**Ref → http://blog.jamesball.co.uk/2013/02/understanding-sma-bluetooth-protocol-in.html**

## Understanding the SMA bluetooth protocol

In my quest to create a Mac OS X program to read data from an SMA Sunny Boy inverter I've had to learn a lot about the protocol and the meaning of the different bytes. I want to record much of that here for use by others (and hopefully improvement if you can work out what some of the unidentified sections are)

If you Google for information there are various implementations of programs to read the data (mainly in C for Linux) but very little detail of the protocol itself. Most programs use an input file, based on packet sniffing an exchange between an inverter and a Sunny Beam, to look for key requests and make the 'correct' response rather than understanding the protocol.

There are two 'levels' to the protocol and this distinction is not well documented online. I also failed to find tabled information on the bytes of the protocol, possible values and meaning.

So, this is what I know.

Much of the protocol is clearly based on SMA Net which is published. All transfers are little endian.

## Connecting to the inverter

When you connect to the inverter you are talking in 'Level 1' (L1) packets. I will create a separate post to detail the full make up of a packet but you have: header, source address, destination address, command and data.

When you first connect to the inverter you will start receiving L1 packets. The source address is the inverter bluetooth MAC address, the destination address is 00:00:00:00:00:00 with a command 0x0200. The data will be set to <00 04 70 00 02 00 00 00 00 01 00 00 00>. I don't know the meaning of the data here - possibly at least one byte is the Net ID but I'm not sure.

This packet will be sent repeatedly up to 10 times and then the connection will close. To keep the connection open you need to return the correct response. This response is to return the data you received back to the inverter. Source address should be 00:00:00:00:00:00, destination address is the inverter MAC address and data is <00 04 70 00 02 00 00 00 00 01 00 00 00>.

If the inverter accepts your response you will receive three packets.

1) From the inverter MAC to 00:00:00:00:00:00, command 0x0a00 with the data set to the inverter MAC, 0x01, then your computer MAC.

2) From the inverter MAC to 00:00:00:00:00:00, command 0x0c00 with the data set to 0x0200

3) From the inverter MAC to 00:00:00:00:00:00, command 0x0500 with the data set to the inverter MAC, 0x0101 then your computer MAC, 0x0201

You are now connected and can send your own commands and receive answers.

The easiest is a request for the bluetooth signal strength.

You create a packet with the header, your computer MAC, inverter MAC, command 0x0300 and the data set to 0x0500.

0x0300 is a command to say that we're making a request for something. That something is 0x0500 (bluetooth strength).

In response you should receive a packet.

It will have a header and come from the inverter MAC to your computer MAC. The command will be 0x0400 (telling us this is a response to our request) will the data set to <05 00 00 00 a5 00>.

The initial 0x0500 is telling us that it's bluetooth strength. I think the next two bytes are padding. Then the next byte is the value (it could be two bytes but I don't think so). So 0xa500 is 165. It's a single byte so values go from 0 to 255. To get a % 165/256 * 100. Our signal strength is 64.5%. Not bad.

If you receive a packet with the command bytes set to 0x0700 I think this means there was an error. I've only received it when I've made a mistake. The data is set to whatever you sent.

**The header**

To send a packet you have to put on the correct header and it includes the packet length and a check digit. An example: <7e 14 00 6a>

The header always starts 0x7e. If you don't have that at the start then something is wrong and you shouldn't process the packet any more.

The next byte is the packet length in bytes. It includes every byte of the packet 0x7e and on.

The next byte is always 0x00. Or has been in every packet I've looked at.

The fourth byte is the check digit. It is calculated by taking the XOR of the three other bytes of the header. so 0x7e ^ 0x14 ^ 0x00 = 0x6a

**The commands**

The ones we know about are:

- 0x0100 - Level 2 packet / end of Level 2 packet
- 0x0200 - Ping from inverter to log in
- 0x0300 - make a request

- 0x0400 - receive result of request
- 0x0500 - part of log in process
- 0x0700 - error
- 0x0800 - Level 2 part packet
- 0x0a00 - part of log in process
- 0x0c00 - part of log in process

The missing values may well be valid commands but I've not seen them when sniffing exchanges between Sunny Explorer and the inverter or as I've sent packets.

**Level 2 packets**

Level 2 (L2) packets (which I'll look at in more detail in a later post) are transferred within the data portion of L1 packets. There is a maximum length of an L1 packet - the longest I've seen is 0x6d. L2 packets can easily be bigger than this so they have to be split between multiple L1 packets.

An L1 packet is 18 bytes long before data so any L2 packet larger than 91 bytes must be split. This is done using commands 0x0100 and 0x0800.

If an L2 packet is shorter than the maximum length then it is sent as the data part of one L1 packet with command 0x0100.

If an L2 packet is longer than the maximum length then it is split at the maximum length. If the L2 packet is 100 bytes then you'll have a 91 byte section and a 9 byte section. This is sent as two L1 packets.

1. A packet with command 0x0800 and 91 bytes.
2. A packet with command 0x0100 and 9 bytes.

The full L2 packet is recreated at the destination by assembling all 0x0800 command packets and the 0x0100 packet. There may be any number of 0x0800 packets.

[This post was updated on 10th May 2013 to correct the example of calculating the check digit.]