cmd	Command type
L	Length
Z	Zone
D	Demand - appears to be an unsigned number in the range 0-200
CS	Checksum

Message 1060

Purpose of this message is unknown. It is sent more than once by each HR80:

Once with the HR80 device Id repeated twice and zero in the Z field, or
Once with the HD80 device Id, the Id of the programmer the HR80 is bound to, and the zone in the Z field.

Example Message	18 12 6a 8b 12 6a 8b 10 60 03 00 ff 01 67 18 12 6a 8b 33 3d 48 10 60 03 01 ff 01 b5 18 11 c9 c1 33 3d 48 10 60 03 02 ff 01 20 H [id] [id] [cmd] L Z ?? ?? CS
H	Header byte
id1	Originating Device Id
id2	Originating Device Id, or programmer that HR80 is bound to
cmd	Command type
L	Length
Z	Zone
??	Unknown
CS	Checksum