Inside Mitsubishi M-NET

by Len Shustek

V1.0, 6 April 2015
V1.1, 12 April 2015

For decades Mitsubishi heating and heating and air conditioning units have been interconnected with a propriety network called M-NET. The company doesn't publicly document anything about it: the electrical signals, protocols, packet formats, or command sequences.

In an attempt to understand our HVAC system, I recreated my experience from 30 years ago as the designer of the first Sniffer™ at Network General for local area network protocol analysis. I figured out how the M-NET signaling works, built a Sniffer to record and decode the communications, and have made good progress on understanding the operation at the command level.

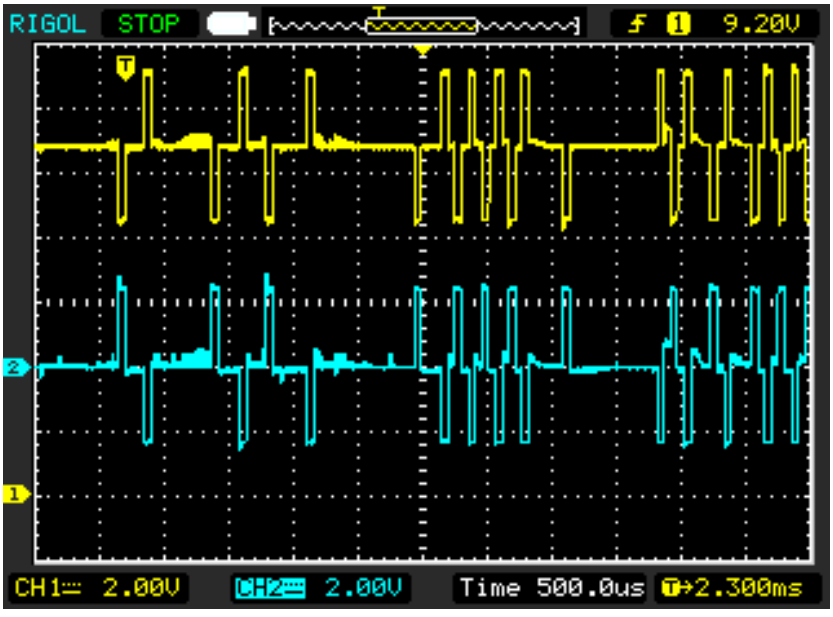
This is reverse engineering based solely on observations and experiments. I have no access whatsoever to Mitsubishi proprietary or confidential documents.

I have only examined our system, which consists of two MXZ-4B36NA outdoor heat pumps, four SEZ-KD18N14 indoor fancoil units attached to MAC-333F-E System Control Interfaces, and a CoolMaster 4000M that connects the M-NET to the Control4 home automation system.

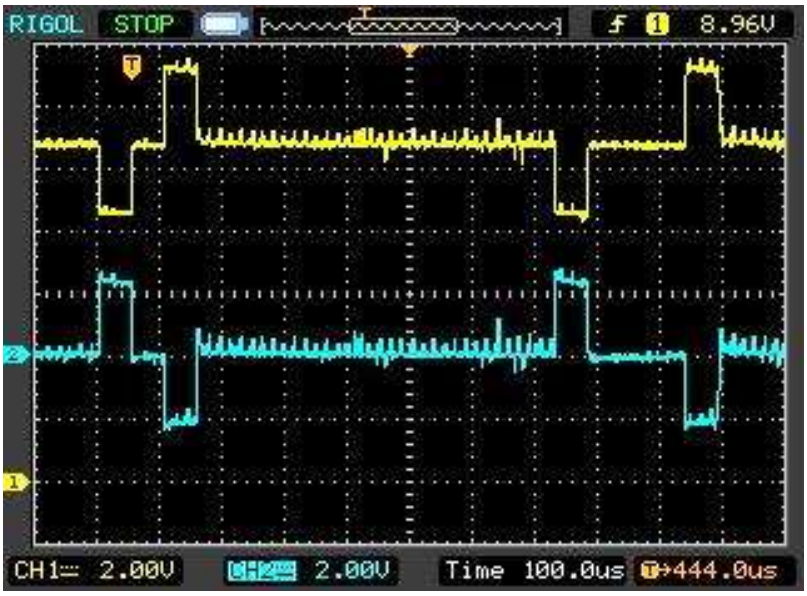
The electrical signals and coding

M-NET is a daisy-chained two-wire communications link. A voltmeter quickly shows that it carries DC power of 10 to 15 volts. The convention for which unit provides the power is not clear; in our configuration it is the CoolMaster interface.

Looking at the live M-NET cable with an oscilloscope reveals bursts of communication riding as pulse modulation with 2-volt positive and negative excursions on both legs of the power.



Looking closer, we can see non-periodic pulses just longer than 50 microseconds of opposite polarity.



Clearly they are trying to accomplish two things: (1) avoid common-mode noise by using differential transmission on the two wires, and (2) avoid voltage build-up by using a bipolar code of some kind.

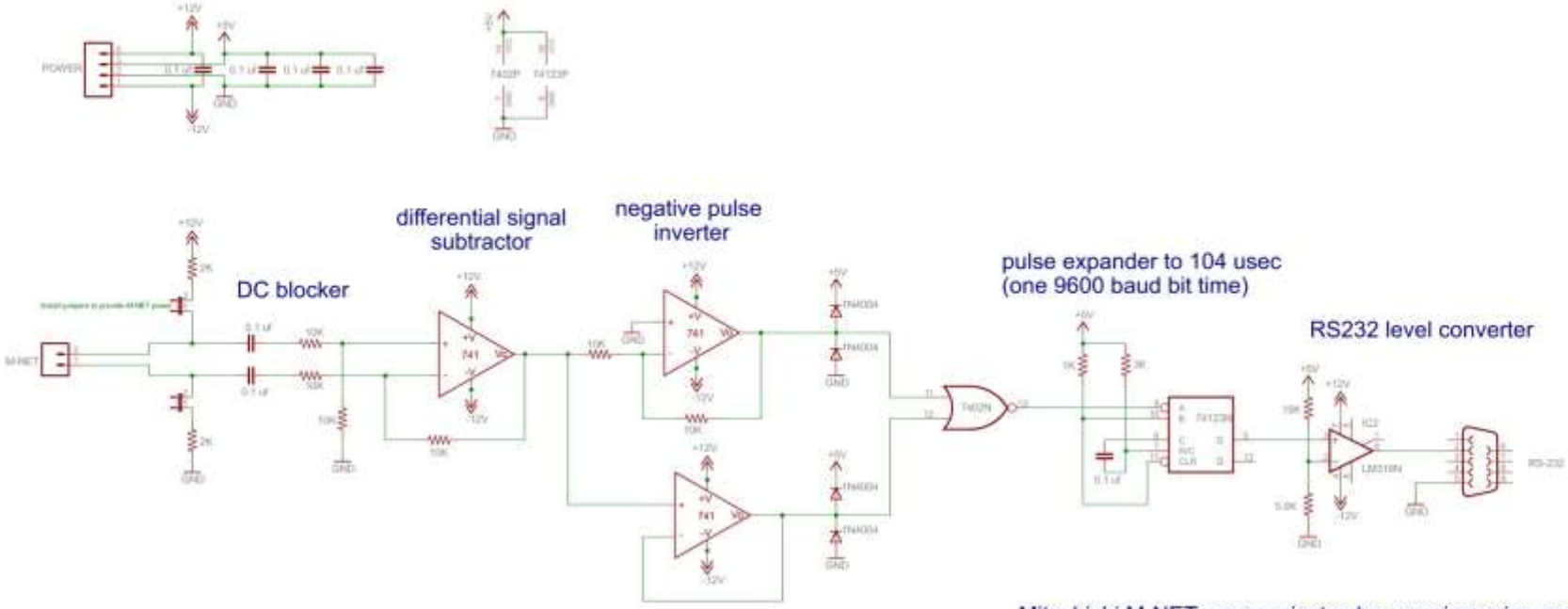
The classic bipolar encoding (see https://en.wikipedia.org/wiki/Bipolar_encoding) is alternate mark inversion, where a binary 0 is encoded as zero volts -- no pulses -- and a binary 1 ("mark") is encoded alternately as a positive or negative voltage, with the opposite polarity on the return wire.

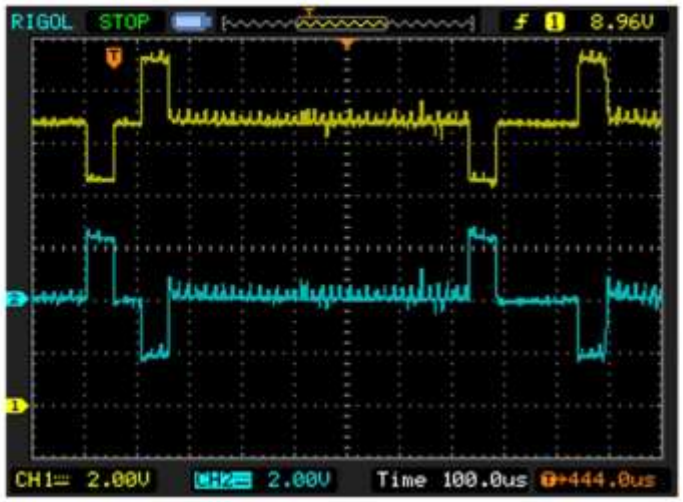
The "just longer than 50 microseconds" pulses that are always followed by a rest afterwards was a hint that they are transmitting serially at the traditional 9600 baud, which has a bit time of 104 microseconds. So each non-zero bit cell starts with a half-baud pulse of 52 microseconds.

If it were traditional serial communication, it would have to begin with a "start" bit, which is a "space", not a "mark". So this variation of bipolar coding must be alternate space inversion. And since RS232 serial communication traditionally uses "space" as a 0-bit and "mark" as a 1-bit, the alternating "space" pulses probably represent 0s within the data, and the 1s are the quiet times.

The was confirmed by manually writing down long strings of the pulses from the oscilloscope screen and noticing the following pattern on a 104 microsecond timeline: a start (space) pulse, followed by 8 data bits with a pulse representing 0 and no pulse representing 1, followed by an even parity bit, and then a single stop (mark) bit. It is a variation of standard serial communication!

The logical next step in building a Sniffer was to design a circuit that converts this bipolar signal into RS232 serial communications. All it takes is four op-amps for differential decoding and level conversion, and a monostable timer that produces 104-microsecond baud levels from the 52-microsecond pulses.



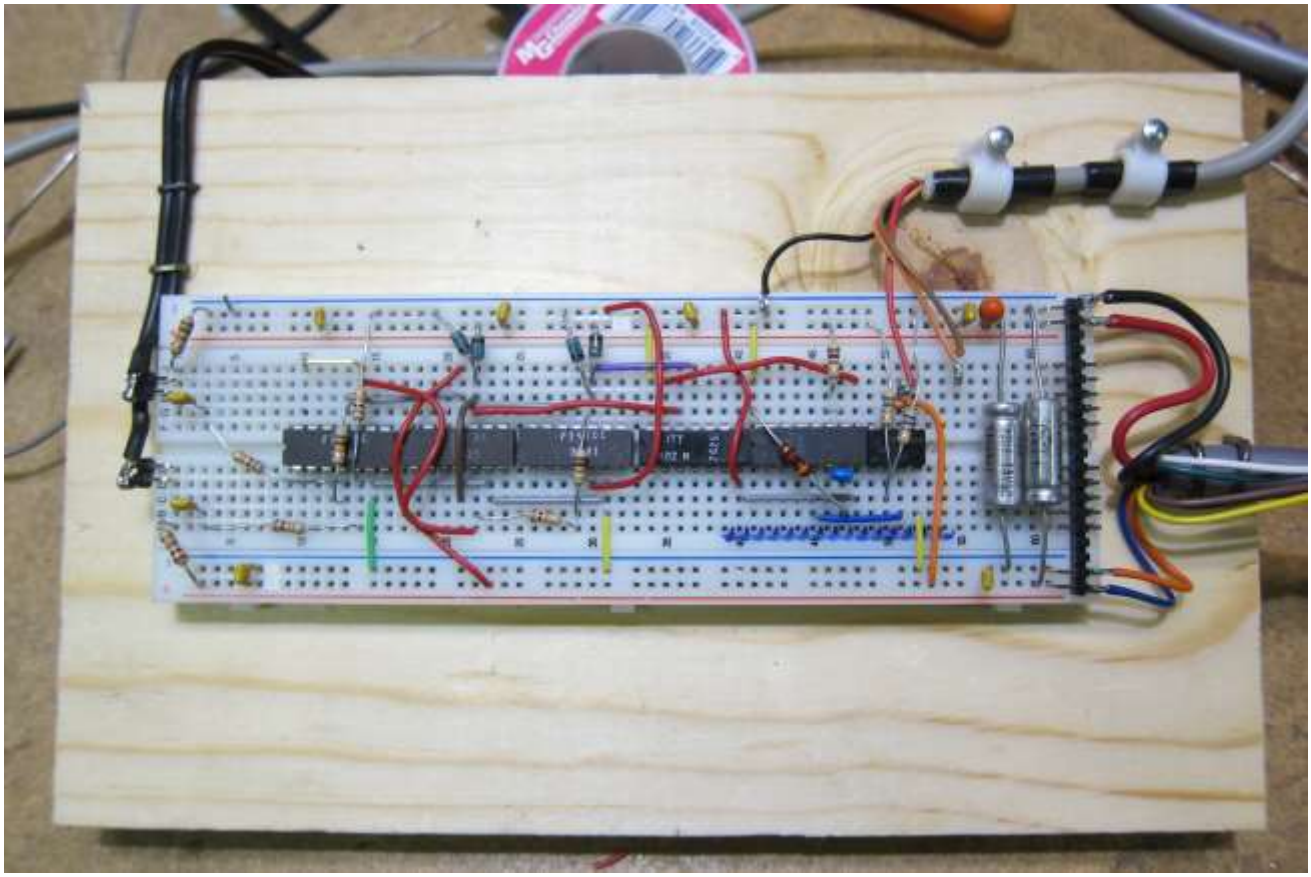


Mitsubishi M-NET communicates by superimposing an AC signal on a single pair of wires that also carries 12V of DC power.

The AC signal is transmited with "bipolar alternate space inversion" coding, so it has zero average DC component. The AC pulses are 52 usec at +2V or-2V and represent a serial data "space", which is a zero bit. Positive and negative pulses alternate for zero bits. One bits ("mark") are represented as no pulse. Transmission is at 9600 baud, so each bit takes 104 usec. The bytes are transmitted with one stop bit, 8 data bits, even parity, and one stop bit.

A sequence of 5 to 16 bytes represents a packet, which ends with an XOR checksum of the preceeding bytes.

Mitsubishi M-NET Sniffer	
TITLE:	
L. Shustek	REV:
Date: 24 March 2015	Sheet: 1/1



Built on a prototype board, this Sniffer hardware takes the M-NET signal in on the left, and outputs a standard RS232 signal on the upper right into a DB-9 connector. The output can be connected to an old-style serial port of a pre-2005 computer, or into a more modern USB port using an inexpensive RS232-to-USB converter like http://www.amazon.com/dp/B005S72HHO.

The packet format

Using a terminal emulator, I began to examine the communications I saw on our M-NET, now decoded into a series of bytes. Unfortunately the traffic doesn't use characters encoded as either ASCII or EBCDIC; it is a binary code.

But it was divided into small packets with 8 to 12 bytes, followed by a rest time. Although I can't tell electrically who is transmitting, each packet ends with a binary 06, which is the ASCII code for "ACK", or "acknowledge". So that must be transmitted by the unit receiving the packet.

I reasoned that there must be some "longitudinal" error checking of the packet in addition to the "vertical" parity on each byte, so I tried interpreting the byte before the ACK as a checksum using various algorithms. It turns out to be a modulo-256 negative sum of all the preceding bytes, so that the packet sums to zero.

A typical packet thus looks like this:

```
BD FB 02 3F 03 35 03 22 AA 06
```

The checksum AA is the last byte sent by the transmitter, followed by the 06 ACK that the receiver uses to indicate that the packet was received with a correct checksum.

There must be identification within the packet of the sender and receiver unit numbers. I knew from the CoolMaster display that our internal fancoil units are numbered from 01 to 04, and I could see that the 02 in the example packet alternates with the FB in subsequent packets. So the 02 must be the destination, and the FB the source address – in this case our CoolMaster's address. The initial byte, which I've only seen as BD or BE, must be the packet type – perhaps indicating a command vs a response.

There must be an internal encoding of the packet length, and the first 03 in the example packet above is the count of the remaining data bytes before the checksum. This packet's data payload is thus `35 03 22`.

So the general format of a packet is:

type	desti-nation	source	flags?	length	data…	check-sum	ACK

The next step was to write a protocol interpreter program to separate and decode the packets. Gradually, as I did experiments and examined the network traffic that they provoked, I was able to create decoded displays of the packets. It's a program written in C, and is currently about 500 lines long. I've posted the code (and all the other documentation) as open source at https://github.com/LenShustek/M-NET-Sniffer.

At this point I've identified the basic format of all the packets I've seen on our system. There are bits and bytes within the packets whose meaning I don't know, and there are surely other packet types I haven't yet seen.

The following listing shows the decoding of the CoolMaster's normal polling of the fancoil units, which it does about every 20 seconds. The first column is the time between the start of the packets in milliseconds, followed by the raw hex data, followed by my interpretation of the meaning. CoolMaster reads the status, mode, setpoint temperature, fan speed, and current temperature of each indoor unit in turn. The "CM" as an address indicates the CoolMaster's address, FB. For clarity in grouping commands and responses, I omit the source address in the decoding if it's the same as the destination address of the previous packet.

```
0.002  BD FB 01 3F 02 2D 01 D8 06              CM->01 get status
0.005  BE 01 FB 3F 05 2D 81 00 00 00 54 06       ->CM  stopped
0.004  BD FB 02 3F 02 2D 01 D7 06              CM->02 get status
0.002  BE 02 FB 3F 05 2D 81 00 00 00 53 06       ->CM  stopped
0.004  BD FB 03 3F 02 2D 01 D6 06              CM->03 get status
0.002  BE 03 FB 3F 05 2D 81 00 00 00 52 06       ->CM  stopped
0.004  BD FB 04 3F 02 2D 01 D5 06              CM->04 get status
0.004  BE 04 FB 3F 05 2D 81 00 00 00 51 06       ->CM  stopped
0.003  BD FB 01 3F 02 2D 02 D7 06              CM->01 get mode
0.003  BE 01 FB 3F 03 2D 82 07 4E 06             ->CM  heat
0.003  BD FB 02 3F 02 2D 02 D6 06              CM->02 get mode
0.003  BE 02 FB 3F 03 2D 82 07 4D 06             ->CM  heat
0.003  BD FB 03 3F 02 2D 02 D5 06              CM->03 get mode
0.003  BE 03 FB 3F 03 2D 82 07 4C 06             ->CM  heat
0.003  BD FB 04 3F 02 2D 02 D4 06              CM->04 get mode
0.003  BE 04 FB 3F 03 2D 82 07 4B 06             ->CM  heat
0.003  BD FB 01 3F 02 25 01 E0 06              CM->01 get setpoint temp
0.003  BE 01 FB 3F 05 25 81 01 90 00 CB 06       ->CM  19.0 deg C
0.004  BD FB 02 3F 02 25 01 DF 06              CM->02 get setpoint temp
0.003  BE 02 FB 3F 05 25 81 01 80 00 DA 06       ->CM  18.0 deg C
0.003  BD FB 03 3F 02 25 01 DE 06              CM->03 get setpoint temp
0.003  BE 03 FB 3F 05 25 81 01 80 00 D9 06       ->CM  18.0 deg C
0.003  BD FB 04 3F 02 25 01 DD 06              CM->04 get setpoint temp
0.004  BE 04 FB 3F 05 25 81 01 80 00 D8 06       ->CM  18.0 deg C
0.003  BD FB 01 3F 02 2D 0E CB 06              CM->01 get fan speed
0.003  BE 01 FB 3F 03 2D 8E 05 44 06             ->CM  medium
0.003  BD FB 02 3F 02 2D 0E CA 06              CM->02 get fan speed
0.003  BE 02 FB 3F 03 2D 8E 06 42 06             ->CM  high
0.003  BD FB 03 3F 02 2D 0E C9 06              CM->03 get fan speed
0.003  BE 03 FB 3F 03 2D 8E 0B 3C 06             ->CM  auto
0.004  BD FB 04 3F 02 2D 0E C8 06              CM->04 get fan speed
0.003  BE 04 FB 3F 03 2D 8E 0B 3B 06             ->CM  auto
0.002  BD FB 01 3F 03 35 03 22 AB 06           CM->01 get current temp
0.004  BE 01 FB 3F 05 35 83 22 02 20 06 06       ->CM  22.0 deg C
0.003  BD FB 02 3F 03 35 03 22 AA 06           CM->02 get current temp
0.003  BE 02 FB 3F 05 35 83 22 02 25 00 06       ->CM  22.5 deg C
0.004  BD FB 03 3F 03 35 03 22 A9 06           CM->03 get current temp
0.002  BE 03 FB 3F 05 35 83 22 02 20 04 06       ->CM  22.0 deg C
0.004  BD FB 04 3F 03 35 03 22 A8 06           CM->04 get current temp
0.003  BE 04 FB 3F 05 35 83 22 02 15 0E 06       ->CM  21.5 deg C
```

The next example shows a little of of the CoolMaster normal polling of the fancoil units, then what happened as, from the Control4 touchscreen, I put unit 2 in "cool" mode, changed the temperature setpoint to 17C, and then turned it off.

```
0.203  BD FB 02 3F 03 35 03 22 AA 06          CM->02 get current temp
0.515  BE 02 FB 3F 05 35 83 22 02 25 00 06      ->CM  22.5 deg C
0.610  BD FB 03 3F 03 35 03 22 A9 06          CM->03 get current temp
0.500  BE 03 FB 3F 05 35 83 22 02 35 EF 06      ->CM  23.5 deg C
0.609  BD FB 04 3F 03 35 03 22 A8 06          CM->04 get current temp
0.516  BE 04 FB 3F 05 35 83 22 02 30 F3 06      ->CM  23.0 deg C
0.609  BD FB 02 3F 05 0D 01 01 00 00 F3 06    CM->02 turn on
0.609  BE 02 FB 3F 03 0D 81 00 75 06            ->CM  ok
0.500  BD FB 02 3F 03 0D 02 08 ED 06            ->02 set mode cool
0.516  BE 02 FB 3F 03 0D 82 00 74 06            ->CM  ok
0.500  BD FB 02 3F 05 05 01 02 10 00 EA 06      ->02 set temp 21.0 deg C
0.609  BE 02 FB 3F 03 05 81 00 7D 06            ->CM  ok
0.516  BD FB 02 3F 05 05 01 02 00 00 FA 06      ->02 set temp 20.0 deg C
0.609  BE 02 FB 3F 03 05 81 00 7D 06            ->CM  ok
0.500  BD FB 02 3F 05 05 01 01 90 00 6B 06      ->02 set temp 19.0 deg C
0.610  BE 02 FB 3F 03 05 81 00 7D 06            ->CM  ok
0.515  BD FB 02 3F 05 05 01 01 90 00 6B 06      ->02 set temp 19.0 deg C
0.610  BE 02 FB 3F 03 05 81 00 7D 06            ->CM  ok
0.500  BD FB 02 3F 05 05 01 01 80 00 7B 06      ->02 set temp 18.0 deg C
0.609  BE 02 FB 3F 03 05 81 00 7D 06            ->CM  ok
0.516  BD FB 02 3F 05 05 01 01 80 00 7B 06      ->02 set temp 18.0 deg C
0.609  BE 02 FB 3F 03 05 81 00 7D 06            ->CM  ok
0.500  BD FB 02 3F 05 05 01 01 70 00 8B 06      ->02 set temp 17.0 deg C
0.610  BE 02 FB 3F 03 05 81 00 7D 06            ->CM  ok
0.515  BD FB 02 3F 05 05 01 01 70 00 8B 06      ->02 set temp 17.0 deg C
0.610  BE 02 FB 3F 03 05 81 00 7D 06            ->CM  ok
0.500  BD FB 01 3F 02 2D 01 D8 06             CM->01 get status
0.453  BE 01 FB 3F 05 2D 81 00 00 00 54 06      ->CM  stopped
0.609  BD FB 02 3F 02 2D 01 D7 06             CM->02 get status
0.469  BE 02 FB 3F 05 2D 81 01 00 00 52 06      ->CM  running
0.609  BD FB 03 3F 02 2D 01 D6 06             CM->03 get status
0.453  BE 03 FB 3F 05 2D 81 00 00 00 52 06      ->CM  stopped
0.610  BD FB 04 3F 02 2D 01 D5 06             CM->04 get status
0.453  BE 04 FB 3F 05 2D 81 00 00 00 51 06      ->CM  stopped
0.609  BD FB 01 3F 02 2D 02 D7 06             CM->01 get mode
0.453  BE 01 FB 3F 03 2D 82 07 4E 06            ->CM  heat
0.516  BD FB 02 3F 02 2D 02 D6 06             CM->02 get mode
0.453  BE 02 FB 3F 03 2D 82 08 4C 06            ->CM  cool
0.516  BD FB 03 3F 02 2D 02 D5 06             CM->03 get mode
0.453  BE 03 FB 3F 03 2D 82 07 4C 06            ->CM  heat
0.500  BD FB 04 3F 02 2D 02 D4 06             CM->04 get mode
0.469  BE 04 FB 3F 03 2D 82 07 4B 06            ->CM  heat
0.500  BD FB 01 3F 02 25 01 E0 06             CM->01 get setpoint temp
0.453  BE 01 FB 3F 05 25 81 01 90 00 CB 06      ->CM  19.0 deg C
0.609  BD FB 02 3F 02 25 01 DF 06             CM->02 get setpoint temp
0.469  BE 02 FB 3F 05 25 81 02 10 00 49 06      ->CM  21.0 deg C
0.609  BD FB 03 3F 02 25 01 DE 06             CM->03 get setpoint temp
0.453  BE 03 FB 3F 05 25 81 01 80 00 D9 06      ->CM  18.0 deg C
0.610  BD FB 04 3F 02 25 01 DD 06             CM->04 get setpoint temp
0.453  BE 04 FB 3F 05 25 81 01 80 00 D8 06      ->CM  18.0 deg C
0.609  BD FB 01 3F 02 2D 0E CB 06             CM->01 get fan speed
0.469  BE 01 FB 3F 03 2D 8E 05 44 06            ->CM  medium
0.500  BD FB 02 3F 02 2D 0E CA 06             CM->02 get fan speed
0.453  BE 02 FB 3F 03 2D 8E 06 42 06            ->CM  high
0.516  BD FB 03 3F 02 2D 0E C9 06             CM->03 get fan speed
0.453  BE 03 FB 3F 03 2D 8E 0B 3C 06            ->CM  auto
0.515  BD FB 04 3F 02 2D 0E C8 06             CM->04 get fan speed
```

```
0.454  BE 04 FB 3F 03 2D 8E 0B 3B 06            ->CM  auto
0.500  BD FB 01 3F 03 35 03 22 AB 06          CM->01 get current temp
0.515  BE 01 FB 3F 05 35 83 22 02 25 01 06      ->CM  22.5 deg C
0.610  BD FB 02 3F 03 35 03 22 AA 06          CM->02 get current temp
0.500  BE 02 FB 3F 05 35 83 22 02 25 00 06      ->CM  22.5 deg C
0.609  BD FB 03 3F 03 35 03 22 A9 06          CM->03 get current temp
0.516  BE 03 FB 3F 05 35 83 22 02 35 EF 06      ->CM  23.5 deg C
0.609  BD FB 04 3F 03 35 03 22 A8 06          CM->04 get current temp
0.500  BE 04 FB 3F 05 35 83 22 02 30 F3 06      ->CM  23.0 deg C
0.609  BD FB 02 3F 05 0D 01 00 00 00 F4 06    CM->02 turn off
0.610  BE 02 FB 3F 03 0D 81 00 75 06            ->CM  ok
0.515  BD FB 01 3F 02 2D 01 D8 06             CM->01 get status
0.453  BE 01 FB 3F 05 2D 81 00 00 00 54 06      ->CM  stopped
0.610  BD FB 02 3F 02 2D 01 D7 06             CM->02 get status
0.453  BE 02 FB 3F 05 2D 81 00 00 00 53 06      ->CM  stopped
0.609  BD FB 03 3F 02 2D 01 D6 06             CM->03 get status
0.469  BE 03 FB 3F 05 2D 81 00 00 00 52 06      ->CM  stopped
0.609  BD FB 04 3F 02 2D 01 D5 06             CM->04 get status
0.454  BE 04 FB 3F 05 2D 81 00 00 00 51 06      ->CM  stopped
0.609  BD FB 01 3F 02 2D 02 D7 06             CM->01 get mode
0.453  BE 01 FB 3F 03 2D 82 07 4E 06            ->CM  heat
```

Clearly more experiments are needed to expand our knowledge of the packet formats and the protocol. But the necessary tools are now at hand.